Stephanie Reveley
073718815
slr7@aber.ac.uk

One-Post - Aggregating social updates to common social media websites, using open source methodologies on an android platform

Richard Shipman
rcs@aber.ac.uk

G401
Computer Science

Version 1
21st November
Draft

Word Count - 2815

Contents

Project Summary

The project was inspired by a post on Google+. A friend commented that wouldn’t it be awesome if an application took your content, ran it against some rules and sent your content to various sites based on these rules.

Some of the rules required photographic recognition, other rules were based on content length. It is the generic rules such as content length rules and where all photos should go that I have chosen to implement in this project, as I do not have any knowledge about image processing.

One-Post is a tool for aggregating social updates, using basic rules like content shorter than 140 characters goes to twitter. The over all aim is to minimise the amount of time it takes an end-user to update their status and upload their photos. Instead of logging in an uploading photos several times on different sites, One-Post stores all your login details then when you want to upload a set of images, it takes them, logs in and uploads them to all the sites you ask it to.

One-Post will, for the purpose of this project, only integrate with Facebook, Twitter, Google+, LinkedIn, TwitPic, Flickr and AudioBoo. These sites represent the main social media sites in everyday use. The application is being specifically designed to allow other sites to be added in the future, thus increasing the popularity of One-Post.

One-Post is being written as an open source android android application as this is an ever growing market. The Android Market rules and project time lines are less stringent than those for the Iphone.

One-Post also has uses for professionals from many industries, recruiters are increasingly looking for prospective employees information on the web. Maintaining your social presence is being heavily emphasised by careers advisory and One-Post can make that task a little easier.

Background

One-Post is using ideas taken from the Open Source Development model, Android itself is Open Source and this was a key factor in choosing Android as the platform base. Both FogelK and RaymondE have helped me to understand what makes a project open source, the material covered in their books also gives valuable pointers for organising and open source project.

As an open source project licensing has been a large concern. StalmanR and LaurenM have helped me to understand licensing issues, as well as the objectives and limitations of specific licenses. When looking into similar applications that may be of use to me I have predominantly been looking at open source projects. The license that these projects are released under has a huge impact on my project. If I were to reuse any GPL licensed code in my project I would have to release my entire project under the GPL, although the GPLs main goal is to secure a projects freedom, I feel that this condition of the GPL license restricts my own freedom to choose my license. As such I will be avoiding all code released under a GPL license.

Ubuntu[11] Me-Menu, as a whole is of little consequence however the broadcast accounts feature is useful. This feature links with Twitter, Facebook and Identi.ca and text based status updates to all 3 accounts if the user has provided his/ her credentials. Although this is useful its use is limited, the Me-Menu has been removed from the last release of Ubuntu and the source code is not available. The Me-Menu is useful in showing on a small scale how One-Post should work, for text based updates.

AudioBoo[1][15] is one of the social sites I will be integrating with in One-Post. AudioBoo has announced that it has make its Android application source code, free to use under the MIT license. Having access to the source code under such a free license allows me to reuse code and help another project by promoting theirs. If I use the source code released by AudioBoo, the sections used will be clearly marked. If I find that I cant use their source in the ways I need it, then the source code will be useful as working examples.

Gwibber[14], is an open source micro-blogging client for linux desktops. Gwibber currently uploads text based updates to Twitter, Facebook, Identi.ca, StatusNet, FriendSteam, Digg and Qaiku. Gwibber also uploads photos to Flikr. The source is in Python and GTK making this no good for me to reuse code, however it demonstrates working examples of areas I have flagged as potentially complicated work, such as integrating and uploading with Flickr. Gwibber not only shows me working examples of uploading content but demonstrates linking to different APIs.

Goals and Objectives

Android

Having decided to develop for the android platform a little research was required to work out what specifically should I be developing for.
According the the android developers website[16], most android devices are currently running either Froyo(Version 2.2) or Gingerbread(Version 2.3) releases of Android. As such I will be compiling my source code to these sources. As Ice Cream Sandwich(version 4) has just bean released I will consider compiling for this as well, subject to the number of devices that upgrade over the next couple of months.

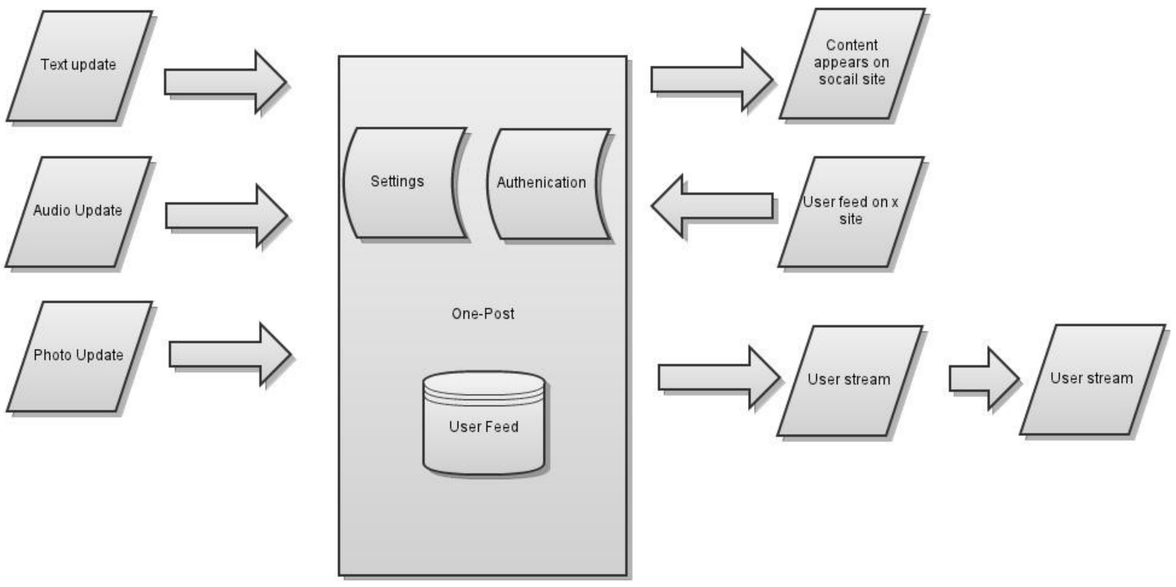
In addition to the software requirements, I have had to consider the sizes of device that One-Post will run on. Android categorises the size of the device in 4 ways; small, normal, large, and xlarge. The density of screen is also categorised in 4 ways; ldpi (low), mdpi (medium), hdpi (high), and xhdpi (extra high), these categories generalise the dots/ pixels per inch. According the the Android Developers site, the most common combination of screen size/ density is normal/ hdpi. In actual sizes this represents devises that are approx 3-5inches diagonally and 190-280 dots per inch.

Available to me for testing are the HTC Sensation, HTC Desire, Sony Ericsson Xperia X10 and the Asus eee pad transformer. Excluding the later device each of my test devices fall under the most common Android devices by way of Android version, size and density. As such I will be able to test my application on a variety of devices that fit my main market.

Inputs/ Outputs

One-Post take content in the form of text, audio and photographic as well as data from the social sites. Text, audio and photo updates are uploaded depending on the users preferences. The data taken from the users social sites, is used to create a central feed of the users social network.

Input Output Diagram



Success?

The success of the project in its simplest form can be measured by having a working version of One-Post available for the general public to download and use from the Android Market. Such a success is the ultimate goal of this project, however success can and will be measured in other aspects such as gaining feedback from the Open Source community.

Standard milestones such as completion of design, implementation and testing also apply, although at times the boundaries between these might not be as definitive.

Design will be considered a work in progress for the duration of this project, however a stable version of the design should be available early on in the project.

Implementation and testing will be completed in stages.
Stage one – application uploads text updates and passes tests on multiple test devices.
Stage two – as stage one, but also uploads photo updates.
Stage three – as stage one and two but also creates a stream of users social feed.
Stage four – as all previous stages, but also uploads audio updates.
Stage five – as all previous stages, but allows users to comment on friends updates accessed via social feed.

Each stage will be subject to unit testing, and small scale user testing, which must be passed before being release to the Open Source community.

Current Progress

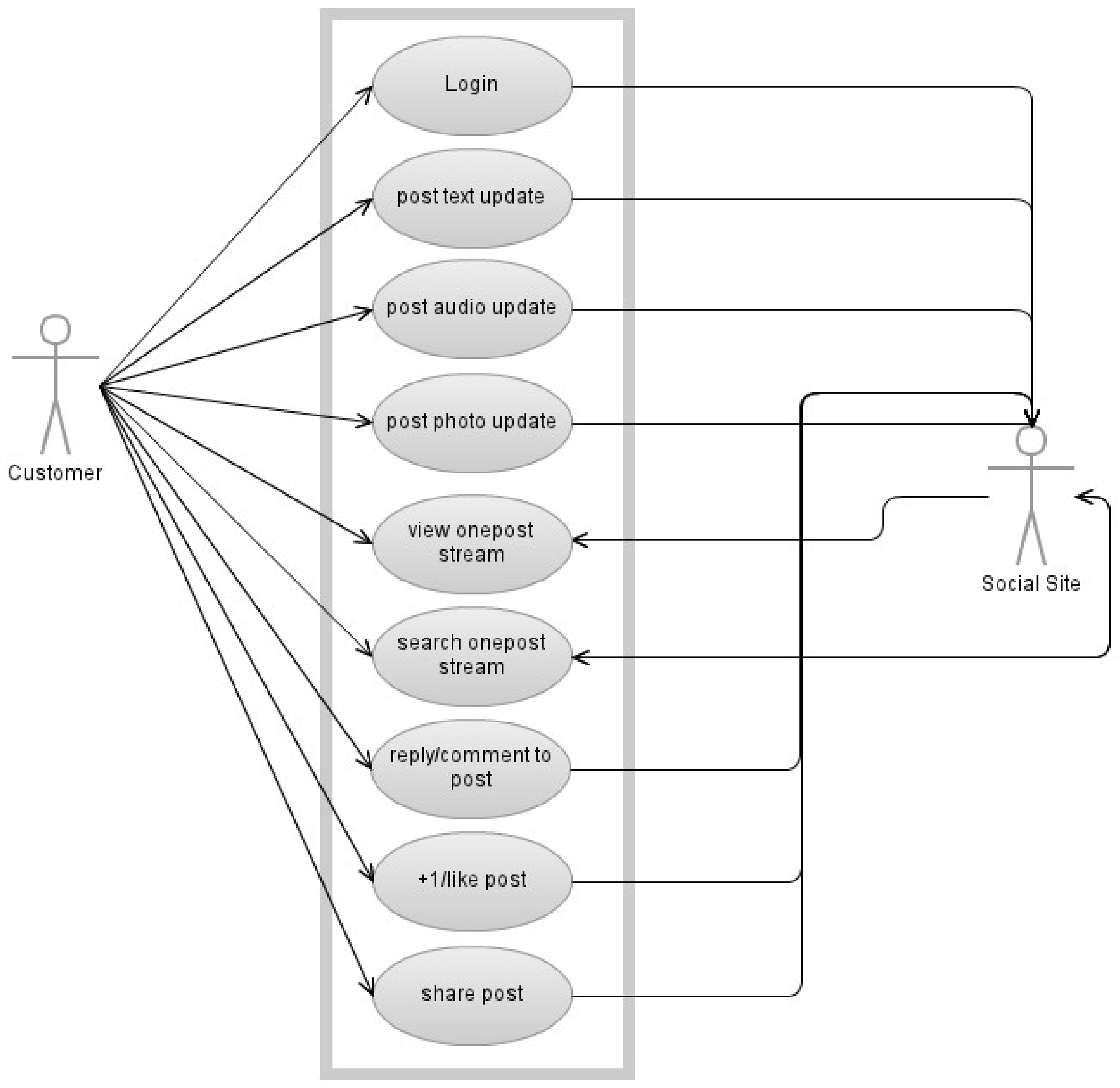
Technical Challenges

There are several challenges that I have already highlighted in the project, in order of risk, these are:

- Integrating with multiple different APIs
 - Understanding API jargon, APIs like Facebook and Flikr are complicated with information my project requires hidden in complex systems. I plan to over come this by implementing the simpler APIs like Twitter and TwitPic to further my understanding of APIs. Going over source code that alread integrates with these sites will also help to deepend my understanding Gwibber will be very useful in this respect.
 - Each API handles data in a slightly different way. Although this is true there are similarities in the type of requests and the ways in which these requests are made. These similarities make hadeling the requests simpler, however by integrating with so many APIs I expect to get confused with site specific symantics.
 - Authentication. Most of the sites use Oauth, authenticating to one site should mean that authenticating to the others will be simple. However having looked around at Oauth there are several cases where Oauth doesn't behave as expected. There isnt a great deal I can do with this, I know there is a risk here but there is no definative issue, meaning that if I come across issues here I will have to debugg them as I go.
- Programming for android
 - Sending http requests over Java, this is possible and tutorials are available online.
 - Connecting to and using options available on the Android device such as WiFi/ GPRS, taking a photo, recording audio. Both the Android developer website and the text Developing for android have in depth examples and tutorials demonstrating how this kind of idea can be implemented.
 - It has been along time since I programmed in Java. There isnt a great deal I can do to change this. I have also been looking at small simple projects predominantly for android that I can use as a quick refresher. To minimise a very steep re-learning curve, I will be focusing on the less complicated aspects of the project and implementing these first, so as to get my eye back into Java before tackling the harder aspect.
- Creating a framework for the project that is flexible, allowing the application to adapt and include more current and future social networks.

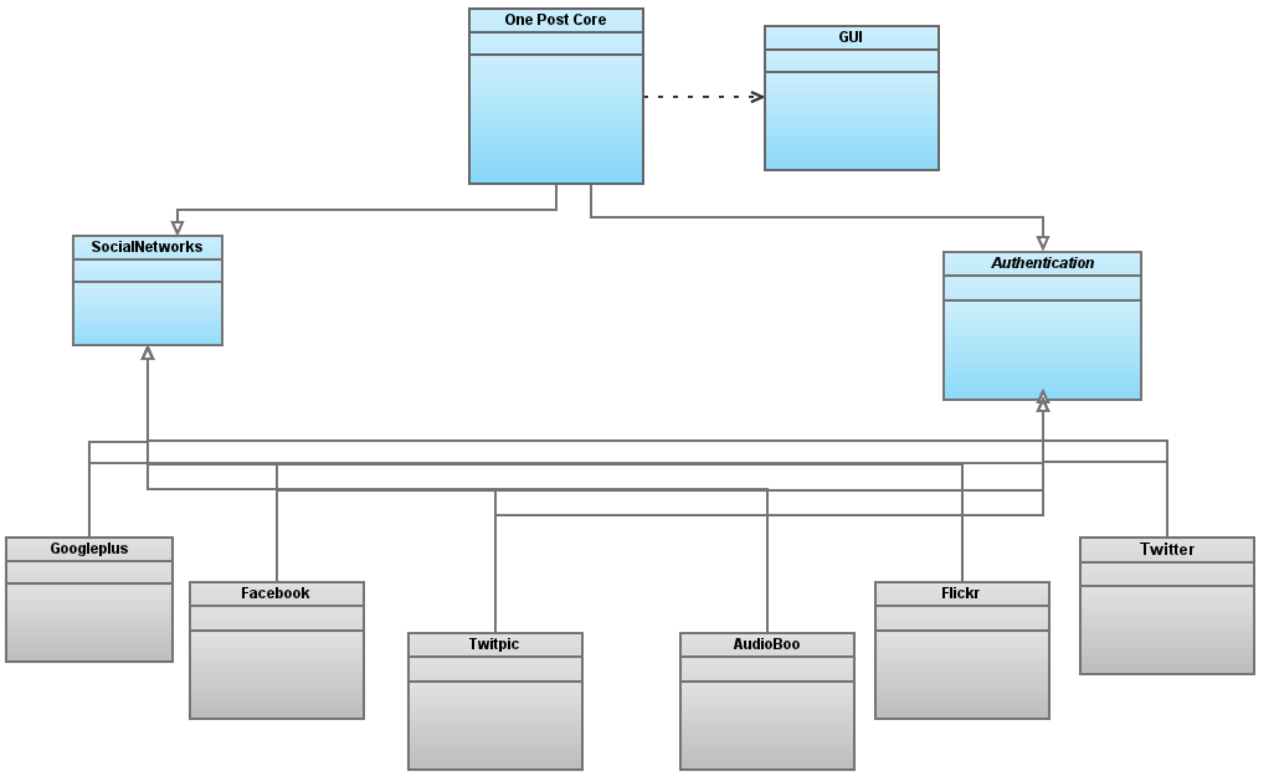
Outline Design

Use Case



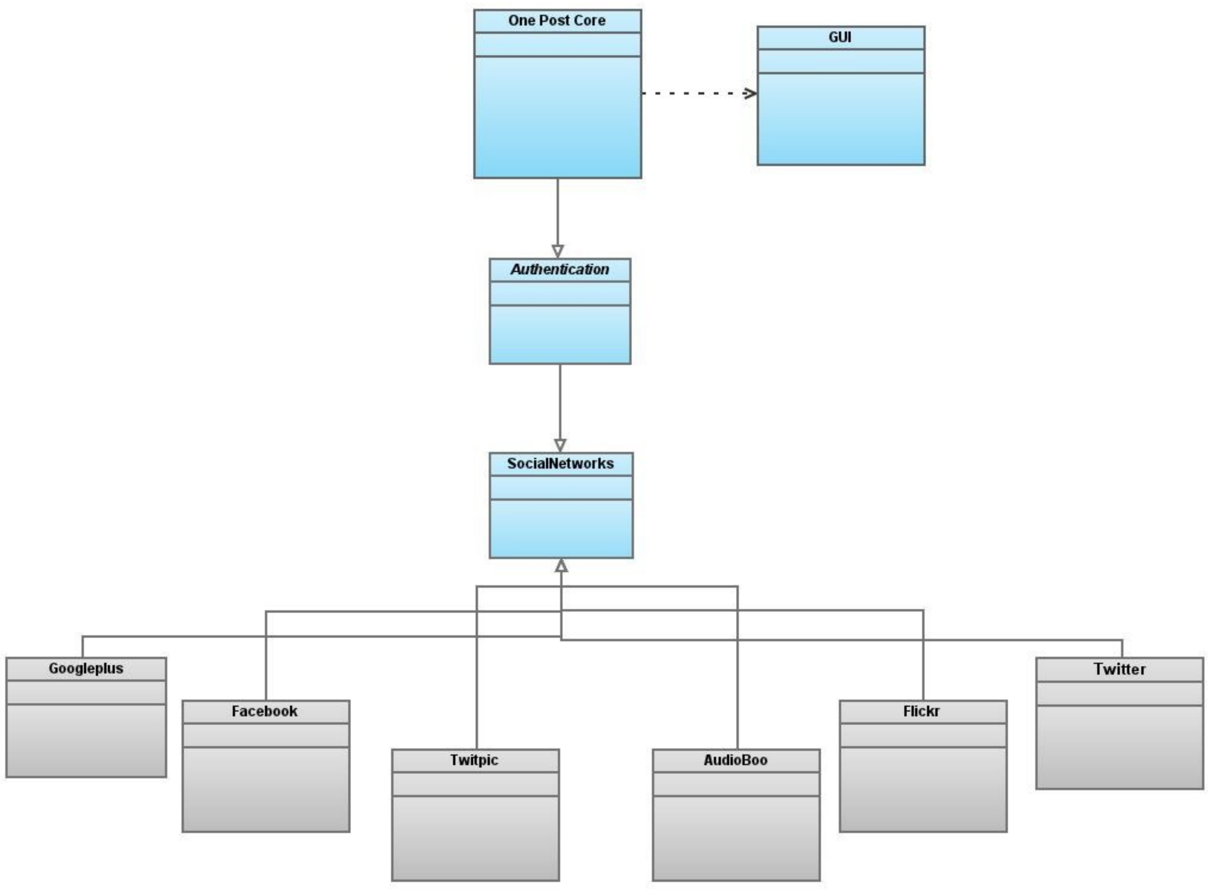
The above use case diagram shows how the user will integrate with the application and how the application/ social site will interact with each other. To minimise lines in this diagram social site represents the generalised for of each of the social sites I plan to implement.

Class Diagram



The above class diagram was a first draft of what my application might look like the one post core handles everything with 2 interfaces one for Authentication and one for posting content. In this structure when posting there is information all over the place, with great risk for code duplication.

Class Diagram



The above class diagram is a revised edition of the first, in this version we see that social updates rely upon authentication and to go to post an update authentication must be gained first. The Social Networks class has also been reconsidered. Although each social network site has its own API there are common aspects that are shared by all. As such the Social Networks class is now and abstract factory class instead of being an interface. This allows for more flexibility in the design, a new social network is already defined by the Social Network class, as such a new social site could be easily dropped in.

Implementation Options and Choices

This project doesn't have any complicated algorithms or processes. The main work of One-Post is fetching information and forwarding it to the relevant site, this forwarding is done by calling the correct elements of various social networking site APIs. All requests to the API are done via HTTP, this in itself is simple although it is made a little harder as One-Post is firstly written in Java and is secondly dependant on a mobile devices WiFi/ GPRS coverage. One-Post will be integrating with the available APIs from, Facebook, Twitter, Google+, LinkedIn, TwitPic, Flickr and AudioBoo.

Facebook, Twitter and Google+ were chosen because they are the most commonly used everyday social networks. These are the ones that are repeatedly in the press, whether it is Facebook helped find a missing person or Manchester Police are tweeting their day or how will Google+ face up to Facebook they’re in the press theyre in the public mind and thyre the most popular social sites.
TwitPic and Flickr were chosen because of their focus on photographic content. Twitpic is the main way to share photos over the Twitter network, and Flickr has long been a home of those showcasing their work both amateur and professional.
AudioBoo was selected because its new, because it takes a different form of content to all the other sites, as as such provides functionality that cant be achieve using any other site.
LinkedIn has been chosen because of its wide spread use by professionals from lots of industry. With the rise of social networking many people are now using LinkendIn to create an online profile which employers can view.

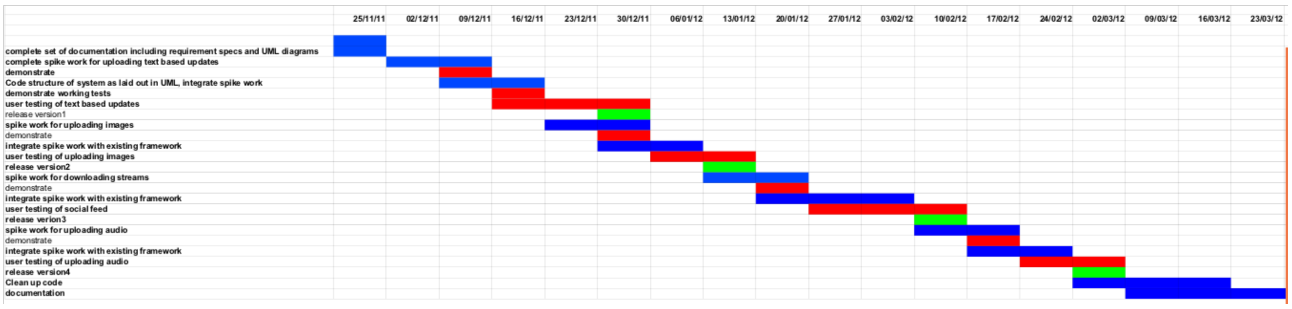
Integrating with some many APIs presents a challenge in itself, however having looked over each of the APIs there are similarities between all of them. Once I have implemented one API the others should fall into place. Some of the APIs are simpler than others and as such I aim to focus on these in the early stages of coding. By doing this I will be able to deepen my understanding on how to integrate with the APIs, this should give me valuable experience, that I will need to work with the more complicated APIs like Facebook.

Project Planning

Process Model

For this project I am very loosely following the Incremental Prototyping model. As an open source project the aim is to release early and often. This model allows me to regularly releases a prototype, each iteration adds a new prototype, eventually creating the final project in the form of multiple prototypes working together. Each stage in incremental prototyping is following a feature driven method, whereby a key feature of the project is chosen and developed to completion. In each stage I am also merging the implementation and testing phases and testing as I go along. By doing this I will be able to highlight bugs and errors in my code early and prevent a pile up of issues just before release. Although I plan to test as I code, I understand that some bugs will only raise their heads when other people use the application, as such I hope to catch the majority of these in small scale user testing prior to general release of the application. This focus on feature specific prototypes and heavy testing, should make my application more stable at release. As each release will give more content, I expect that users will use the most upto date verion of my application.
This modified approach to the process model takes inspiration from feature driven development, test driven development, incremental prototyping and the open source development model.

Weekly Plan for the Project



The work flow plan gives each iteration a time period of 1month from start to finish, this is a heavy load but should be achievable, there is a period at the end where by development can overrun, however this in not an ideal situation.

Demonstration Plan

Demonstrations will form and integral part of testing, as each section is completed the application will go into user testing. The Application in its current state will be presented to the users and they will be given a period of time to play with the devices and the application. These users will also receive an early release of the application so that they can test it on their own devices and report back with any bugs.
For user testing I will be supplying the Sony Ericsoon Experia X10, and the HTC Sensation. I will need to borrow an eee pad transformer and a HTC Desire. Users

with Android devices will receive a copy of the project as generated from my laptop. User taking part in testing will have to have accounts on Facebook, Twitter, Google+, LinkenIn, Flickr, TwitPic and AudioBoo. It is not essential for all testing users to have all accounts however if they have all accounts I will be able to get a more accurate and large set of test data.

Spike work and unit testing will also be used to demonstrate my work to my supervisor, so that progress can be monitored. Estimations on when these demonstrations will happen are shown on the Gantt chart shown above.

I also plan to demonstrate my project during a lightening talk at FOSDEM 2012. By doing this I will be able to actively introduce my project to a large number of the open source community. The aim of demonstrating at this conference will be to get feedback and hopefully input form the open source community allowing me to get feedback from a wider audience.

Annotated Bibliography

[1] AudioBoo, “http://code.google.com/p/audioboo-api/,” 2011, 27/10/11

A useful collection of documentation related to integrating with the API, The materials found on this site include procedures for authentication and examples such as how to upload images. From here developers can also create their Application key for needed to create a Facebook application and authenticate

[2] Facebook, “https://developers.facebook.com,” 2011, 27/10/11

A useful collection of documentation related to integrating with the API. The materials found on this site include procedures for authentication and examples such as how to upload images. Some of the advanced Facebook concepts are a little hard to understand however detailed examples help to overcome this. From here developers can also create their Application key for needed to create a Facebook application and authenticate.

[3] K. Fogel, “producing open source software, how to run a successful free software project,” 2005, pp. 33–35,45–87,231–241.

A simple guide to free software. The book doesnt focus on a specific project but points out key ideas for a for a successful project. The book covers key concepts and practices that allow a free software project to run smoothly, such asusing mailing lists and forums to engage with the open source community. Tips, tricks and general advice, a useful reference for starting a free software project, but also provides pearls of wisdom for dealing with those little project bumps. The book also gives a rather generic overview of some of the Licenses available to free software.

[4] Google, “https://developers.google.com/+/,” 2011, 27/10/11

A useful collection of documentation related to integrating with the API. The materials found on this site include procedures for authentication over OAuth2 and details for uploading and downloading content from a users page. From here 2 developers can create multiple application each with their own API key and Secret Key.

[5] D. Kuperman, “http://twitpic.com/api.do,” 2011, 27/10/11

Twitpic has a minimal API and links very heavily with twitter. Users login with their Twitter user name and password to upload photos. There doesnt appear to be any application specific security or registration.

[6] A. M. S. Lauren, “Understanding open source and free software licensing,” 2004, pp. 14–16, 62–80, 174–176.

Details of specific free software licenses and choosing the right license for a project. The page references focus on the licenses currently under review for the One-post project. This book covers each and every aspect of the main free software licenses, reviewing the legal jargon and giving a human readable review of what each bit means. This method of reviewing each section of the license allows for better evaluation of the licenses, allowing a more informed choice.

[7] LinkedIn, “https://developer.linkedin.com,” 2011, 27/10/11

A useful collection of documentation related to integrating with the API. The materials found on this site include procedures for authentication over OAuth2 and details for uploading and downloading content from a users page. From here developers can create multiple application each with their own API key and Secret Key.

[8] E. Raymond, “The cathedral and the bazaar,” 2002.

The Cathedral and the Bazaar is a rather concise book marvelling at how open source methodology is so different from main stream development techniques. The book goes through various aspects, such as communications and explains how the open source methodology shouldnt work, but some how does. The Cathedral and the Bazaar covers the main aspects of developing a project from a 1st hand view point and as such provides feedback on what, how and when to do things.

[9] R. Stallman, “Free software, free society,” 2002, pp. 165–192.

A rather bias view of the GPL licenses, giving simple to read explanations of the key points in each license. The collection of essays covering the GPL in its many forms, clears up all the legal jargon found in the licenses and makes it easy to read. In addition the essays make it clear that the licenses, although good at keeping software free, are well EVIL.

[10] Twitter, “https://dev.twitter.com,” 2011, 27/10/11

A useful collection of documentation related to integrating with the API. The materials found on this site include procedures for authentication over OAuth2 and details for uploading and downloading content from a users page. From here developers can create multiple application each with their own API key and Secret Key.

[11] Ubuntu, “https://wiki.ubuntu.com/MeMenu/,” 2011, 13/11/11.

Me-Menu is an integrated part of Ubuntu 10.04, 10.10 and 11.04, this feature is soon to be replaced. This page goes over the design and process details for the Me-Menu broadcast features, these are the features most similar to my application.

[12] Yahoo, “http://www.flickr.com/services/api/,” 2011, 27/10/11.

Flikr has a similar method for authenticating to that on Facebook, Google+ etc. The documentation found on here demonstrates each step of authentication as well as uploading im-ages and images with comments.

[13] G. M. Zigurd Mednieks, Laird Dornin and M. Nakamura, “Developing for android,” 2011, pp. 3–9, chapters 10,12,14,17.

A very useful text covering all Android topics, from the original set up of a developing environment through to handling data with SQLite, handling multimedia and integrating with APIs. The text uses gives simple walk through instructions at points but also uses complicated Java examples to demon- strate a idea. The text answers many questions about best practice implementation techniques. This book is useful both for the beginning Android programmer and the programmer who just needs a quick reference.

[14] Gwibber11, “https://launchpad.net/gwibber” 2011, 23/11/11.
Gwibber is an open source linux desktop, microblogging client. Gwibber intgrates with multiple social networks relaying updates to all accounts the user has provided credentials for. Gwibber provides valuable code that integrates with APIS that I have deemed complicated, whether I reuse the code or take inspiration from the Gwiubber source, this resource will be very useful to my project.

[15] AudioBoo “https://github.com/Audioboo/audioboo-android” 2011, 22/11/11.
This is the AudioBoo source code. The source is released under the MIT licesnce and as such is free for me to use in my project. Although I could reuse the code, I will more likely use the code as a working example from which I can take inspiration, this is because I am integrating with multiple social networks not just one.

[16] Google,”http://developer.android.com/resources/dashboard/screens.html”. 2011, 3/9/11

The Android developer site provides valuable documentation and tutorials, This is a great resource along with the O'Reily Developing for Android book. The developer site also provides statistics on Android devices, this has been crucial in planning.