System Demo Report

Team Name: Black Team

# Our Target Audience

Our target user is highly educated, affluent and connected to the world at large. They seek out the best and most targeted news with the limited time available in their lives. In the future, news will have to find them and will have huge incentives to do so. The incentives for the company that gets this right and can be trusted are huge. Old media companies (notably New York Times) currently occupy this space and other companies (notably Bloomberg) are coming at this from a position as a well known and trusted source. They spend a lot of money to reach these influencers. A gap is opening however, as new media brands are gaining in an unfamiliar space for the old companies.

These users are important as they influence our world, via their business and financial dealings. The world has changed, no longer do banks and financial institutions grab the best and brightest from top universities. Increasingly, the technology sector is seen as the most exciting and impactful industry to be in and to have knowledge of. With the entry of Apple News and Facebook Instant Articles, we can see that the big technology companies increasingly see this as the only way to keep these influential users interested in the services they provide. Users are not easily conned however, as they increasingly will see Facebook and Apple as the arbiters of news and willing to use that power for their own good.

The most important differentiator we bring will be our independence. It will be clear to the user the algorithms we use are influenced by their own will to control what they see and what they want to see. We give them choice and an independent voice that won’t be swayed by incestuous editorial relationships, since we will aggregate any quality source of information.

Our targeted audience will also include people in and outside the technology industry. Such people will have roles as diverse as - software engineers, project managers, line managers, students of many disciplines and naturally hobbyists who tinker with electronics and software.

# The Core Problem We Solve

Our system exists to personalise the news to the individual user. The current content of any major news website is heavily biased by editorial. We aim to specialise on the the technology sector to start, as we feel special purpose aggregators build more loyalty and should be more accurate. We aim to mix curation (of sources) and algorithmic tailoring to the individual user. We will use our narrow focus to hone in on what drives our users and their needs. This will also serve as a proving ground to get the fundamentals of our product working to a high standard. After this, expansion of our core technology and concepts should be much, much easier.

The core technical problem is that algorithms to choose news articles for the user generally aren’t very effective. We can see this issue by comparing traffic to individual major publishers, along with aggregators. If aggregators were 100% effective, very few people would visit front pages of major publications to see the latest stories. Yet, we clearly see a demand for aggregators in their current state. One of the largest, if not the largest general purpose aggregator is Google News, a small side business of Google. There is clear evidence that publishers see traffic from aggregators are very important and thus try to game the algorithms, leading to a lack of trust.

Another core problem is accuracy in tailoring to the users’ needs. We believe a mix of a **ranking system**, **curation of feeds** and **cues from the user** will aid us in trying to solve this problem. We can demonstrate this by the fact that major aggregators still do not try to personalise the news for the user, or take an all algorithmic approach (Google News) or an all user curated approach (Reddit/Hacker News). We believe both methods have their advantages and disadvantages, which is why our method combines a mixture of both.

We can graphically illustrate the problem, by showing how aggregators show offensive, inappropriate content and conflicts of interest. One example is offensive language being included in a Google News headline[[1]](#footnote-0).

<PROVIDE EXAMPLES HERE>

The closest competitor in the technology news aggregation space is Techmeme. Techmeme has no user personalisation, just a list of curated headlines to “suit” every user who visits the website. Other websites of note include the Reddit /r/technology subreddit and Hacker News. The Reddit technology section is heavily moderated, and posts can be gamed or moderators paid. Hacker News is run by a big venture capital firm, Y Combinator. As such, Hacker News is regularly used to hire for and promote the products of Y Combinator companies, sometimes with no clear demarcation of possible conflict of interest issues.

# The Technical Solution

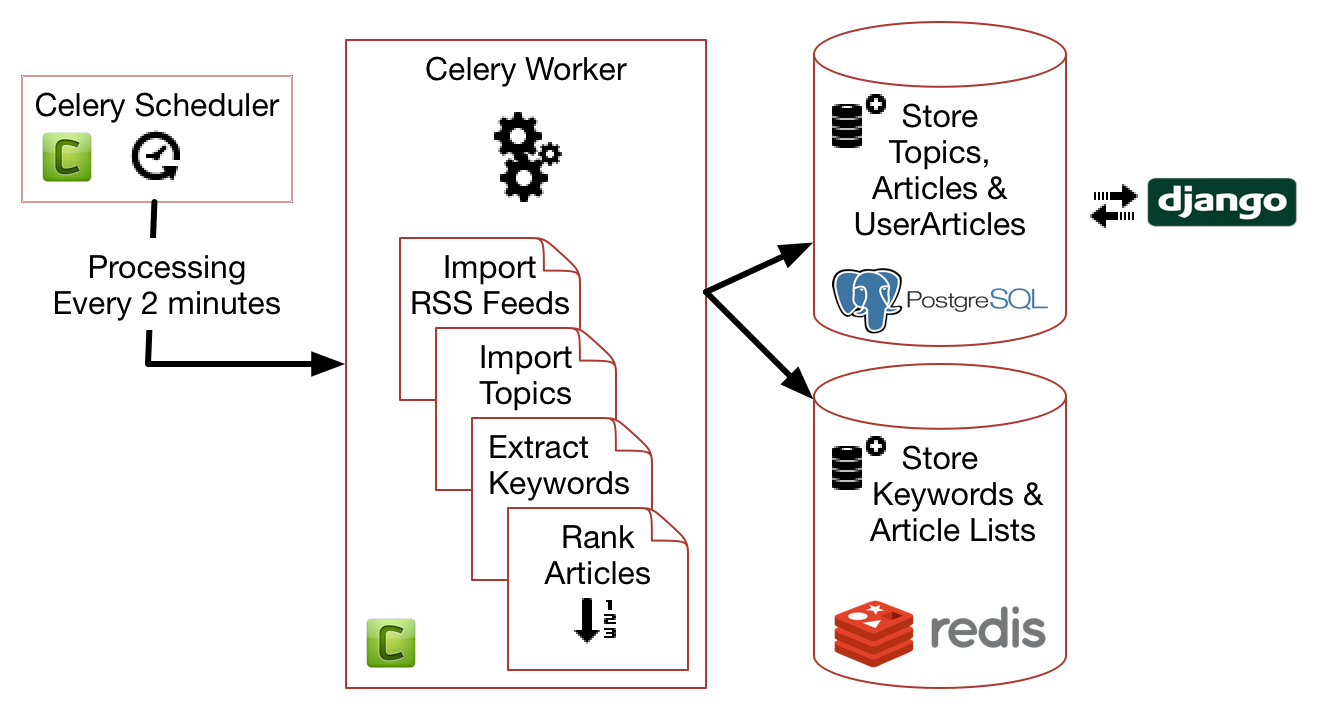
## What does it do ?

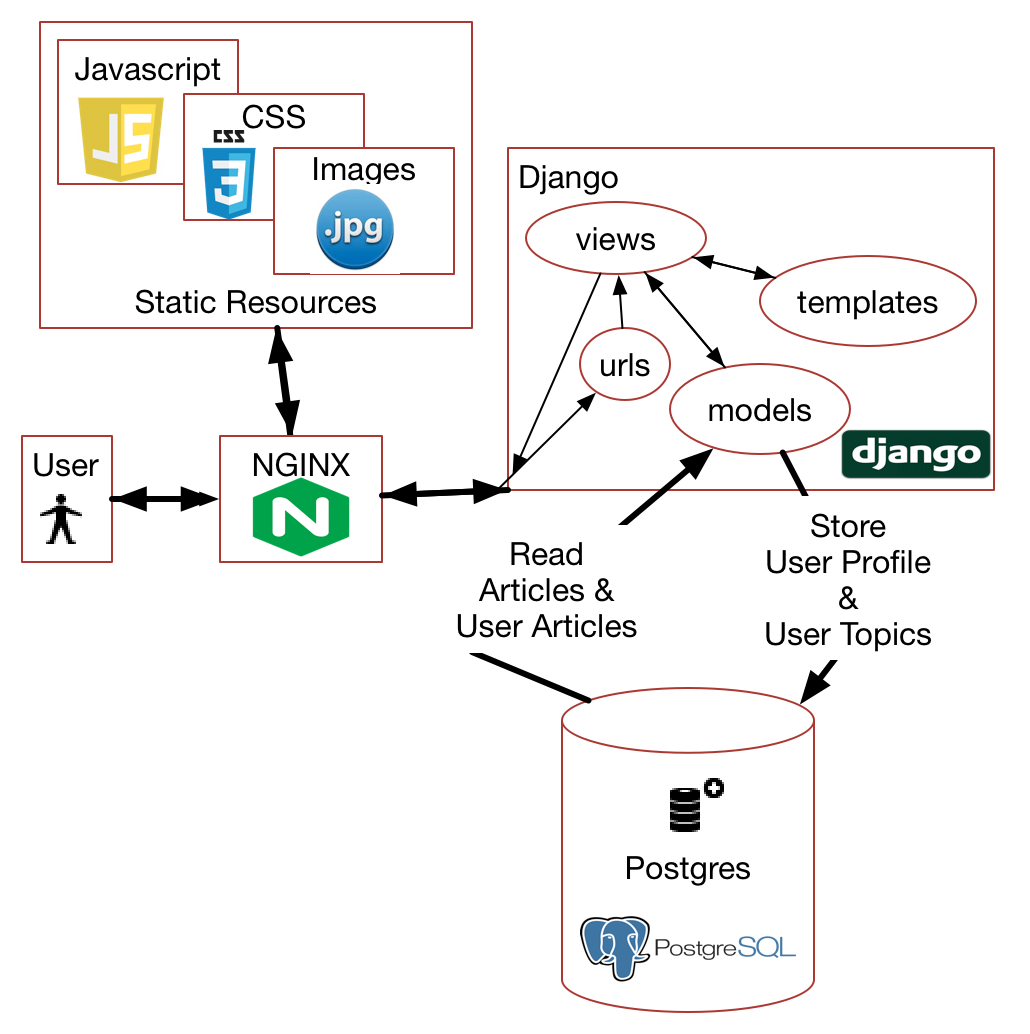
The Newsfast system:

* Imports a large number of RSS feeds related to Technology news - this can be ‘curated’ on the Django Admin GUI
* When the user goes to the website, they are shown default news articles with an option to sign in via their Twitter account
* When the user signs in, they can pick from a list of technology news topics
* The Technology news displayed on the website is then filtered or personalised according to the topics
* The articles are ranked according to a scoring system, the aim of which is to present the most interesting news to the user
* The system also learns user preferences, based on what articles they click on

# How does it work

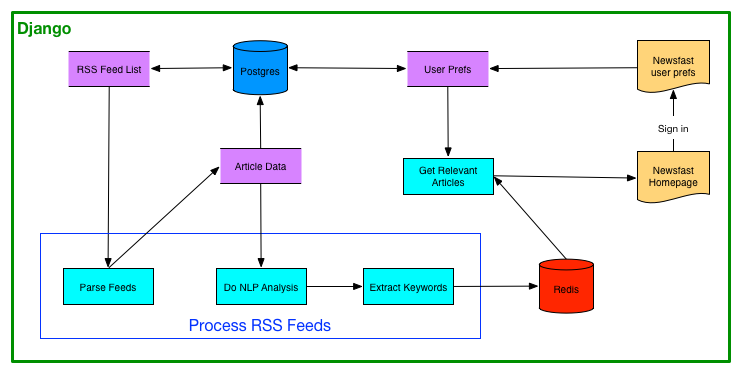
**System Diagram**





The diagram above shows the main system components, showing the docker containers and the interaction between them.

Logical system diagram



**System Startup**

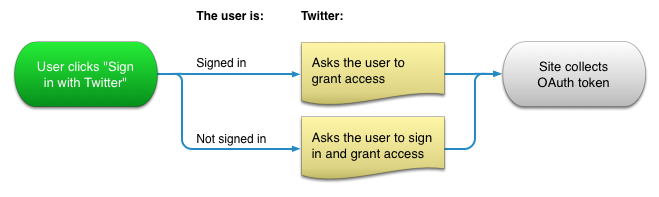
On startup the Celery Worker calls the following functions once only, this is to store important data into the postgres database where it can be edited on the django admin interface

* *import\_feeds* - read the RSS feed url information from file and store in the database. There are about 50 RSS feeds.
* *import\_topics* - read the global topic list and store in database
* *import\_tech\_keywords* - read the tech keyword list and store in database

**User Login**

When the user goes to the newsfast website for the first time, they have the option of signing in via Twitter. The user is prompted to grant access to the Newsfast application. If the user grants access, the application receives an access token from Twitter. This access token is stored as a cookie in the web session. The access tokens for each user are unique and allow us to get their twitter user id which is unique and doesn’t change (Twitter allows username changes).

The sign in process is shown below:



When the user login is complete, their Twitter Id and Twitter Screen name is stored in the Django UserProfile table. The Twitter Id is then used in our system to store other user data.

**User Selects Topics**

After Twitter sign in, the user can select tech topics they are interested in. The topics will be used to personalise the news for them. The user selected topics are stored in the UserTopic table.

**Processing RSS Feeds**

Every 2 minutes the Celery scheduler queues a task to process RSS feeds which is done by the Celery worker. Processing RSS feeds is the most intensive part of the system as a lot of data processing is done. The following steps are involved:

* Iterate through the articles in each feed
* Save the articles in the database in table Article
* For each user extract keywords from the article, according to user topics selected by the user
  + Keywords are stored in Redis for quick and easy access
  + If the article matches the user topic preference, then store Article ID and Topic ID as per user in the database (table UserArticle) - this will be used to display Articles relevant to the user selected topics
  + The user article is assigned a ranking based on a number of factors which are designed to ensure the most relevant articles are presented to the user

**Extract Keywords**

Keywords are extracted from each article and stored in Redis keystore. The extraction is done using nltk python library and our own python code. Keywords are extracted from both article title and article body and stored separately. In redis we use sadd to add to a set (set gives you unique values).

sadd format is sadd(key,value).

So for keyword android in article 20, we do:

sadd(android, 20) - this gives you set of articles per keyword

sadd(20, android) - this gives you set of keywords per article.

Example of set of articles per keyword:

‘android’: (20, 334, 3, 5, 67)

Example of set of keywords per article:

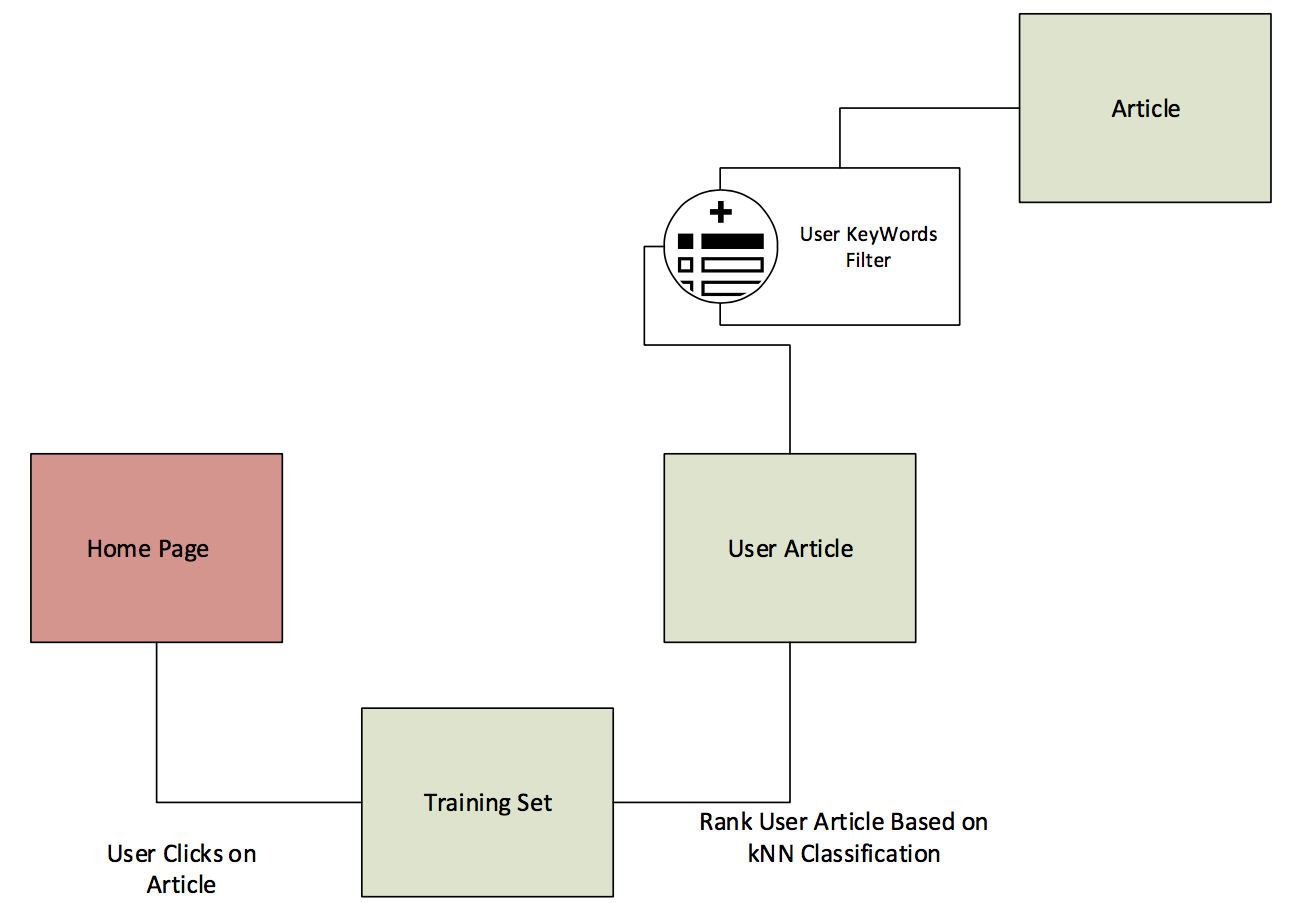
20: (‘apple’, ‘google’, ‘android’)

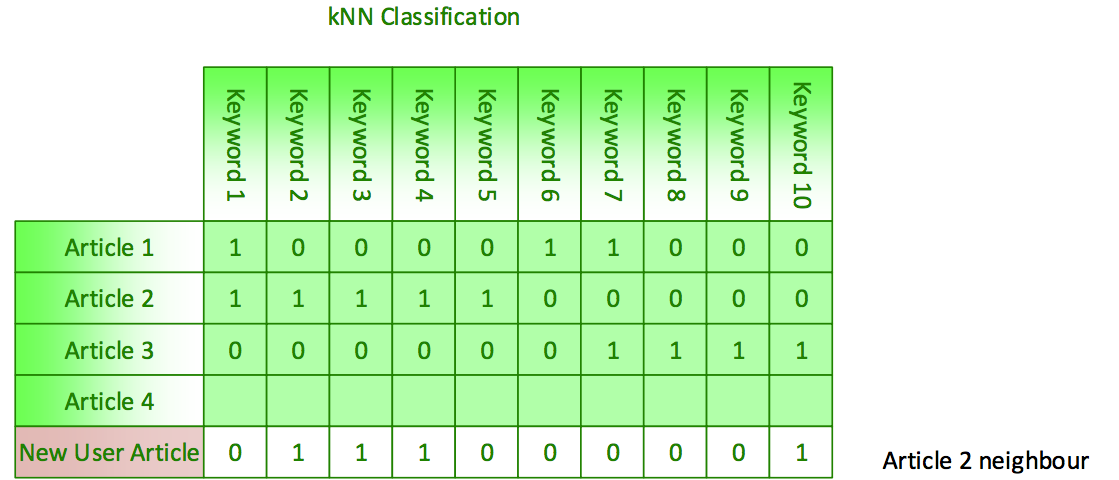
Once the extraction has completed this will be used elsewhere in the system to find articles matching a topic that the user has selected.

**Ranking Articles**

UserArticles are stored with a ranking value, the ranking value is the key criteria for displaying articles on the web page per user, so the ranking is key to personalisation for the user. The ranking scheme for articles is shown below, note that it is and can be adjusted during on going testing and customer feedback.

|  |  |  |
| --- | --- | --- |
| Factor | Value | Comment |
| Freshness (brand new) | +30 | First time article is processed |
| Freshness >= 12hrs | -10 | Every 12 hours, minus 10 to maximum of 32 hours (-30 points) |
| Source | +20 | Optional, can add +20 bonus to better quality sources |
| Has Tech Keyword | +10 | Multiplier, every tech keyword adds 10 points (no maximum for now) |
| Has User Keyword | +20 | Multiplier, every tech keyword present for user adds 20 points (no maximum for now) |
| Has picture | +20 | Once off score, may be adjusted to only work on sources with good quality pictures |

**Learning User Preferences**



An important feature in the system is to learn user preferences in order to recommend articles that they would be interested in, based on what articles they have previously clicked on.

The key points to explain how this is done is as follows:

* When the user clicks on an article, an entry is added to the TrainingSet table, this contains User Id, Article ID, and Tech Keywords
* After some time there will be a large number of TrainingSet entries per user
* When new articles in the rss feeds are processed, the new article is checked for similarity to the training set for the user, by counting how many keywords from the article also exist in the training set
* A rank is assigned based on the percentage of keywords from the new article that match the training set

# Front End - User Interface components

The user interface components are:

* Javascript
* jQuery
* Ajax
* Bootstrap

# Backend - Technical Components / Data sources

The backend technical components are

|  |  |
| --- | --- |
| **Technology** | **Reason used** |
| Docker | for build and deployment |
| feedparser | used to parse RSS feeds |
| nginx | web server, reverse proxy to dynamic/static files |
| gunicorn | python dynamic web server |
| redis | used for easy and quick keystore access |
| requests | used for HTTP encoding / decoding |
| selenium | used for automated web testing |
| tweepy | used to get Twitter information |
| authomatic | used for Twitter sign in |
| celery | task queue |
| nltk | NLP processing |
| django-admin-bootstrapped | used for Django Admin look and feel |
| coverage, coveralls, pep8, pyflakes | used for code quality |

**Docker technology**

We are building and deploying our system using Docker containers. They key benefits for us doing that are:

* Enables us to use continuous integration - code pushed to github is automatically pushed to the UCD VM
* Guarantees that we all use the same environment for local development and testing
* Software services are isolated in Docker containers - this reduces complexity, eases integration and also allows the containers to be scaled if needed. For example, to double our processing power we would just need to add a second Celery worker container

The Docker containers are as follows:

* NGINX
* Web - containing the main Django MVC system
* Postgres DB
* Redis Keystore
* Celery Scheduler
* Celery Worker (which can be scaled up or down by adding or removing workers)

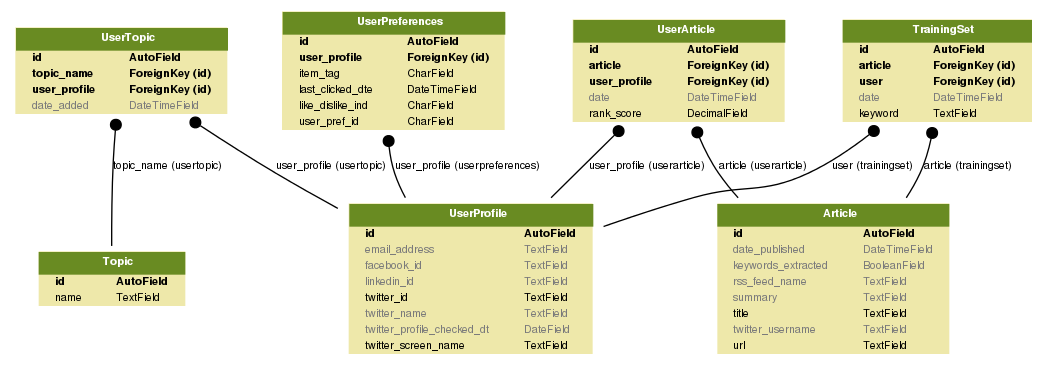
**Celery Worker and Scheduler**

Celery is a distributed task queue. The system has a Celery scheduler which allows us to schedule periodic tasks, and a Celery worker which does the task processing.

Using Celery means that we can keep the main data processing separate from the web service, which means the website continues to be responsive for the user while data is being processed intensively. The task queue is important as it means that the Celery worker will queue any pending tasks and process them when ready. Using Celery would allow us to scale the system if needed, by adding more Celery workers. Celery also gives us the option of moving the data processing to a separate physical machine.

**Newsfast ERD**

The diagram below is automatically generated from django models and shows the relations between django tables.



The tables contain information as follows:

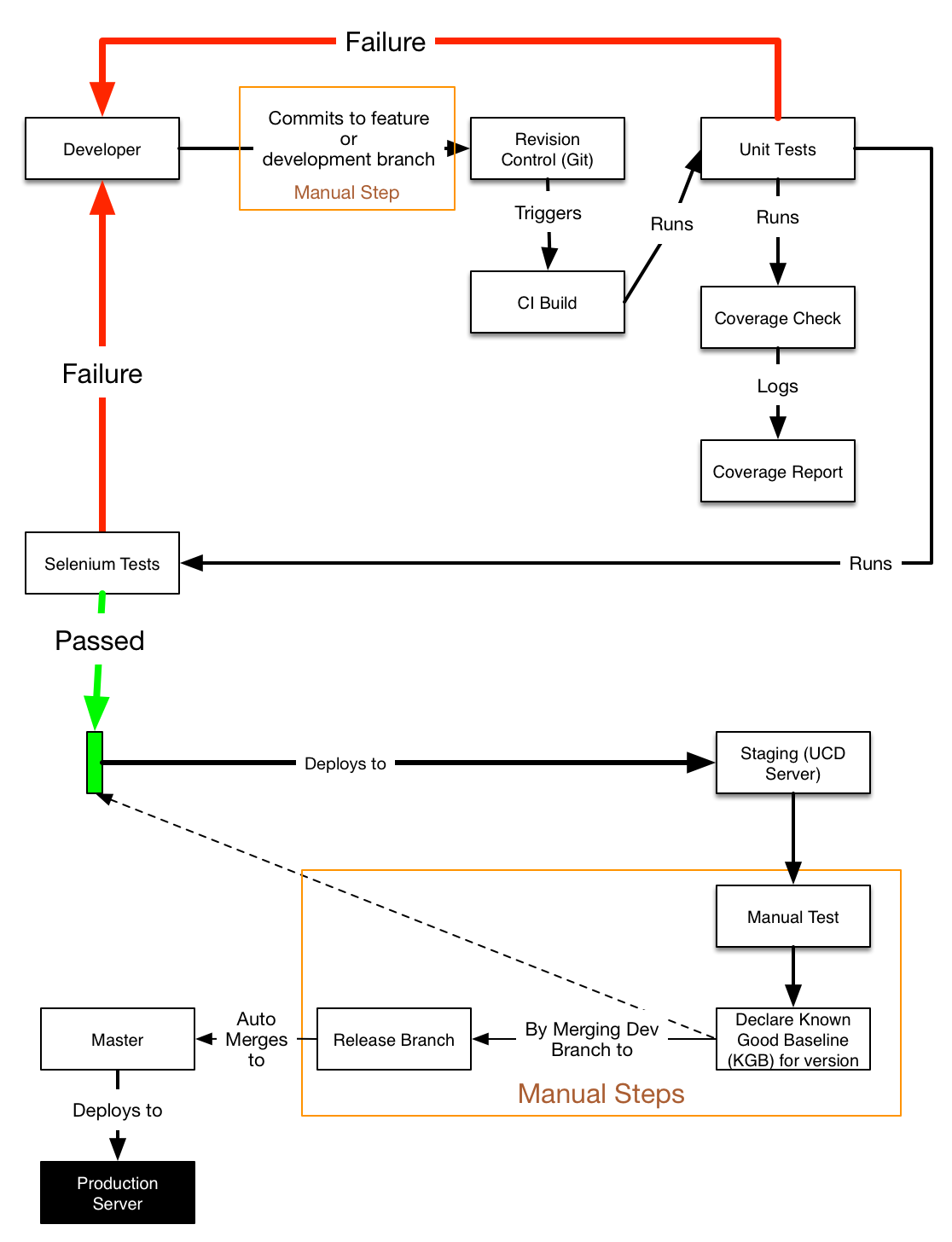
* Topic contains global topics entered by administrator
* **UserProfile** contains the twitter id and name. Any per user data will have a foreign key relationship to UserProfile
* Article contains the articles from the RSS feeds
* UserTopic contains the topics per user - it has foreign key of Topic and foreign key of UserProfile
* UserArticle contains the IDs of Articles per user - these are the personalised Article IDs which will be shown to the user. This table has foreign key of Article and foreign key of UserProfile
* TrainingSet contains ID of Article that User has clicked on. This is used for learning. This table has FK of Article and FK of User

**Continuous Integration**

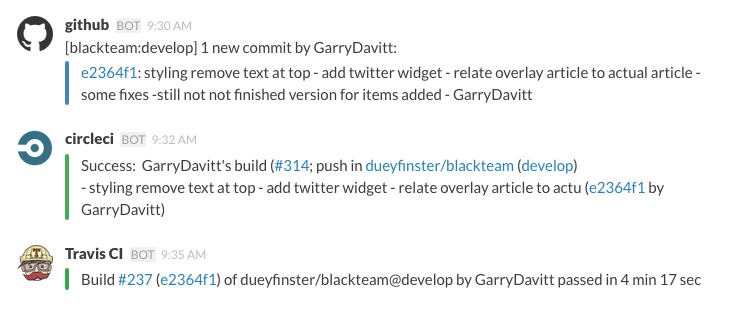
One of the benefits of Docker is that it enables us to use continuous integration, because the docker environment is the same on all our laptops and the virtual machines.

The development process is as follows

* Develop on local Docker environment
* Push change to github
* Circle CI and Travis CI automated tests run
* If tests are successful, automatic deployment to one of the UCD VMs
* We receive notifications from github, Circle CI and Travis CI on our continuous integration channel in Slack

The full CI process is shown here:

Example notification on slack is shown below:



# 

# 

# Example input/output and interaction with different components

This is an example walk through of how the system works, the technical detail is described earlier in the document

* The system starts up, the Celery worker imports the list of RSS feeds and defined Topics into the database
* Every 2 minutes the Celery scheduler queues a task for the Celery worker to process RSS feeds
* The Celery worker processes RSS feeds, storing each article in the database table Article.
* The User goes to the newsfast website for the first time. The newest articles from Articles are displayed.
* The User signs in via Twitter, their Twitter details are stored in UserProfile table
* The User selects Tech Topics that they are interested in, these topics are saved in table Topic for the User
* Once the User saves topics, a function is called to get Articles matching these topics, the Article Ids and Topics are saved in UserArticle per user
* Articles in UserArticle are displayed to the User
* Any new Articles added from RSS feeds are added to UserArticle for the user, if the new Article contains keywords matching the Users topics
* As the user clicks on articles, the articles are added to the users training set, this is learning the preference of the User
* When new articles arrive, they are checked for similarity to the training set and added to the UserArticle table if they are similar, with a rank assigned based on percentage similarity

# Integration

We are also using Twitter to authenticate our users using OAuth 1.0a technology. This provides us with a fast and efficient method or registering users to our website and to authenticate them every time they want to use it. Crucially the user will have limited interaction with their computer. If they are already logged on to Twitter then they will only need to click on a button on our webpage and the registration process will run in the background.

Google Adsense will provide us with our source of income. During the initial stages we will be relying on this technology as our sole source of income as it’s simple, fast and the content is managed by google.

Our website interacts with and enhances the the RSS & OAuth technologies to provide news content that is personalised. It is this personalisation aspect of our technical solution that is our Unique Selling Point (USP). There are many websites already in existence that collate RSS feeds from different sources into one destination for users. Typically these sites offer little or no customisation and simply show the latest news to all users regardless of their interests. NewsFast will use OAuth to keep track of the individual users. Initially the personalisation will be based on a list of topics that the users has selected as being relevant to them.

Risks: the main risk for the website, is that we are relying on free content from the RSS providers. There is no reason why they should provide this content to a successful website, especially if the website is commercially successful based on their content.

If the website was very successful the RSS provider could remove the RSS feed, or even just blacklist the newsfast website IP and continue to provide the RSS feed to non-commercial sites.

# The Impact We Have

Our system will be impactful in the fact that it blends curation and algorithmic processing to present the user with the best news for them. No other service seems to have catered to this market successfully. How we plan to do this is by aggregating the most successful news feeds/competitors and using them as feedback for our algorithmic choices. We see many competitors in the market who just do aggregation (Techmeme) or user-submitted aggregation (Reddit), but none have got personalisation done right. Arguably, Facebook currently has the best personalization and is now trying to extend this to news (Instant Articles). But it’s still a walled garden of outlets that have opted in and not the best of the open web.

We also think it is the correct approach to start in a small subsection of the news and expand the service later to cover more topics of interest. The best sources of really valuable content tend to serve their niches (Financial Times, The Economist) really well and then expand beyond that once they have a trusting and loyal audience. In this way, the impact, at least initially, will be a slow build. After all, the Economist went from a regional London title to a globe spanning operation over the span of about 100 years, surely we can replicate this in the digital age in a much shorter timeframe!

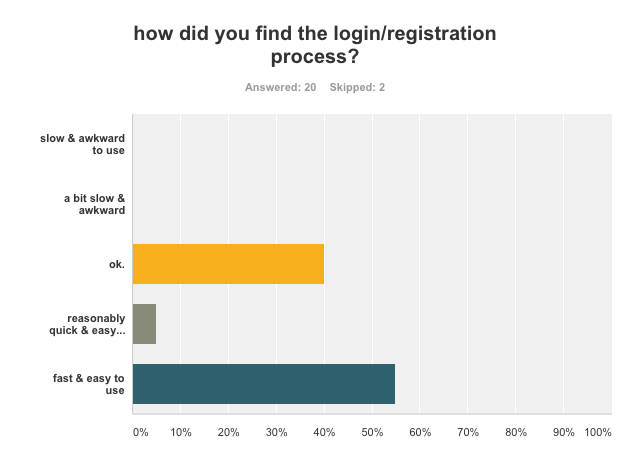
It will start having an impact once we can get appropriate scale to test and fortify our algorithms and feed choices. We expect to be able to grow fast if we can convince a core group of influencers in the media space to reference us as a news source. We can also aim to have a big social media presence, being referenced with the “via” annotation. People like to share what they feel is interesting and relevant to them, so it is by presenting them with this content we can start to get ripple effects of being shared across social media.

We also feel although mobile gets the most buzz and attention, actually the “desktop web” is the place to be able to iterate quickly. Much research (QUOTE) proves that users will immediately uninstall a mobile application they do not like. However, a website like ours, the user has no such ties to us and we can easily test retention and other key metrics, which are harder to record on mobile devices. Flipboard is a hugely popular aggregator on mobile, which also offers limited personalisation, but grew its audience by (QUOTE) 1,000% by broadening its offering and reach to a more traditional website. In summation, we feel we’re well placed to make an impact with scale and focusing on quality of personalisation.

# Reflections

How successful is the solution ?

From a technical point of view it has been moderately successful. The main features of personalisation are working. The user interface of the website is looking quite professional now. We would have liked to add more features to differentiate our website more from the competition. From a customer point of view we are working hard to improve the feedback we are getting, by adding features and improving the user interface. We are working towards competing with commercial websites by adding polish to our offering and tightening the user experience.



How appropriate were the choices you made about technologies to use ?

The choice of Docker to enable continuous integration, and ensuring we all have the same environment has been very successful. It has meant that we’ve never spent any time integrating work as the local environment is the exact same as the deployed environment.

It has been more painful to use for Windows users, and that has been frustrating at times, but it has certainly saved the group a lot of time.

Docker has also made some of the other technical choices easier, for example integrating Celery and Redis was straightforward as there were Docker images already available for these services. Redis was a good choice for storing keywords and article ids, as it’s well suited to storing data types like lists, dictionaries and sets. Using Celery for task queuing has worked very well, it means the intensive processing happens on the Celery worker container and not the Django web container - hence the website remains responsive for the user.

Celery also gives us a good story for scaling the system, we can add Celery worker containers or move the Celery processing to a different virtual machine quite easily.

On the user interface we choose Javascript, JQuery and Bootstrap. They are good, safe choices as those technologies are used in a large number of commercial websites. The python modules we’ve chosen have meant we’re not ‘re-inventing the wheel’.

What were the biggest challenges you faced as a team ?

Time is a constant issue, it’s difficult to juggle jobs and families with the project. Different people will have different time pressures which will then change. Another challenge is location, because the team is spread out with some people in Dublin, some in Athlone/Galway, some travelling in Wales. We’ve dealt with that very well using Slack chat for communication and Google hangout calls for the whole group at least three times a week. The group interaction has been very high despite not being able to physically meet up, so location hasn’t been a problem. Technically it’s worked out well, as we’ve tended to specialize as time went on, for example between back end, front end, and documentation deliverables. We’ve also switched around these roles as required, which keeps it interesting for us.

How effective were the project management and software development methodologies ?

We’ve used Scrum and Trello which has been quite effective. We split the time into sprints at the start and we have been good at updating Trello. It was challenging at the start of the project, where there was sometimes a conflict between agile “let’s get it working quickly” to “let’s agree a document on how it should work”. We found a good middle ground by having online session where we drew on a whiteboard and then made decisions about the structure of the project very quickly.

What lessons have you learnt - what problems are hard/easy ?

* working with people

One of the biggest challenges early on was scheduling. As we are a part time team, we have to constantly work around other commitments. This meant every team member had to be flexible in their approach to meetings and communicating. It also meant us being conscious to divide work carefully so we knew we could hit our sprint commitments.

* user feedback

At the start, we made the mistake of thinking we could just design the product for ourselves. This culminated in us saying we were our typical user at the halfway presentation point. We then quickly learned we need a lot of actual user feedback, from dispassionate people not directly involved. They could be critical and helpful, and they formed decisions for us as the project drew on. We even had the mantra of letting the user decide, and altering the survey to ask about the product as it evolved.

* garry UI challenges

Bibliography

“10 Principles Of Effective Web Design – Smashing Magazine.” *Smashing Magazine*. N.p., 2008. Web. 13 Aug. 2015. <http://www.smashingmagazine.com/2008/01/10-principles-of-effective-web-design/>

“120 Useful CSS3 Tutorials Examples And Tricks for Designer.” *Fresh Design Web RSS*. N.p., Mar. 2015. Web. 13 Aug. 2015. <https://www.freshdesignweb.com/css3-tutorials-examples/>

Alchin, Marty. *Pro Django*. Berkeley, CA: Apress, 2009. Print.

“CSS Overlay Techniques.” *Codrops CSS Overlay Techniques Comments*. N.p., Jul. 2013. Web. 13 Aug. 2015. <http://tympanus.net/codrops/2013/11/07/css-overlay-techniques/>

“CSS3.” *CSS3 Info RSS*. Web. 13 Aug. 2015. <http://www.css3.info/>

“CSS3 Introduction.” *CSS3 Introduction*. Web. 13 Aug. 2015. <http://www.w3schools.com/css/css3\_intro.asp>

“Django Development With Docker Compose And Machine.” *Django Development with Docker Compose and Machine*. Web. 12 Aug. 2015. <https://realpython.com/blog/python/django-development-with-docker-compose-and-machine/>

“Embedded Timelines.” *Embedded Timelines*. Web. 13 Aug. 2015. <https://dev.twitter.com/web/embedded-timelines>

“HTML(5) Tutorial.” *HTML Tutorial*. Web. 13 Aug. 2015. <http://www.w3schools.com/html/>

“How To Create an Overlay in Css.” *CSS Overlay Techniques,* Web. 13 Aug. 2015. <http://www.corelangs.com/css/box/overlay.html>

“Introduction To Celery¶.” *Introduction to Celery — Celery 3.1.18 documentation*. Web. 12 Aug. 2015. <http://celery.readthedocs.org/en/latest/getting-started/introduction.html>

“JavaScript.” *Codecademy*. Web. 13 Aug. 2015. <https://www.codecademy.com/tracks/javascript>

“JavaScript Tutorial.” *JavaScript Tutorial*. Web. 13 Aug. 2015. <http://www.w3schools.com/js/>

“Key Concepts.” *Knockout : Home*. Web. 13 Aug. 2015. <http://knockoutjs.com/>

“Welcome To the Docker User Guide.” *The Docker user guide*. Web. 12 Aug. 2015. <https://docs.docker.com/userguide/>

“JQuery.” *jQuery*. Web. 13 Aug. 2015. <https://jquery.com/>

1. "Offensive wording in headline of Google news article. - Google ..." 2012. 25 Jul. 2015 <<http://productforums.google.com/d/msg/news/94H2NxpE-nY/L6szH0Md7EsJ>> [↑](#footnote-ref-0)