**Electronic Submission Coversheet**

**TO BE COMPLETED BY STUDENT**

**Important** – choose one of the following statements   
**(DELETE TWO THAT DO NOT APPLY):**

 **This is my FINAL submission for this assignment.**

 **I am intending to submit to this assignment within 5 working days of the   
 submission date. (LATE Submission)**

 **I am not intending to submit and will make an extenuating circumstance  
 application.**

**By electronically submitting this work, I certify that:**

* This assignment is my own work
* It has not previously been submitted for assessment
* Where material from other sources has been used it has been acknowledged properly
* This work meets the requirement of the University’s ethics policy

**Student Name: James Catterall**

**Student Number : Q09708146**

**Faculty: MARTEC**

**Level of study: FHEQ 6**

**Course title: COMPUTER GAMES DEVELOPMENT**

**Unit title: SWD303**

**Assignment title: MSAA Report 2**

**Assignment tutor: Dr. Kalin Penev  
Word count: 2243**

**Learner request for feedback:**

**TO BE COMPLETED BY STAFF**

**Tutor feedback:**

**Areas of Strength:**

**Areas for Improvement:**

**Grade mark:**

**Submitted on time (Y/N):**

**Tutor signature:**

**Date:**

|  |
| --- |
|  |
| Mobile Software Applications and Architecture |
| Report 2 |
|  |
| **James Catterall** |
| **4/25/2014** |

|  |
| --- |
|  |

Contents

[Introduction 5](#_Toc387071153)

[Background 5](#_Toc387071154)

[Android Features 5](#_Toc387071155)

[Receivers 5](#_Toc387071156)

[SearchView 5](#_Toc387071157)

[TextView 5](#_Toc387071158)

[ImageView 5](#_Toc387071159)

[WebView 5](#_Toc387071160)

[GridView 6](#_Toc387071161)

[Loopers 6](#_Toc387071162)

[Handlers 6](#_Toc387071163)

[Activities 6](#_Toc387071164)

[Localisation 6](#_Toc387071165)

[Aims and Objectives 7](#_Toc387071166)

[Aim 7](#_Toc387071167)

[Objectives 7](#_Toc387071168)

[Requirements Specification 7](#_Toc387071169)

[User requirements 7](#_Toc387071170)

[System requirements 7](#_Toc387071171)

[Implementation and options analysis 8](#_Toc387071172)

[Application flow 8](#_Toc387071173)

[Critical Evaluation 13](#_Toc387071174)

[Aim 13](#_Toc387071175)

[Objectives 13](#_Toc387071176)

[Application Evaluation 14](#_Toc387071177)

[Recommendations 14](#_Toc387071178)

[Bibliography 15](#_Toc387071179)

[References 15](#_Toc387071180)

[Appendix A: Source Code 16](#_Toc387071181)

[FlickrFetchr.java 16](#_Toc387071182)

[GalleryItem.java 22](#_Toc387071183)

[NotificationReceiver.java 23](#_Toc387071184)

[PhotoGalleryActivity.java 24](#_Toc387071185)

[PhotoGalleryFragment.java 26](#_Toc387071186)

[PollService.java 34](#_Toc387071187)

[SingleFragmentActivity.java 39](#_Toc387071188)

[StartupReceiver.java 40](#_Toc387071189)

[ThumbnailDownloader.java 41](#_Toc387071190)

[VisibleFragment.java 45](#_Toc387071191)

[Activity\_fragment.xml 47](#_Toc387071192)

[Fragment\_photo\_gallery.xml 47](#_Toc387071193)

[Gallery\_item.xml 47](#_Toc387071194)

[Fragment\_photo\_gallery.xml 48](#_Toc387071195)

[Strings.xml 48](#_Toc387071196)

[Styles.xml 48](#_Toc387071197)

[Strings.xml (fr) 49](#_Toc387071198)

[Searchable.xml 49](#_Toc387071199)

[AndroidManifest.xml 50](#_Toc387071200)

# Introduction

## Background

The original application which was planned to be developed for this project was UniNav which was a map application to allow students to navigate Southampton Solent University, although the project was changed due to unrealistic and unfeasible estimates in regards to knowledge regarding Android application development and time estimates. The project has subsequently been changed to a Flickr image feed viewer which allows the user to view thumbnail images of a random Flickr feed or a specific feed using the advanced search feature of the application.

There is a huge demographic for users who view image websites such as Imgur, Instagram and Tumblr from their mobile devices, although there is a distinct lack of Flickr image viewers which let you view a large amount of images at a fast rate on the Google play store. Therefore the application taps into a niche market of users who want to view a large amount of specific images from Flickr using a thumbnail viewer, and the application fulfils these needs as successfully as possible.

The Flickr viewer is exceptionally easy to use due to the seamless image loading feature which allows the user to scroll vertically to load more images if required. The GUI for the application is simplistic in regards to the amount of buttons and items that the user can interact with while remaining robust, accurate and responsive.

## Android Features

### Receivers

Receivers were used to handle small tasks such as the recurring alarm in PollService which needed to be reset when the system finishes starting, therefore the receiver needed to know when the alarm should be on or off. In addition to standard receivers, dynamic broadcast receivers were used to check whether the PhotoGalleryFragment was alive as a standard receiver would always receive the intent and need another way of knowing that PhotoGalleryFragment was alive.

### SearchView

A SearchView was used in an ActionBar as an action view for the search feature of the application and allows the user to enter a search query and submit a request to the search provider (Android developers, 2012).

### TextView

TextViews were used to display text to the user regarding the name of the application and other text in the application.

### ImageView

ImageViews were a large feature of the application as there are icons that need to be displayed in addition to images loaded from resources and content providers within the application from Flickr, and is the key component of the application.

### WebView

WebViews were a significantly important feature of the application as they allowed web content to be included in the application and therefore allows for the Flickr content to be integrated. The application uses a WebView class to display the web content within the PhotoGallery.

### GridView

A GridView was used in conjunction with the WebView to dynamically display the Flickr web content in the form of images to display the Flickr images in an easy to view grid format.

### Loopers

A looper is used in the application to handle the thread’s message queue by grabbing the messages off of its message queue and performs the task it specifies (Android Programming 2013).

### Handlers

Handlers were used in the application to dynamically download and display the images from Flickr into the PhotoGallery. The Handlers were specifically used to manage the interface for creating and posting messages within the application.

### Activities

The application contains two activities the PhotoGalleryActivity which handles the Search features of the application and the SingleFragmentActivity which sets up the activity fragment and fragment container, both of the activities can be started with an Intent.

### Localisation

Localisation (Android developers 2012) of the application is in two languages, English is the default and French texts will only be displayed when the devices language is switched accordingly. The texts for the two different localisations can be found in the appropriate “strings.xml” file within the values and values-fr folders.

# Aims and Objectives

## Aim

The aim of the project is to thoroughly design and develop a Flickr Image feed application.

## Objectives

* Simple GUI for the user to interact with.
* Loading images from a Flickr feed as thumbnails.
* Displaying additional thumbnail images when the user scrolls vertically.
* Displaying additional thumbnail images per row when the user rotates the device using accelerometers.
* Search functionality using advanced techniques such as SearchView’s.
* A safe search functionality to censor explicit content.
* Text being able to be displayed in English and French using Localisation features. The default language will be English and will change to French if the user’s device is set to French.

# Requirements Specification

## User requirements

* Start the application.
* Pause the application when they exit.
* Continue using the application from where they set off.
* Scroll to display additional images.
* Search for specific images.
* Poll for new images.

## System requirements

* Display images on the screen from the Flickr feed.
* Load and display additional images on the screen when the player scrolls vertically.
* Has to react to different states which can occur during the game: onCreate(), onStart(), onResume(), onPause(), onStop(), onRestart(), onDestroy() regarding where the player currently.
* Display specific images using the search feature.
* Poll new images when the user uses the “Poll new images” button
* Rotate the screen using the accelerometer.

# Implementation and options analysis

## Application flow

**Starting the application:**

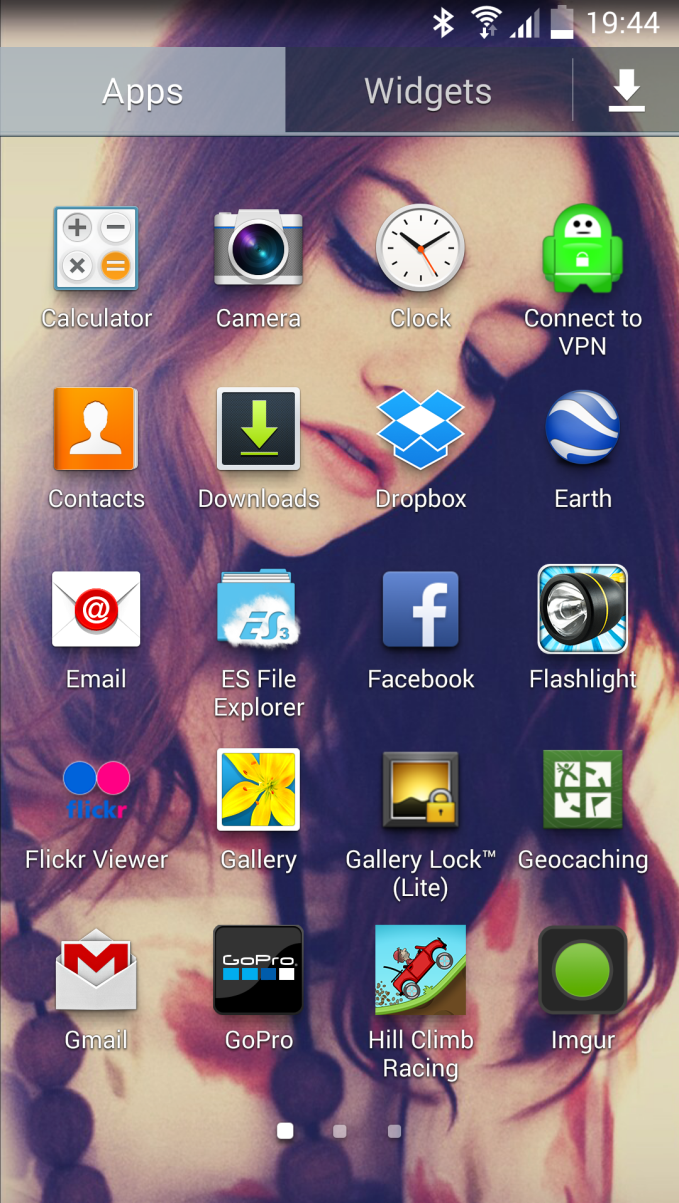
The application can be started by clicking the following icon within the applications list on the device:-

Figure 1: Application Icon.

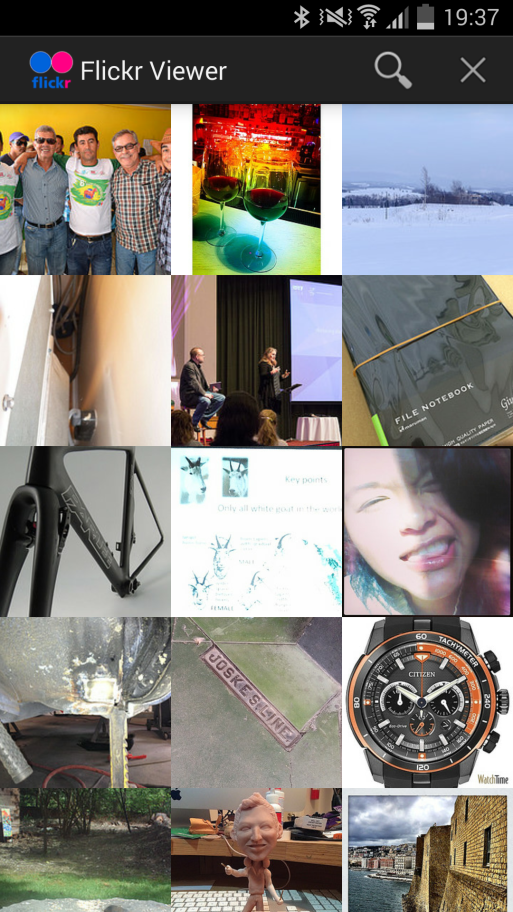
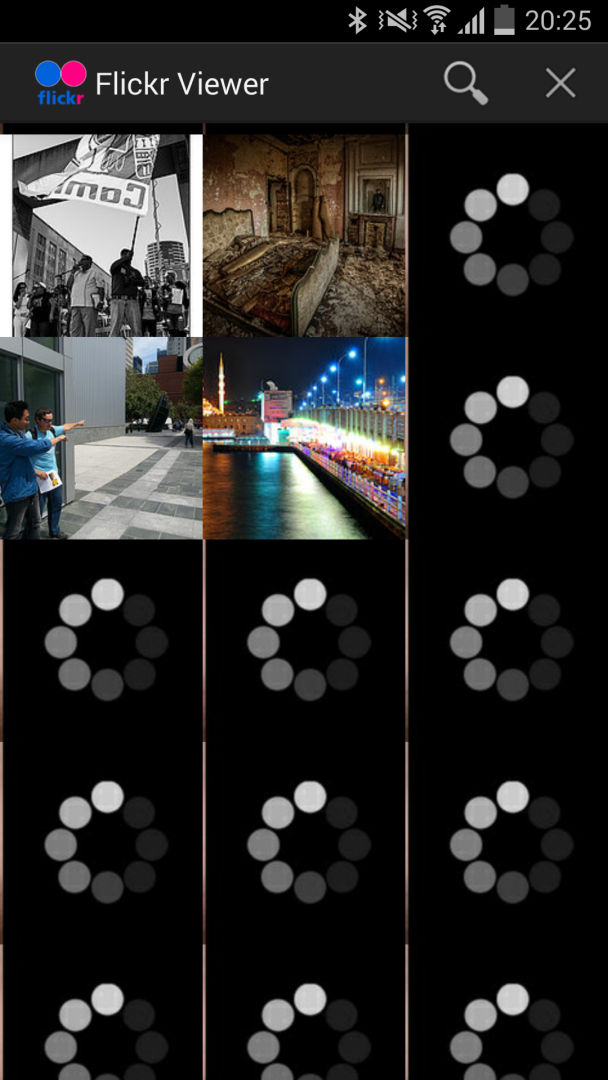
After starting the application the user will be presented with the main application window and the application will begin loading the images from the Flickr feed using the Flickr API key. The application window contains three clickable buttons, the search button, the poll new images, the stop polling button, the application icon and the application name alongside.

Figure 2: Application Window.

After the initial game window has been loaded the user is presented with 15 images using a GridView, in order to display more images from the same Flickr feed the user can swipe the screen vertically to scroll down, a buffering image is displayed while the image is loading from the feed and will eventually be replaced with the Flickr image once the image is fully loaded. The user can reload the Flickr feed and in turn load new images from the Flickr feed (essentially refreshing the Flickr feed) by clicking the “X” button in the corner of the screen. The “X” button in the corner of the screen is the “Poll for new Images” button, and once pressed will begin the process of refreshing the Flickr feed.

Figure 3: Polling for new images.

Once the user has begun the polling process for new images after using the “Poll for new Images” button a new button will appear in the right hand corner of the users screen, the button will stop the polling process if pressed and the user will be presented with their new Flickr image feed which has just been refreshed.

Figure 4: Loading images.

The buffering images are displayed when the images being loaded from the users Flickr feed and not able to be displayed. The images from the Flickr feed eventually replace the buffering image once they are fully loaded and ready to be displayed using the GridView. New images will be loaded if the user is polling for new images, scrolling down within the window to load additional images or searching for specific images.

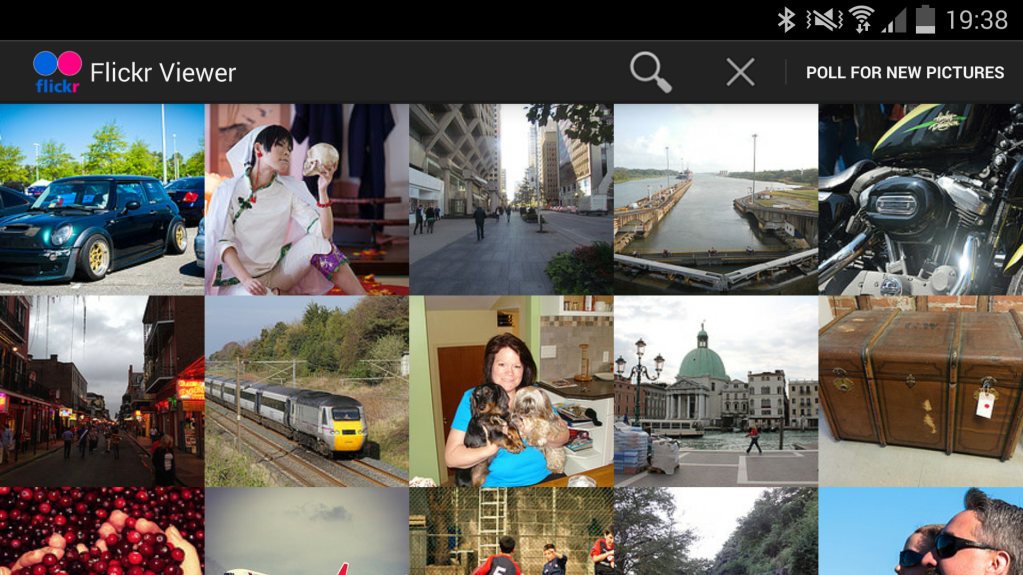
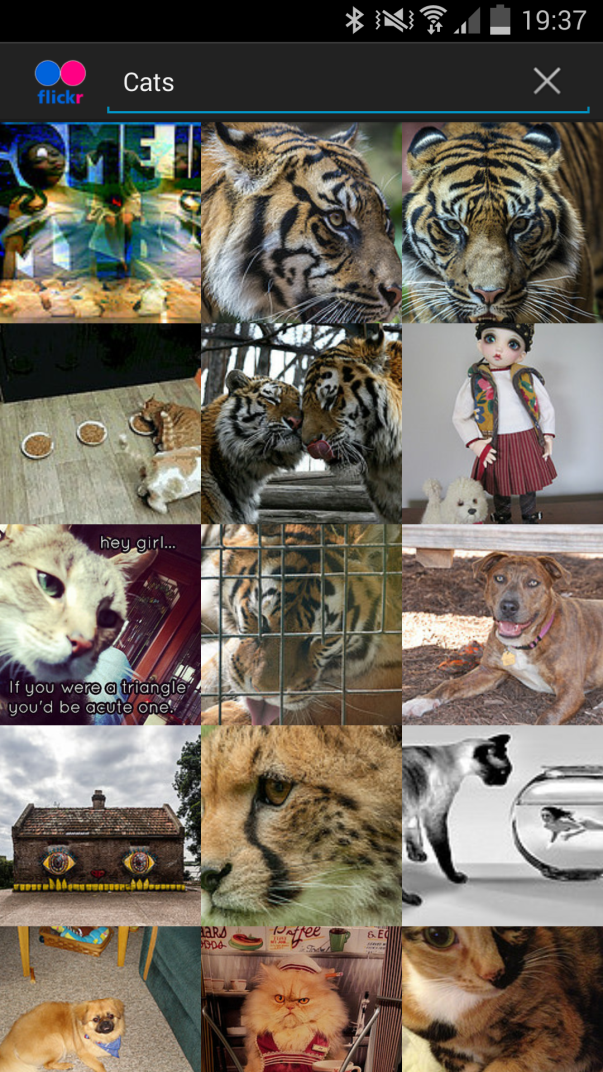


Figure 5: Screen orientation.

The user can rotate the application from portrait to landscape using the accelerometer in order to display more images per row on the screen, increasing the images per row from 3 to 5. In addition to the increase of images per row, the resolution of the images increases making them easier to see on the screen. The “poll new images” button appears much more clearly in the landscape orientation which makes the application much easier to use.

Figure 6: Search functionality.

Once the user interacts with the magnifying glass seen in “Figure 2” a searching field is displayed using the SearchView which allows the user to input a searching query into the Flickr image feed in order to specify an image to search for. Once the user has entered a search query the application searches Flickr for images tagged from their search query from the search field, the Flickr feed is refreshed displaying buffering images as the feed is being reloaded, and will eventually display the specific images which are linked to the search query which the user has input. In the case of “Figure 2” cats have been searched, and pictures of cats have been displayed to show the search functionality works as intended.

Resource Implications

The application was tested on a Samsung Galaxy S4 model number: GT-I9505 smartphone on Android version 4.4.2, and will run on most up to date Android devices.

The development was completed using the Eclipse IDE using the Android SDK windows 64bit bundle, and Android developer tools.

The application has been tested and runs perfectly on both the Samsung Galaxy S4 device in addition to the inbuilt Android emulator within the Android SDK.

# Critical Evaluation

## Aim

The aim of the project is to thoroughly design and develop a Flickr Image feed application.

## Objectives

* Simple GUI for the user to interact with.
* Loading images from a Flickr feed as thumbnails.
* Displaying additional thumbnail images when the user scrolls vertically.
* Displaying additional thumbnail images per row when the user rotates the device using accelerometers.
* Search functionality using advanced techniques such as SearchView’s.
* A safe search functionality to censor explicit content.
* Text being able to be displayed in English and French using Localisation features. The default language will be English and will change to French if the user’s device is set to French.

The overall aim of the project to “develop a Flickr Image feed application” was successfully met in good time and with relative ease as there were learning resources to follow and aid the development process.

A simple functional GUI was implemented as per the objective while offering a robust, accurate and responsive interface for the user to interact with. Concise and well-designed assets have been created to make the GUI exceptionally easy to see while remaining aesthetically pleasing overall.

The images are successfully loaded from the Flickr feed as thumbnails and implemented as per the objective, there are 15 images loaded per row, load exceptionally fast and allow the user to view a large amount of image thumbnails quickly. An additional feature of a buffering image has been added to the project to show that an image isn’t loaded fully, and allows the user to see when images are about to appear.

The orientation feature using the accelerometer has been implemented fully and functions perfectly, allowing the user to rotate the device to show more images per line and in higher resolution in the landscape orientation.

The advanced search functionality with safe search filters was the most difficult feature to implement in the application and required the most complex android features. The feature has been fully implemented, tested in the Android emulator in addition to Android devices and works fully as intended.

Localisation has been fully implemented allowing the user to view the application in English by default and in French if the device’s default language has been set to French. The feature was very easy to implement using the Android development tools although the translations may not be fully accurate as online translators are used.

# Application Evaluation

The application works as intended with all of the desired features implemented, the application is robust in both design and implementation allowing for little to no bugs or crashing errors. That being said the application isn’t perfect and there are features which could still be added if time allowed, and some minor errors fixed.

The polling for new images feature works fully as intended, allowing the user to view a wide array of images from a random Flickr feed, although there are some instances where the polling process may take longer than expected slowing the viewing process down. It is suspected that this issue could possibly due to the Flickr web servers.

The process of scrolling the screen for new images works seamlessly allowing the user to view a large amount of images in a short time frame, loading the image thumbnails as fast as the user can scroll which fully fulfils the applications niche market. Although there are some rare cases where some thumbnails due to the resolution of the images load slower than others and can further slowdown the viewing process.

The searching functionality works perfectly in the application, allowing the user to search for a wide variety of images, while displaying a large amount of relevant images. A nice addition to the search functionality was the safe search features which were implemented to censor explicit content from the application as the application is designed to be used by all age groups.

# Recommendations

If the development process of the application was continued, or if there was additional time for further implementation there are a large amount of features which could be added to the already robust application. Some of these features include:-

A splash screen which would be displayed when the application is started, showing the name of the application, and the development team which developed the application.

A further addition to the image viewing process could be implemented to allow the user to open the images individually in higher resolution images instead of only being able to view the images as thumbnails.

# Bibliography

PHILIPS, BILL and HARDY, BRIAN. 2013, *Android Programming The Big Nerd Ranch Guide,* 1989 College Ave, Atlanta GA: Big Nerd Ranch

VOGEL, LARS, 2013, JSON in Android – Tutorial, [online] [viewed: 18/4/14]. Available at: <http://www.vogella.com/tutorials/AndroidJSON/article.html>

# References

ANDROID DEVELOPERS, (2014a). **Looper** [online] [viewed: 21/4/14]. Available at: <http://developer.android.com/reference/android/os/Looper.html>

ANDROID DEVELOPERS, (2014a). **Localization** [online] [viewed: 20/4/14]. Available at: <http://developer.android.com/guide/topics/resources/localization.html>

ANDROID DEVELOPERS, (2014a). **SearchView** [online] [viewed: 21/4/14]. Available at: <http://developer.android.com/reference/android/widget/SearchView.html>

ANDROID DEVELOPERS, (2014a). **WebView** [online] [viewed: 20/4/14]. Available at: <http://developer.android.com/reference/android/webkit/WebView.html>

# Appendix A: Source Code

## FlickrFetchr.java

package com.jamescatterall.flikr\_viewer;

import java.io.ByteArrayOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.StringReader;

import java.net.HttpURLConnection;

import java.net.URL;

import java.util.ArrayList;

import org.xmlpull.v1.XmlPullParser;

import org.xmlpull.v1.XmlPullParserException;

import org.xmlpull.v1.XmlPullParserFactory;

import android.net.Uri;

import android.util.Log;

public class FlickrFetchr {

public static final String TAG = "PhotoFetcher"; //Initialising new string TAG "PhotoFetcher"

public static final String PREF\_SEARCH\_QUERY = "searchQuery"; //Initialising new string PREF\_SEARCH\_QUERY "searchQuery"

public static final String PREF\_LAST\_RESULT\_ID = "lastResultId"; //Initialising new string PREF\_LAST\_RESULT\_ID "lastResultId"

private static final String ENDPOINT = "http://api.flickr.com/services/rest/"; //Initialising new string endpoint with the Flickr API services.

private static final String API\_KEY = "69ac1bd42546a9c10aa50ab9f24c430c"; //Initialising new string API\_KEY using the Flickr developer API key.

private static final String METHOD\_GET\_RECENT = "flickr.photos.getRecent"; //Initialising new string METHOD\_GET\_RECENT which gets the recent photos.

private static final String METHOD\_SEARCH = "flickr.photos.search"; //Initialising new string METHOD\_SEARCH gets the flickr photos search.

private static final String PARAM\_EXTRAS = "extras";

private static final String PARAM\_TEXT = "text";

private static final String EXTRA\_SMALL\_URL = "url\_s";

private static final String XML\_PHOTO = "photo";

public byte[] getUrlBytes(String urlSpec) throws IOException //Converts the bytes fetched by getUrlBytes into a string.

{

URL url = new URL(urlSpec); //URL object pointed at the URL.

HttpURLConnection connection = (HttpURLConnection)url.openConnection(); //Returns the URL connection.

try

{

ByteArrayOutputStream out = new ByteArrayOutputStream();

InputStream in = connection.getInputStream();

if (connection.getResponseCode() != HttpURLConnection.HTTP\_OK)

{

return null;

}

int bytesRead = 0; //Creates new integer bytesread and sets to 0.

byte[] buffer = new byte[1024]; //Creates new buffer using byte size 1024.

while ((bytesRead = in.read(buffer)) > 0) //While look to check whether the bytes read in are greater than 0.

{

out.write(buffer, 0, bytesRead); //If so continues to write.

}

out.close();

return out.toByteArray(); //Returns out to ByteArrayOutputStream.

}

finally

{

connection.disconnect(); //Disconnects connection.

}

}

String getUrl(String urlSpec) throws IOException //Returns the URL to a string.

{

return new String(getUrlBytes(urlSpec)); //Returns new string using getUrlBytes.

}

public ArrayList<GalleryItem> fetchItems() //fetchItenms method

{

String url = Uri.parse(ENDPOINT).buildUpon() //Sets url to endpoint url.

.appendQueryParameter("method", METHOD\_GET\_RECENT)

.appendQueryParameter("api\_key", API\_KEY)

.appendQueryParameter(PARAM\_EXTRAS, EXTRA\_SMALL\_URL)

.build().toString();

return downloadGalleryItems(url); //returns downloadGalleryItems using url.

}

public ArrayList<GalleryItem> downloadGalleryItems(String url) //downloadGallery items method

{

ArrayList<GalleryItem> items = new ArrayList<GalleryItem>(); //New arraylist GalleryItem items.

try

{

String xmlString = getUrl(url);

Log.i(TAG, "Received xml: " + xmlString); //Logs received xml using xmlString.

XmlPullParserFactory factory = XmlPullParserFactory.newInstance(); //Creates new xmlPullParser factory using newInstance method.

XmlPullParser parser = factory.newPullParser(); //Creates new xmlPullParser parser using factory pullparser method.

parser.setInput(new StringReader(xmlString));

parseItems(items, parser); //Calls parseItems using items and parser.

}

catch (IOException ioe) //Robust error checking

{

Log.e(TAG, "Failed to fetch items", ioe); //Logs failed to fetch items.

}

catch (XmlPullParserException xppe) //Robust error checking

{

Log.e(TAG, "Failed to parse items", xppe); //Logs failed to parse items.

}

return items;

}

public ArrayList<GalleryItem> search(String query) //Search functionality method returning query.

{

String url = Uri.parse(ENDPOINT).buildUpon() //Creates string url using Flickr.

.appendQueryParameter("method", METHOD\_SEARCH)

.appendQueryParameter("api\_key", API\_KEY)

.appendQueryParameter(PARAM\_EXTRAS, EXTRA\_SMALL\_URL)

.appendQueryParameter(PARAM\_TEXT, query)

.build().toString();

return downloadGalleryItems(url); //returns downloadGalleryItems using url.

}

void parseItems(ArrayList<GalleryItem> items, XmlPullParser parser) throws XmlPullParserException, IOException //parseItems method.

{

int eventType = parser.next(); //Creates new integer eventType using parser next method.

while (eventType != XmlPullParser.END\_DOCUMENT) //While loop of eventType is not equal to END.DOCUMENT string.

{

if (eventType == XmlPullParser.START\_TAG && XML\_PHOTO.equals(parser.getName())) //If eventtype is equal to START\_TAG string and XML\_PHOTO string.

{

String id = parser.getAttributeValue(null, "id"); //Sets string id using parser and string id.

String caption = parser.getAttributeValue(null, "title"); //Sets caption using parser and string title.

String smallUrl = parser.getAttributeValue(null, EXTRA\_SMALL\_URL); //Sets smallUrl using parser and string EXTRA\_SMALL\_URL.

GalleryItem item = new GalleryItem(); //Creates new arrayList galleryitem item.

item.setId(id); //Sets item arraylist id to id.

item.setCaption(caption); //Sets item arraylist caption to caption.

item.setUrl(smallUrl); //Sets item arraylist url to smallUrl.

items.add(item); //Adds to item arraylist item.

}

eventType = parser.next(); //Calls parser.next on eventType.

}

}

}

## GalleryItem.java

**package** com.jamescatterall.flikr\_viewer;

**public** **class** GalleryItem

{

**private** String mCaption; //Define caption string variable.

**private** String mId; //Define id string variable.

**private** String mUrl; //Define url string variable.

**public** String getCaption() //Method to return caption variable.

{

**return** mCaption;

}

**public** **void** setCaption(String caption) //Method to set the caption variable.

{

mCaption = caption;

}

**public** String getId() //Method to return the id variable.

{

**return** mId;

}

**public** **void** setId(String id) //Method to set the id variable.

{

mId = id;

}

**public** String getUrl() //Method to return the url variable.

{

**return** mUrl;

}

**public** **void** setUrl(String url) //Method to set the url variable.

{

mUrl = url;

}

**public** String toString() //Method to return the caption toString.

{

**return** mCaption;

}

}

## NotificationReceiver.java

package com.jamescatterall.flikr\_viewer;

import android.app.Activity;

import android.app.Notification;

import android.app.NotificationManager;

import android.content.BroadcastReceiver;

import android.content.Context;

import android.content.Intent;

import android.util.Log;

public class NotificationReceiver extends BroadcastReceiver

{

private static final String TAG = "NotificationReceiver"; //Define string TAG as "NotificationReceiever".

@Override

public void onReceive(Context c, Intent i) //onReceieve activity life cycle.

{

Log.i(TAG, "received result: " + getResultCode()); //Logs the getresultcode.

if (getResultCode() != Activity.RESULT\_OK) //Checks if resultcode is not equal to activity result.

return;

int requestCode = i.getIntExtra("REQUEST\_CODE", 0);

Notification notification = (Notification)i.getParcelableExtra("NOTIFICATION"); //Creates new notification called notification using

NotificationManager notificationManager = (NotificationManager) //Creates new notificationmanager notificationmanager

c.getSystemService(Context.NOTIFICATION\_SERVICE);

notificationManager.notify(requestCode, notification);

}

}

## PhotoGalleryActivity.java

package com.jamescatterall.flikr\_viewer;

import com.jamescatterall.flikr\_viewer.R;

import android.app.SearchManager;

import android.content.Intent;

import android.preference.PreferenceManager;

import android.support.v4.app.Fragment;

import android.util.Log;

public class PhotoGalleryActivity extends SingleFragmentActivity //Public class PhotoGalleryActivity which extends SingleFragmentActivity

{

private static final String TAG = "Flickr"; //Define string TAG as "Flickr"

@Override

public Fragment createFragment() //Create fragment method.

{

return new PhotoGalleryFragment(); //Returns new PhotoGalleryFragment.

}

@Override

public void onNewIntent(Intent intent) //onNewIntent method.

{

PhotoGalleryFragment fragment = (PhotoGalleryFragment)getSupportFragmentManager()

.findFragmentById(R.id.fragmentContainer); //Creates new fragment using fragment id within resources/id/fragmentcontainer id.

if (Intent.ACTION\_SEARCH.equals(intent.getAction())) //Search functionality checks.

{

String query = intent.getStringExtra(SearchManager.QUERY); //Sets string query to searchManager.Query.

Log.i(TAG, "Received a new search query: " + query); //Logs that a new search query has been received.

PreferenceManager.getDefaultSharedPreferences(this) //Search functionality.

.edit()

.putString(FlickrFetchr.PREF\_SEARCH\_QUERY, query)

.commit();

}

fragment.updateItems(); //Updates fragment.

}

}

## PhotoGalleryFragment.java

package com.jamescatterall.flikr\_viewer;

import java.util.ArrayList;

import com.jamescatterall.flikr\_viewer.R;

import android.annotation.TargetApi;

import android.app.Activity;

import android.app.SearchManager;

import android.app.SearchableInfo;

import android.content.ComponentName;

import android.content.Context;

import android.os.AsyncTask;

import android.os.Build;

import android.os.Bundle;

import android.os.Handler;

import android.preference.PreferenceManager;

import android.view.LayoutInflater;

import android.view.Menu;

import android.view.MenuInflater;

import android.view.MenuItem;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ArrayAdapter;

import android.widget.GridView;

import android.widget.ImageView;

import android.widget.SearchView;

public class PhotoGalleryFragment extends VisibleFragment

{

GridView mGridView; //Define GridView mGridView.

ArrayList<GalleryItem> mItems; //Initialise empty arraylist GalleryItem as mItems.

ThumbnailDownloader mThumbnailThread; //ThumbnailDownloader as mThumbnailThread.

@Override

public void onCreate(Bundle savedInstanceState) //onCreate activity method.

{

super.onCreate(savedInstanceState); //Calls onCreate method using savedInstanceState.s

setRetainInstance(true); //Set retain instance to true.

setHasOptionsMenu(true); //Set hasOptionsMenu to true.

updateItems(); //Calls updateItems method when application is started.

mThumbnailThread = new ThumbnailDownloader(new Handler()); //Creates new handler thread as a new ThumbnailDownloader.

mThumbnailThread.start(); //Start mThumbnailThread handler thread.

}

public void updateItems() //Update FetchItems.

{

new FetchItemsTask().execute(); //Calls execute on fetchItemTask.

}

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) //onCreateView method.

{

View v = inflater.inflate(R.layout.fragment\_photo\_gallery, container, false); //Creates new view v using inflater from resources/layout/fragment\_photo\_gallery.

mGridView = (GridView)v.findViewById(R.id.gridView); //Sets gridView to gridview using resources/id/gridview.

setupAdapter(); //Calls setupAdapter.

return v; //Returns v.

}

@Override

public void onDestroy() //onDestroy life cycle.

{

super.onDestroy(); //Calls onDestroy on super.

mThumbnailThread.quit(); //Calls quit on ThumbnailThread on close.

}

@Override

public void onDestroyView() //onDestroyView life cycle.

{

super.onDestroyView(); //Calls onDestroyView on super.

mThumbnailThread.clearQueue(); //Clears clearQueue on mThumbnailThread to empty queue.

}

@Override

@TargetApi(11)

public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) //onCreateOptionsMenu method.

{

super.onCreateOptionsMenu(menu, inflater); //Calls onCreateOptionsMenu using menu and inflater.

inflater.inflate(R.menu.fragment\_photo\_gallery, menu); //Calls inflater.inflate using resources/menu/fragment\_photo\_gallery.

if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.HONEYCOMB) { //Checks if buildversion of honeycomb.

// pull out the SearchView

MenuItem searchItem = menu.findItem(R.id.menu\_item\_search); //Creates menuItem searchItem using resources/id/menu\_item\_search.

SearchView searchView = (SearchView)searchItem.getActionView(); //Creates SearchView searchView using searchItem.getActionView.

// get the data from our searchable.xml as a SearchableInfo

SearchManager searchManager = (SearchManager)getActivity()

.getSystemService(Context.SEARCH\_SERVICE); //Creates new SearchManager searchManager using SEARCH\_SERVICE string.

ComponentName name = getActivity().getComponentName();

SearchableInfo searchInfo = searchManager.getSearchableInfo(name);

searchView.setSearchableInfo(searchInfo);

}

}

@Override

@TargetApi(11)

public boolean onOptionsItemSelected(MenuItem item) //onOptionsItemSelected method.

{

switch (item.getItemId()) //Switch using itemId.

{

case R.id.menu\_item\_search: //Switch case using resources/id/menu\_item\_search

getActivity().onSearchRequested(); //calls get

return true;

case R.id.menu\_item\_clear: //Switch case using resources/id/menu\_item\_clear

PreferenceManager.getDefaultSharedPreferences(getActivity())

.edit()

.putString(FlickrFetchr.PREF\_SEARCH\_QUERY, null)

.commit();

updateItems();

return true; //Returns boolean to true.

case R.id.menu\_item\_toggle\_polling: //Switch case using resource/id/menu\_item\_toggle\_polling.

boolean shouldStartAlarm = !PollService.isServiceAlarmOn(getActivity()); //Sets shouldStartAlarm based on whether the servicealarm is on.

PollService.setServiceAlarm(getActivity(), shouldStartAlarm);

if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.HONEYCOMB) //Checks if the SDK\_INT is greater than or equal to the build code of the honeycomb.

getActivity().invalidateOptionsMenu();

return true;

default:

return super.onOptionsItemSelected(item);

}

}

@Override

public void onPrepareOptionsMenu(Menu menu) //onPrepareOptionsMenu

{

super.onPrepareOptionsMenu(menu);

MenuItem toggleItem = menu.findItem(R.id.menu\_item\_toggle\_polling); //Creates new menuItem toggleitem using resources/id/menu\_item\_toggle\_polling.

if (PollService.isServiceAlarmOn(getActivity())) //Checks if service alarm is on.

{

toggleItem.setTitle(R.string.stop\_polling); //Sets toggleItem to resource/string/stop\_polling.

}

else

{

toggleItem.setTitle(R.string.start\_polling); //Sets toggleItem to resource/string/start\_polling.

}

}

void setupAdapter() { //SetupAdapter method

if (getActivity() == null || mGridView == null) return; //Checks if the activity is equal to null or the gridview is equal to null.

if (mItems != null) { //If the mItems arraylist is null.

mGridView.setAdapter(new GalleryItemAdapter(mItems)); //Sets the adapter of the gridview to the mItems arraylist.

}

else

{

mGridView.setAdapter(null); //Sets the mGridview adapter to null.

}

}

private class FetchItemsTask extends AsyncTask<Void,Void,ArrayList<GalleryItem>> //FetchItemsTask method.

{

@Override

protected ArrayList<GalleryItem> doInBackground(Void... params)

{

Activity activity = getActivity(); //Creates new activity using getActivity method.

if (activity == null) //checks if activity is equal to null.

return new ArrayList<GalleryItem>(); //Returns arrayList.

String query = PreferenceManager.getDefaultSharedPreferences(activity)

.getString(FlickrFetchr.PREF\_SEARCH\_QUERY, null);

if (query != null) //Checks if query is not equal to null.

{

return new FlickrFetchr().search(query); //Returns search query.

}

else

{

return new FlickrFetchr().fetchItems(); //Returns fetched items.

}

}

@Override

protected void onPostExecute(ArrayList<GalleryItem> items) //onPostExecute method.

{

mItems = items; //Sets mItems to items.

if (items.size() > 0) //Checks if items arraylist size is greater than 0.

{

String resultId = items.get(0).getId(); //Sets string resultId to arraylist size getid.

PreferenceManager.getDefaultSharedPreferences(getActivity())

.edit()

.putString(FlickrFetchr.PREF\_LAST\_RESULT\_ID, resultId)

.commit();

}

setupAdapter(); //Calls setupAdapter method.

}

}

private class GalleryItemAdapter extends ArrayAdapter<GalleryItem> //GalleryItemAdapter method.

{

public GalleryItemAdapter(ArrayList<GalleryItem> items) //GalleryItemAdapter using GalleryItem arraylist Item.

{

super(getActivity(), 0, items);

}

@Override

public View getView(int position, View convertView, ViewGroup parent) //getView method.

{

if (convertView == null) //Checks if convertView is equal to null.

{

convertView = getActivity().getLayoutInflater()

.inflate(R.layout.gallery\_item, parent, false); //Sets convertView to resources/layout/gallery\_item.

}

GalleryItem item = getItem(position); //Creaes new GalleryItem item using position.

ImageView imageView = (ImageView)convertView

.findViewById(R.id.gallery\_item\_imageView); //Creates new imageview using resources/id/gallery\_item\_imageView.

imageView.setImageResource(R.drawable.buffering); //Sets imageView to resource/drawable/buffering.

mThumbnailThread.queueThumbnail(imageView, item.getUrl());

return convertView; //Returns convertView.

}

}

}

## PollService.java

package com.jamescatterall.flikr\_viewer;

import java.util.ArrayList;

import com.jamescatterall.flikr\_viewer.R;

import android.app.Activity;

import android.app.AlarmManager;

import android.app.IntentService;

import android.app.Notification;

import android.app.NotificationManager;

import android.app.PendingIntent;

import android.content.BroadcastReceiver;

import android.content.Context;

import android.content.Intent;

import android.content.SharedPreferences;

import android.content.res.Resources;

import android.net.ConnectivityManager;

import android.preference.PreferenceManager;

import android.support.v4.app.NotificationCompat;

import android.util.Log;

public class PollService extends IntentService //Handles service intents (commands).

{

private static final String TAG = "PollService"; //Define string TAG as "PollService".

private static final int POLL\_INTERVAL = 1000 \* 60 \* 5; // 5 minutes

public static final String PREF\_IS\_ALARM\_ON = "isAlarmOn"; //Define service alarm.

public static final String ACTION\_SHOW\_NOTIFICATION = "com.jamescatterall.flikr\_viewer.SHOW\_NOTIFICATION"; //Define show notification using package.

public static final String PERM\_PRIVATE = "com.jamescatterall.flikr\_viewer.PRIVATE";

public PollService()

{

super(TAG); //Sets pollservice to super using TAG string "PollService".

}

@Override

public void onHandleIntent(Intent intent) //onHandleIntent method

{

ConnectivityManager cm = (ConnectivityManager)

getSystemService(Context.CONNECTIVITY\_SERVICE);

@SuppressWarnings("deprecation")

boolean isNetworkAvailable = cm.getBackgroundDataSetting() &&

cm.getActiveNetworkInfo() != null;

Log.i(TAG, "In service! network available: " + isNetworkAvailable);

if (!isNetworkAvailable) return;

SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(this);

String query = prefs.getString(FlickrFetchr.PREF\_SEARCH\_QUERY, null);

String lastResultId = prefs.getString(FlickrFetchr.PREF\_LAST\_RESULT\_ID, null);

ArrayList<GalleryItem> items; //Initialise new arraylist gallery items called items.

if (query != null) //Checks if query is not equal to null.

{

items = new FlickrFetchr().search(query); //Defines items using search query.

}

else

{

items = new FlickrFetchr().fetchItems(); //Defines items as item after fetched.

}

if (items.size() == 0) //Checks if items arraylist is equal to 0.

return;

String resultId = items.get(0).getId(); //Creates new string resultsId using items array id.

if (!resultId.equals(lastResultId)) //Checks if resultsId is equal to the lastresultid.

{

Log.i(TAG, "Got a new result: " + resultId); //Log the result id.

Resources r = getResources();

PendingIntent pi = PendingIntent

.getActivity(this, 0, new Intent(this, PhotoGalleryActivity.class), 0); //PendingIntent using activity from photogalleryacivity class.

Notification notification = new NotificationCompat.Builder(this) //Notification buildergetting image title, image and text.

.setTicker(r.getString(R.string.new\_pictures\_title))

.setSmallIcon(android.R.drawable.ic\_menu\_report\_image)

.setContentTitle(r.getString(R.string.new\_pictures\_title))

.setContentText(r.getString(R.string.new\_pictures\_text))

.setContentIntent(pi)

.setAutoCancel(true)

.build();

showBackgroundNotification(0, notification);

}

prefs.edit()

.putString(FlickrFetchr.PREF\_LAST\_RESULT\_ID, resultId) //Edits in last result id.

.commit();

}

public static void setServiceAlarm(Context context, boolean isOn) //setServiceAlarm method.

{

Intent i = new Intent(context, PollService.class);

PendingIntent pi = PendingIntent.getService(

context, 0, i, 0);

AlarmManager alarmManager = (AlarmManager)

context.getSystemService(Context.ALARM\_SERVICE); //Creates new alarmManager alarmmanager using system service.

if (isOn) //Checks if the service is off.

{

alarmManager.setRepeating(AlarmManager.RTC,

System.currentTimeMillis(), POLL\_INTERVAL, pi);

}

else //Checks if alarm is off.

{

alarmManager.cancel(pi); //Cancels alarm.

pi.cancel();

}

PreferenceManager.getDefaultSharedPreferences(context)

.edit()

.putBoolean(PollService.PREF\_IS\_ALARM\_ON, isOn)

.commit();

}

public static boolean isServiceAlarmOn(Context context) //isServiceAlarmOn method.

{

Intent i = new Intent(context, PollService.class); //Creates new intent i using pollservice.

PendingIntent pi = PendingIntent.getService(

context, 0, i, PendingIntent.FLAG\_NO\_CREATE); //Creates pendingintent pi using service state.

return pi != null; //Returns if pi is not equal to null.

}

void showBackgroundNotification(int requestCode, Notification notification) //showBackgroundNotification method.

{

Intent i = new Intent(ACTION\_SHOW\_NOTIFICATION); //Creates new intent using show\_notification.

i.putExtra("REQUEST\_CODE", requestCode);

i.putExtra("NOTIFICATION", notification);

sendOrderedBroadcast(i, PERM\_PRIVATE, null, null, Activity.RESULT\_OK, null, null);

}

}

## SingleFragmentActivity.java

package com.jamescatterall.flikr\_viewer;

import com.jamescatterall.flikr\_viewer.R;

import android.os.Bundle;

import android.support.v4.app.Fragment;

import android.support.v4.app.FragmentActivity;

import android.support.v4.app.FragmentManager;

public abstract class SingleFragmentActivity extends FragmentActivity

{

protected abstract Fragment createFragment();

@Override

public void onCreate(Bundle savedInstanceState) //onCreate activity.

{

super.onCreate(savedInstanceState); //Calls onCreate using savedInstanceState.

setContentView(R.layout.activity\_fragment);

FragmentManager manager = getSupportFragmentManager(); //Creates new fragmentmanager manager.

Fragment fragment = manager.findFragmentById(R.id.fragmentContainer); //Creates new fragment called fragment using the fragment manager.

if (fragment == null) { //Checks if fragment is equal to null

fragment = createFragment(); //Creates fragment

manager.beginTransaction()

.add(R.id.fragmentContainer, fragment) //Adds fragment to the fragmentContainer.

.commit();

}

}

}

## StartupReceiver.java

package com.jamescatterall.flikr\_viewer;

import android.content.BroadcastReceiver;

import android.content.Context;

import android.content.Intent;

import android.content.SharedPreferences;

import android.preference.PreferenceManager;

import android.util.Log;

public class StartupReceiver extends BroadcastReceiver

{

private static final String TAG = "StartupReceiver"; //Define string TAG for StartupReceiver.

@Override

public void onReceive(Context context, Intent intent)

{

Log.i(TAG, "Received broadcast intent: " + intent.getAction()); //Logs whether the broadcasted intent has been receieved.

SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(context);

boolean isOn = prefs.getBoolean(PollService.PREF\_IS\_ALARM\_ON, false); //Turns the poll service alarm on.

PollService.setServiceAlarm(context, isOn); //Sets the serviceAlarm to on.

}

}

## ThumbnailDownloader.java

package com.jamescatterall.flikr\_viewer;

import java.io.IOException;

import java.util.Collections;

import java.util.HashMap;

import java.util.Map;

import android.annotation.SuppressLint;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.os.Handler;

import android.os.HandlerThread;

import android.os.Message;

import android.util.Log;

import android.widget.ImageView;

public class ThumbnailDownloader extends HandlerThread

{

private static final String TAG = "ThumbnailDownloader"; //Initialise string TAG as "ThumbnailDownloader".

private static final int MESSAGE\_DOWNLOAD = 0; //Initialise integer MESSAGE\_DOWNLOAD as 0.

Handler mHandler; //Create handler mHandler

Map<ImageView,String> requestMap =

Collections.synchronizedMap(new HashMap<ImageView,String>());//Creates new ImageView map and requests map using synchronizedMap.

Handler mResponseHandler;

public ThumbnailDownloader(Handler responseHandler) //Handles the thumbnail downloading

{

super(TAG);

mResponseHandler = responseHandler; //Sets mResponseHandler to responseHandler.

}

@SuppressLint("HandlerLeak")

@Override

protected void onLooperPrepared() //Looper method

{

mHandler = new Handler() //New handler mHandler.

{

@Override

public void handleMessage(Message msg) //Message handler

{

if (msg.what == MESSAGE\_DOWNLOAD) //Checks if msg.what is equal to MESSAGE\_DOWNLOAD.

{

ImageView imageView = (ImageView)msg.obj; //Sets imageview to message.

Log.i(TAG, "Got a request for url: " + requestMap.get(imageView)); //Logs the request for URL.

handleRequest(imageView); //Handles request equal to imageView.

}

}

};

}

private void handleRequest(final ImageView imageView) //Handlerequest for imageview downloading images.

{

try

{

final String url = requestMap.get(imageView); //Initialise string URL using requestmap.

if (url == null) //Returns if url is null.

return;

byte[] bitmapBytes = new FlickrFetchr().getUrlBytes(url);

final Bitmap bitmap = BitmapFactory

.decodeByteArray(bitmapBytes, 0, bitmapBytes.length); //Constructs bitmap with the array of bytes returned.

mResponseHandler.post(new Runnable()

{

public void run()

{

if (requestMap.get(imageView) != url) //Checks if imageView is not equal to URL.

return;

requestMap.remove(imageView); //Removes imageView from requestMap.

imageView.setImageBitmap(bitmap); //Sets the image from bitmap constructed above.

}

});

}

catch (IOException ioe) //Robust error checking

{

Log.e(TAG, "Error downloading image", ioe); //Returns image downloading error if unable to download.

}

}

public void queueThumbnail(ImageView imageView, String url) //Thumbnail queue using imageview.

{

requestMap.put(imageView, url); //Requests the map using imageview and url.

mHandler

.obtainMessage(MESSAGE\_DOWNLOAD, imageView) //Obtains message from handler.

.sendToTarget(); //Sends message to the handler.

}

public void clearQueue() //Clears handler queue.

{

mHandler.removeMessages(MESSAGE\_DOWNLOAD); //Removes messages from handler.

requestMap.clear(); //calls clear on requestmap.

}

}

## VisibleFragment.java

package com.jamescatterall.flikr\_viewer;

import android.app.Activity;

import android.content.BroadcastReceiver;

import android.content.Context;

import android.content.Intent;

import android.content.IntentFilter;

import android.support.v4.app.Fragment;

import android.util.Log;

public abstract class VisibleFragment extends Fragment

{

public static final String TAG = "VisibleFragment"; //Define string TAG as "VisibleFragment".

private BroadcastReceiver mOnShowNotification = new BroadcastReceiver()

{

@Override

public void onReceive(Context context, Intent intent) //Called when application is receieved

{WW

Log.i(TAG, "canceling notification"); //Cancel notification

setResultCode(Activity.RESULT\_CANCELED); //Set activity result to canceled.

}

};

@Override

public void onResume() //Called when application is resumed.

{

super.onResume(); //onResume is called.

IntentFilter filter = new IntentFilter(PollService.ACTION\_SHOW\_NOTIFICATION); //Defines intents filter using PollService.

getActivity().registerReceiver(mOnShowNotification, filter, PollService.PERM\_PRIVATE, null);

}

@Override

public void onPause() //Called when application is paused.

{

super.onPause(); //onPaused is called.

getActivity().unregisterReceiver(mOnShowNotification); //Sets unregisterdReceiver to mOnShowNotification.

}

}

## Activity\_fragment.xml

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<FrameLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:id=*"@+id/fragmentContainer"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

/>

## Fragment\_photo\_gallery.xml

<GridView xmlns:android=*"http://schemas.android.com/apk/res/android"*

xmlns:tools=*"http://schemas.android.com/tools"*

android:id=*"@+id/gridView"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:columnWidth=*"120dp"*

android:numColumns=*"auto\_fit"*

android:stretchMode=*"columnWidth"* >

</GridView>

## Gallery\_item.xml

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<ImageView xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:id=*"@+id/gallery\_item\_imageView"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"120dp"*

android:layout\_gravity=*"center"*

android:scaleType=*"centerCrop"* >

</ImageView>

## Fragment\_photo\_gallery.xml

<menu xmlns:android=*"http://schemas.android.com/apk/res/android"*>

<item android:id=*"@+id/menu\_item\_search"*

android:title=*"@string/search"*

android:icon=*"@android:drawable/ic\_menu\_search"*

android:showAsAction=*"ifRoom"*

android:actionViewClass=*"android.widget.SearchView"*

/>

<item android:id=*"@+id/menu\_item\_clear"*

android:title=*"@string/clear\_search"*

android:icon=*"@android:drawable/ic\_menu\_close\_clear\_cancel"*

android:showAsAction=*"ifRoom"*

/>

<item android:id=*"@+id/menu\_item\_toggle\_polling"*

android:title=*"@string/start\_polling"*

android:showAsAction=*"ifRoom"*

/>

</menu>

## Strings.xml

<resources>

<string name=*"app\_name"*>Flickr Viewer</string>

<string name=*"menu\_settings"*>Settings</string>

<string name=*"title\_activity\_photo\_gallery"*>Flickr Viewer</string>

<string name=*"search\_hint"*>Search Flickr</string>

<string name=*"search"*>Search</string>

<string name=*"clear\_search"*>Clear Search</string>

<string name=*"start\_polling"*>Poll for new pictures</string>

<string name=*"stop\_polling"*>Stop polling</string>

<string name=*"new\_pictures\_title"*>New Flickr Pictures</string>

<string name=*"new\_pictures\_text"*>You have new pictures in Flickr.</string>

</resources>

## Styles.xml

<resources>

<style name=*"AppTheme"* parent=*"android:Theme.Light"* />

</resources>

## Strings.xml (fr)

<resources>

<string name=*"app\_name"*>Flickr spectateur</string>

<string name=*"clear\_search"*>Effacer la recherche</string>

<string name=*"menu\_settings"*>Paramètres</string>

<string name=*"new\_pictures\_text"*>Vous avez de nouvelles photos dans Flickr</string>

<string name=*"new\_pictures\_title"*>Nouveau Flickr Photos</string>

<string name=*"search"*>Recherche</string>

<string name=*"search\_hint"*>Rechercher Flickr</string>

<string name=*"start\_polling"*>Sondage pour de nouvelles photos</string>

<string name=*"stop\_polling"*>Arrêtez scrutin</string>

<string name=*"title\_activity\_photo\_gallery"*>Flickr spectateur</string>

</resources>

## Searchable.xml

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<searchable xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:label=*"@string/app\_name"*

android:hint=*"@string/search\_hint"*

/>

## AndroidManifest.xml

<manifest xmlns:android=*"http://schemas.android.com/apk/res/android"*

package=*"com.jamescatterall.flikr\_viewer"*

android:versionCode=*"1"*

android:versionName=*"1.0"* >

<uses-sdk

android:minSdkVersion=*"8"*

android:targetSdkVersion=*"17"* />

<permission android:name=*"com.jamescatterall.flikr\_viewer.PRIVATE"*

android:protectionLevel=*"signature"* />

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

<uses-permission android:name=*"android.permission.ACCESS\_NETWORK\_STATE"* />

<uses-permission android:name=*"android.permission.RECEIVE\_BOOT\_COMPLETED"* />

<uses-permission android:name=*"com.bignerdranch.android.photogallery.PRIVATE"* />

<application

android:icon=*"@drawable/ic\_launcher"*

android:label=*"@string/app\_name"*

android:theme=*"@style/AppTheme"* >

<activity

android:name=*"com.jamescatterall.flikr\_viewer.PhotoGalleryActivity"*

android:launchMode=*"singleTop"*

android:label=*"@string/title\_activity\_photo\_gallery"* >

<intent-filter>

<action android:name=*"android.intent.action.MAIN"* />

<category android:name=*"android.intent.category.LAUNCHER"* />

</intent-filter>

<intent-filter>

<action android:name=*"android.intent.action.SEARCH"* />

</intent-filter>

<meta-data android:name=*"android.app.searchable"*

android:resource=*"@xml/searchable"*/>

</activity>

<service android:name=*"com.jamescatterall.flikr\_viewer.PollService"* />

<receiver android:name=*"com.jamescatterall.flikr\_viewer.StartupReceiver"*>

<intent-filter>

<action android:name=*"android.intent.action.BOOT\_COMPLETED"* />

</intent-filter>

</receiver>

<receiver android:name=*"com.jamescatterall.flikr\_viewer.NotificationReceiver"*

android:exported=*"false"*>

<intent-filter

android:priority=*"-999"*>

<action android:name=*"com.bignerdranch.android.photogallery.SHOW\_NOTIFICATION"* />

</intent-filter>

</receiver>

</application>

</manifest>