Technical Spec v2 – Protagonist

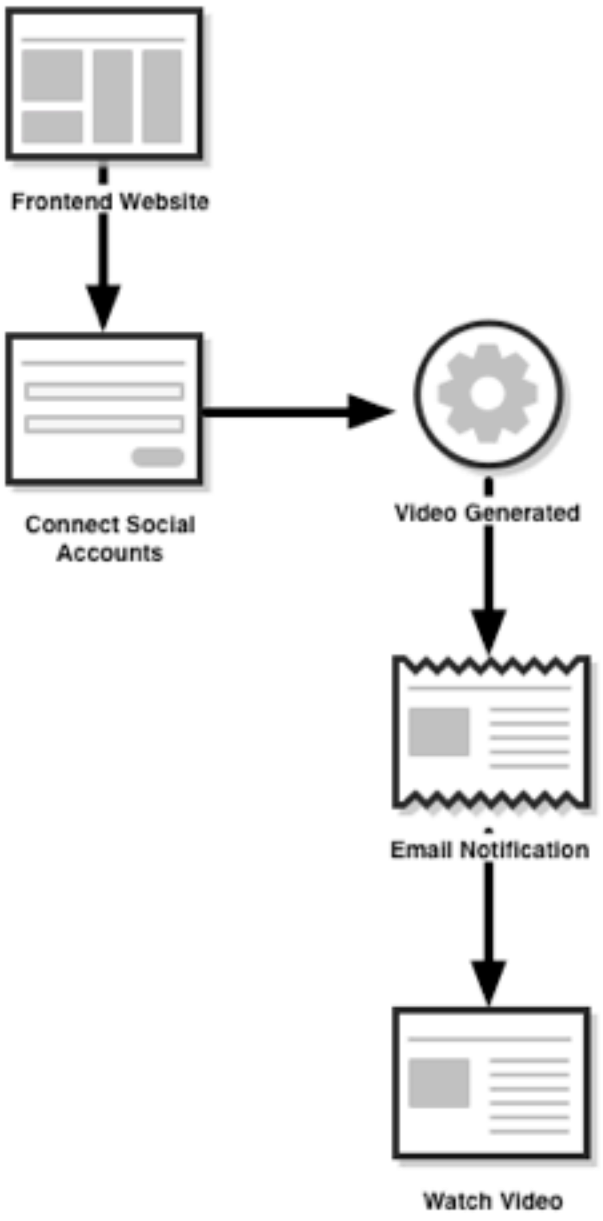
This document outlines a technical approach for building Protagonist, a personalised video created entirely out of media shared to personal social media.

Assumptions

- The service will require users to connect social media accounts to create a video. Facebook login is required. The other supported networks are Instagram & Twitter.
- The service is designed to process UK English social media language only.
- The service is designed to last for 6 months.
- The service is anticipated to generate 10,000 videos.

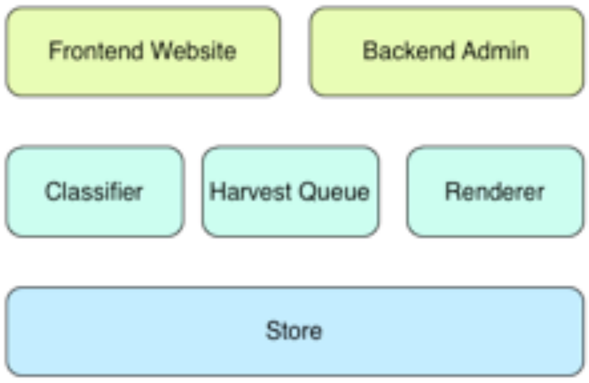
High level user journey

The user facing elements of the service are a website where users connect up their social media accounts. An email notification when a video is ready to watch and finally a page displaying the video.



Technical Architecture

The service is comprised of 6 key components.



Frontend Website

The frontend website is the only user facing component of the software. It allows users to sign up and begin the process of generating a new video. This section is content managed allowing admins to manage pages.

Backend Admin

Allows admins to view overall statistics, see all videos, tweak classifier settings and perform moderation. It provides graphs of the current state of the Classifier and the Harvest Queue, the Renderer and the Store.

Classifier

The classifier takes social media authentication tokens and begins to scan social networks for interesting content. The classifier connects to each social network via API, queries the data and passes interesting content to the Harvest Queue.

Harvest Queue

The Harvest Queue harvests highlighted data from connected social networks and stores it ready for the Renderer. User content is securely stored and labeled to prevent data crossing between users.

Renderer

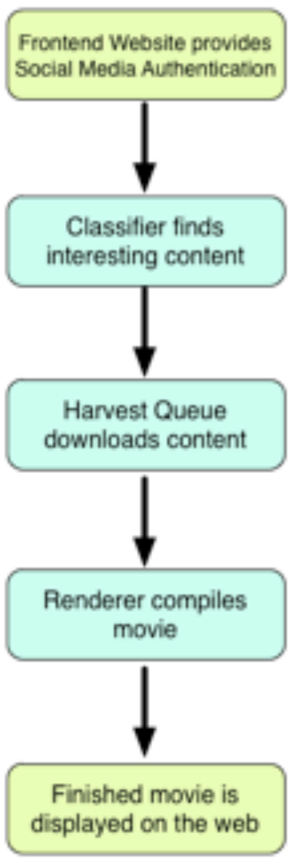
The renderer uses existing video templates and the content from the Harvest queue to render the video.

Store

Keeps user media until a video has finished rendering. Stores all final rendered video.

Video Generation: Step-by-Step

A high level overview of how a new video is generated.


Frontend Website provides Social Media Authentication
Classifier finds interesting content
Harvest Queue downloads content
Renderer compiles movie
Finished movie is displayed on the web

1. A user visits the Frontend website and connects up their social media accounts.
2. Once the social media accounts has been authenticated, the user is issued a unique ID. This ID makes up the unique URL of the final video e.g. http://videos.protagonistproject.com/watch/abc123
3. The Classifier is given the unique ID and any authentication tokens required to access the users social media accounts.
4. The Classifier scans the users social media accounts using classifiers and finds interesting content.
5. Content found by the Classifier is passed to the Harvest Queue.
6. The Harvest Queue downloads user media to the Store.
7. Once both the Classifier and Harvest Queue have finished, they start the Renderer.
8. The Renderer creates the movie and uploads it to the Store. It then deletes the original user media from the Store.
9. The user is notified that the movie is available and they visit the unique URL to watch the video.

Components in Detail

The following section describes the six core components. Each component is broken down into a list of key requirements. Followed by recommendations on best practices and approaches.

Frontend Website

The frontend website is the only user facing component of the software. It allows users to sign up and begin the process of generating a new video. This section is content managed to allow admins to manage pages.

Requirements:

- Allow admins to add and remove pages.
- Allow viewers to watch videos.
- Allow users to connect up their social networks.
- Uses a responsive template to allow users on any device to visit the site.
- Allow the video owner to remove the video

Allow admins to add and remove pages

The website should be built using a simple content management system such as WordPress. WordPress is widely available and easy to develop for, using an existing CMS will speed up development.

Allow viewers to watch videos

It is recommended that the videos use the following URL structure: http://videos.protagonistproject.com/watch/abc123 the abc123 fragment of the URL indicates the video ID.

Allow users to connect up their social networks

Users are presented with a form to connect their social networks to generate a video. Users are *required* to connect Facebook, but any additional networks are optional. Once they have connected the networks want to use, they are issued with a unique URL, which is displayed on screen and sent to the user via email.

Uses a responsive theme to allow users on any device to visit the site

The look and feel for this site can be custom designed or an off the shelf WordPress theme may be used. This theme should be responsive, this allows users on desktop, tablet or smartphone devices to use the service.

Allow the video owner to remove the video

When a video owner is comes to watch their video on the site, they may login (using the one of the network credentials they used to create the video) and opt to remove the video. The user is asked a series of questions about why they are deleting the video, this might give us some insight into potential improvements to the service.

When a video is deleted it should be immediately be moved from a public area of the store to a private area. The video is marked as deleted in Backend Admin and only admin users should be able to see the video. This will allow admins to check deleted videos for technical errors or glitches, which might have caused the user to choose to delete a video. After 3 days the video is automatically deleted

Backend Admin

The backend admin gives an admin a view into the overall state of the service. Including the number of finished videos and those still in progress, it also allows the admin change classifier settings.

Requirements:

- Allow admins to see a list of all videos and current state.
- Allow admins to moderate videos.
- Allow admins to change classifier settings.
- Allow admins monitor and control the state of the Classifier, Harvest Queue, Renderer and Store.
- Use a pre-existing PHP framework such as Laravel
- Use a pre-existing theme to save time

Allow admins to see a list of all videos and current state

Admins should be able to see all videos in the system and their current state. They should be able to see the current number being rendered, being classified etc

Allow admins to moderate videos

Every time a new video is generated an admin receives an email with a link for moderation. This backend admin panel will allow admins to hide or delete videos from the site.

Allow admins to change classifier settings

Some classifiers may have settings, which allow small changes to be made. This includes a list of culturally interesting events, or trigger words for personal achievement. Settings can be added, removed, disabled or disabled for a particular user.

Allow admins monitor and control the state of the Classifier, Harvest Queue, Renderer and Store.

Note: There is more about on-demand hosting later in this document.

Some of aspects of the service use on-demand hosting.The admin panel should display the current usage of these services and their current state.

Use a pre-existing PHP framework such as Laravel

A pre-existing PHP framework should be used to save time and reduce errors. Laravel would be a good fit for building this Backend Admin.

Use a pre-existing theme to save time

An existing admin theme should be used such as Twitter Bootstrap. This theme is responsive and provides ready made elements for the buttons, drop-downs etc

Classifier

The classifier finds interesting user content from social media accounts. Each classification step is broken down in detail, it lists the input data required, the APIs used and the networks it applies to. Some steps in the classification process might depend on a previously step.

We break this into three major sections:

- Friend Analysis, finding out who is important
- Media Analysis, finding out what is important
- Media Tuning, removing the junk.

Friend Analysis

A. Build blended friends list
B. Build affinity graph

Media Analysis

C. Find significant media from significant dates
D. Find significant achievements
E. Match significant historical / celebrity news

Media Tuning

F. Meme Removal
G. Human API video analysis

Video asset list

Once the classification process is complete, a Video asset list is generated. This contains a list of the media which has been recommended for inclusion in the video. The Harvest queue downloads this media and places it into the store, the Renderer uses as a guide to which assets should appear in the video.

A. Build Blended Friends List

Users might connect multiple profiles representing themselves. They will often share the same friends across multiple social networks e.g. Tom on Facebook, @tom Twitter and @tombot Instagram. We need to blend these into one, so we can match media from the same friend across Facebook, Twitter and Instagram.

We'll use the FullContact API to retrieve these contacts for us, as they've already crawled the web to join up these connection.

Input
- List of Email addresses
- List of Twitter
- List of Facebook profile URLs

Output
- List of Facebook contacts, with any connections to other networks

APIs and Networks
- Full Contact Person API

B. Build Affinity Graph

This will map out the friends / family the user is most closely connected with. Specifically we will look at the Facebook Graph API, to find family / spouse connections. These are marked as special relations can be queried directly.

Next we'll look at affinity over time to get a normal sense of who is important to a user. Events where a high number of photos have been taken can skew counting the raw number of photos that a user is tagged with. This might give a false sense of affinity for a family member or friend, if they have been to a wedding or holiday where lots of photos were taken.

We expect close friends to be tagged in repeat photos and events over time. Instead of counting the total volume of photos for a friend. We will award points for each day a user is tagged with a friend. For each user and day, they will get 1 point. We'll look back over 3 years of photos to find the most popular people and sort these into a list.

For example: Over the past year, I have been tagged with Katherine Jewkes, 55% of the time, Benedict Hodge 25% of the time, Richard Hewitt 11% of the time and Adam Keene 6% of the time. The users which have a higher percentage represent my closest friends.

Input
- Information about a users family from Facebook
- List of Facebook statues / photos from the past 2 years.
- List of Instagram photos from the past 2 years.
- List of Twitter statuses (with photos) from the past 2 years.

APIs and Networks
- Family Connections: Facebook FQL Family
- Facebook Tagged Statuses: Facebook Graph /me/feed/tagged
- Instagram Photos: Instagram API /users/self/feed
- Twitter Photos: User Timeline, filtered by Media

Output
- List of family members / children / spouses
- List of Facebook users ordered by popularity
- List of Instagram users ordered by popularity
- List of Twitter users ordered by popularity

C. Find significant media from significant dates

This process finds media around events which might be personally significant to the user. The first step is to get a list of a friends birthdays (this information comes from Facebook), these dates, plus the closest weekends are added to a list of significant dates.

The second step is to crawl a years worth of statuses looking for a list of event phases e.g. Birthday, Anniversary, Party, Wedding etc. The dates when these events are posted are added to the same list. Additional weighting is added for the number of statues found on the same day.

Once dates have been found and ordered by weighting, the third step is to use these dates to find media, across all of the users connected networks. Any media found which on Facebook or Instagram is ordered by our Affinity Graph (see previous process).

Example: Our users has connected Facebook and Twitter. Our users Birthday is March 24th, we find other statuses on Twitter containing the word birthday on March 25th & March 24th. We search for media on Facebook on March 24th, 25th and the following weekend. We find 60 photos in total. We order these photos by our affinity graph and return photos that the user and their closest friends ranked highest.

Input
- Users statuses from the past 3 years across all networks
- Users birthday
- Friends birthday
- List of keywords

APIs and Networks
- Facebook Statuses: Facebook Graph /me/feed
- Instagram Photos: Instagram API /users/self/feed
- Twitter Photos: User Timeline, filtered by Media

Output
- List of media taken around the time of significant events, ordered by affinity.

D. Find Significant Achievements

This process tries to find significant personal achievements and media associated with it. Significant achievements can be identified by scanning user statues looking for keywords which related to a particular achievement. A high number of keyword in the achievement whitelist increases the achievement score. Keywords in the blacklist remove

Each Achievement is classified by two word lists; a Whitelist and Blacklist, which can be controlled via the Backend Admin. They can also be put on hold or particular achievements can be muted for a particular user.

For example: An admin creates an Achievement in the backend called "Marathon". In the whitelist they add the words, "justgiving.com", "marathon", "cancer", "donate". In the blacklist they add the words "drinking" and "booze".

This will match statuses which contained: "I've just finished a marathon, so sponsor me on justgiving.com" but would not match statues containing the phases "About to go on a weekend drinking marathon".

Some level of pluralisation and tenses searching will be automatic, for example the term drink, will automatically match "drinks" and "drinking". Media from found events is automatically extracted and added to the media list.

Input
- Users statuses from the past year across all networks
- List of achievements
- List of achievement whitelists & blacklists

APIs and Networks
- Facebook Statuses: Facebook Graph /me/feed
- Twitter Statues: User Timeline, filtered by Media

Output
- Media from significant Achievements for a particular user. Ordered by high score.

E. Match significant historical / celebrity events

We'll build a database of culturally significant events ordered by importance. Each item requires a name, description, date-time and importance score. This can administered in the Backend Admin. An initial batch of UK events can be automatically imported from Wikipedia, with later events being added / removed by an admin.

We'll look for statuses which match the same dates as our celebrity events, any events found will be extracted and added to the media list.

Input
- List of culturally UK significant events
- Facebook Statuses: Facebook Graph /me/feed
- Twitter Statues: User Timeline, filtered by Media
- Instagram Photos: Instagram API /users/self/feed

Output
- Maximum 5 stores (preferring photos)

Networks
- Facebook
- Instagram
- Twitter

F. Meme Removal

Users often publish photos from other sources on their Facebook timelines. These are often liked and commented on by friends, but they are not photos of a particular user. Unlike video, it is hard to determine if this is user generated content or re-posted content.

Using the TinyEye Reverse Image service, we can check to see if images are original or have been posted elsewhere online. This will help remove non-user media from the list of items under consideration.

Input
- All user media currently under consideration

APIs
- TinyEye Commercial API

Output
- Media found on other sites is automatically removed from the queue.

G. Human API Video analysis

Automatically analysing videos for the most important moment is difficult. In order to Videos must be run through various facial recognition APIs, audio and diagnosis is almost impossible to extract without significant training of speech recognition software.

Our recommendation is that video processing is done by humans using a quick extraction tool. If no human is online the process the video, then this section is skipped from the rendered.

How it works:

1) The Facebook API is used to search for all Facebook uploaded video. This is ranked using the affinity graph and the significant dates.
2) If an admin is logged into the Backend Admin, they received a popup notification to moderate new video. This popup is available for 3 minutes.
3) The top 3 videos are presented to the admin user, using a simple flash based highlighting tool. The admin user highlights the most significant part of the video and presses save.
4) If done within the 3 minute window, then the video is added to the render, if not then this section is skipped from the video.

Input
- All user videos currently under consideration

APIs
- A human.

Output
- A short video clip, automatically added to the render queue.

H. Sentiment Analysis

Sentiment Analysis is the process in which text can be automatically searched to see if it contains emotional text such as "angry," "sad," and "happy."

A final list of user statuses will be processed by our Sentiment analysis. Statuses which score high for “sad” and “angry” will be removed. A list of terms can be adjusted in the backend admin.

Input
- All user statuses currently under consideration.

APIs
- A PHP based Naive Bayes classifier should be used to extract sentiment from posts.

Output
- Statuses with negative statuses are removed.

Classifier Summary

At the end of this process, the classifier will have looked at all user content. It has found important friends and family members, using Friend Analysis. Based on these results, it's then found important media around significant dates and achievements.. Finally Media Tuning has removed negative or meme based media from the list.

An (short) but finalised list of media from the classifier would look something like this:

```
{
	"user_id": 1234,
	"events": [
		{
			"type": "birthday"
			"mediatype": "photo",
			"mediapath": "12312312312.jpg"
			"text" : "my 28th birthday"
			"with": ["Katherine Jewkes", "Benedict Hodge","Kelly Williams"]
			"date": 21/12/13
		},
		{
			"type": "achievement"
			"achievement-id": "marathon",
			"mediatype": "photo",
			"mediapath": "12312312312.jpg"
			"text" : "finally did it!"
			"with": "Katherine Jewkes"
			"date": 03/6/12
		},
		{
			"type": "celebrity"
			"celebrity-id: "royal-wedding"
			"mediatype": "photo",
			"mediapath": "12312312312.jpg"
			"text" : "Eating a cake"
			"with": "Katherine Jewkes"
			"date": 10/2/12
		},

	]
}
```

This sample list, indicates three events which the classifier might have picked from my account.

The first is a photo from my birthday, the classifier has picked this because the date is significant, it's also a photo which contains three friends, who are ranked as closest.

Next it's picked a photo from the end of a marathon, the classifier has picked this based on an event and also because I was tagged with my friend.

Thirdly, the classifier has found a photo of me eating a cake which happened to be the same date as the royal wedding.

The Renderer will use this as a guide to which assets should appear in the video. For example, there might be a special video template for "royal-wedding" or "birthday" where as marathon might use one of the default templates.

Harvest Queue

The Harvest Queue uses the media list generated by the Classifier and downloads it ready for the Renderer. Each users content is securely stored and labeled to prevent data crossing between users.

Requirements:
- Scale reliably for thousands of users
- Fail elegantly and retry where possible
- Report our status to the Backend admin

Scale reliably for thousands of users
The Harvest queue needs to reliably download data from thousands of users. During peek activity there could be thousands of items to be processed. The queue processing should be done using Amazon Simple Queuing System (more details in hosting section).

An item in the queue should contain the following information:

- Unique user ID
- Social Network Origin (e.g. Facebook, Instagram etc)
- Authorisation Keys
- URL to download resource
- Number of retries

A lightweight client takes an item out of the queue and tries to download it to the Store. If successful then the image is removed from processing.

Fail elegantly and retry where possible
If any items are unsuccessfully downloaded, they are placed back into the queue to be tried again. After 3 tries, they are removed from the queue. Images should be validated when they are downloaded, to ensure they are not corrupt and are the expected image dimensions.

Report our status to the backend admin
Make sure that we're always able to report the status and size of items in the queue to the backend admin. This can be achieved by building a simple API which reports these statistics to the backend admin.

Renderer

The renderer uses existing video templates and the content from the Harvest queue to render the video.

Requirements:

- Build a video template
- Render the video

Build a video template

Using the media from the media list, a video template is made. This template file describes which video templates should be used for which media items. For example, a wedding item might use a special wedding video template, where as a photo would use a default photo template.

Example video template:

```
{
        “user_id”: 1234,
        “events”: [
                {
                        “mediatype”: “photo”,
                        “mediapath”: “12312312312.jpg”
                        “text” : “getting hitched”
                        “template”: “wedding-1”
                }
                ,
                {
                        “mediatype”: “photo”,
                        “media”: “123.photo”
                        “text” : “wow!”
                        “template: photo-1”
                }
        ]
}
```

This file is then send to encoding.com and provides the order in which to render the video.

Render the video

This is a complex part of the process, the media will be in a variety of sizes, formats and quality. It's important that videos are resizes and re-scaled to match the final video output format.

We'll use the hosted service encoding.com for all of these requirements. Using a set of templates, we can take a variety of input videos and transpose them over our set of template videos. The final part of the render applies our soundtrack. This service then returns to use a completed video which is then uploaded to the store.

We need to carefully consider what happens when we don't have 100% of the video from the user. Either we use stock fallback footage or sections of the video should be skipped entirely.

Store

Keeps user media until a video has finished rendering. Stores rendered video.

Requirements:

- Notify users and admins that a new video has finished rendering.
- Remove old user media.
- Securely store the final video

Notify users and admins that a new video has finished rendering

Once our render has a placed a new video in the store, we should notify users and admin via email that a video is ready to watch.

Remove old user media

All downloaded media should be removed from the temporary storage once the rendering is complete.

Securely store the final video

The video should be stored in a directory which is accessible only via the Backend Admin or via the unique ID given to the user. It should not be directly accessible via other methods.

Technical Approach

Broad principles which all project developers should follow while working on this project.

Use stable, widely available languages

A straightforward and simple stack of Linux, Nginx, MySQL and PHP will be used on this project. These technologies are stable, widely available, widely understood and easy to develop for.

Use off the shelf libraries and technology

Our time and money should be spent on building solutions for this project, not for building solutions which might be useful on other projects.

Consume services, instead of building and self hosting

Our goal is reliability and predicability. If we need a specialist or complex process, we should seek to consume these from third parties instead of self hosting.

Work on three screens

We should embrace responsive design so our project works on Desktop, Tablet and Mobile.

Security by default

We work by building each service to use secure connections. Starting from SSL in the browser to secure video transfer. User media should never travel across the unencrypted web.

Automated Deployments and Source Control

We collaborate and minimise risk by using source control. We work faster by automating software deployments.

Hosting & Services

This project requires that user media be quickly converted into a video which can be watched on both desktop and mobile platform. To do this in the most cost effective way, we've split up the components across different types of hosting which is most suitable for the task at hand.

- Permanent Hosting: Always available, flat rate per month.
- On-demand Hosting: Available when needed, flexible cost per hour.
- Third-party Service: Available when needed, flexible cost use

These costs assume we are going to create 10,000 videos over a 6 month period. (A full breakdown of costs, in table format is available at the end of this document) and that each rendered video is around 5MB in size.

Permanent Hosting

Permanent hosting is pretty simple, we rent a server and pay at the end of each month. The costs are fixed and the server is always available.

Frontend Website & Backend Admin

The frontend website and backend admin should always be available. They provide visitors a way to watch existing videos, to start to create a new video or for admins to moderate content. We recommend that this element of the project is hosted on a low memory Linode.

On-demand

On-demand hosting is flexible simple, we rent a server and pay for the number of hours used. This allows us to scale up when we are busy and switch off when no videos need processing. We'll use Amazon Web Services to provide us with on-demand services,

Classifier

Since Classifying involves looking at lots of user content, this process is memory intensive. It is difficult to estimate how long classification will take per user, up to 15 minutes per user is a reasonable guess. This process will be run on a single high memory instance. During high load the Classifier could be running for 8 hours a day. During low load it might run for just 2 hours.

Services

Image Queue

We'll use Amazon Simple Queuing System to maintain the queue of images to be downloaded. This is free for the first 1 million requests each month, so the total cost for the project is zero.

Renderer

This process is CPU and Memory intensive which requires software which can accept a wide variety of images and videos in a variety of formats. In this case, It's cheaper for us to find a provider which specialists in this process, instead of building, hosting, configuring and maintaining a system which does this flawlessly for 6 months

We'll use encoding.com which specialists in this type of video manipulation http://www.encoding.com/programmatic-video-editing/

Assuming for each user we need to convert up to 50MB of video, we will need to purchase $800 (£400) of credit.

Store

The store is where user content is saved during processing and final videos are stored. Due to the dynamic nature of the project, it is difficult to anticipate the total amount of storage required.

Therefore we recommend using Amazon RDS to store user account information and Amazon S3 to store both the user media and final rendered videos. Amazon RDS and Amazon S3 are both pay-as-you-go storage which will scales based on our usage.

There are costs incurred transferring media into Amazon S3. Assuming each video uses 40 photos and 5 videos and video is 2 minutes in length. Total storage for 10,000 videos will be around 50GB.

Misc costs

The full costs breakdown also includes costs for CDN (domain names, email notifications etc) these are all documented at the end of the document.

Testing

In order to best test the progress of the system, three test accounts should be commonly used to test the system. A basic users, a medium user and a heavy user.

The basic uses Facebook + one other network. They have the minimum number of media items required to create a video and has little activity on social networks.

A medium user should be a developer or producer on the project. They use Facebook + one other network and have an average number of friends and photos. They have hundreds of photos and statuses over the last 2 years.

A heavy user should be a someone close to the project which is heavy user of social media, they have thousands of photos across all three networks.

Beta Release and User Testing

When the project is functionally complete and close to release, a group of 6 beta users should be recruited for the project. Using a service like usertesting.com they should be remotely guided through the service. Asking them to connect their real accounts and to create a video.

This will give you useful feedback and allow the classifier parameters to be tweaked. For example, testing might uncover changes that can be made to the classifier which will improve the quality of the finished video.