CS3051 SCRUM Report

Group 3: 24/10/16

The news aggregator can be found at:
http://apps.thecharlesblake.co.uk/HotOffThePress/

Introduction

Our news aggregator is an “ultra-live” news service named *Hot Off The Press*. Our aim in creating an “ultra-live” service, was one that not only took in data from a large number of separate RSS feeds and showed them to the user, but also gave the user an update if the feed was updated. Our application can reliably take in data from over 50 different feeds and give the user an update if a new article is posted, within 30 seconds of the new article being posted on the feed. The aim of all of this is to give the user the sense that they are watching the world’s news being published in front of them in real-time - and hence that the news they’re receiving on our site is more “live” than anywhere else on the web.

Our UI has an “endless scroll” feature (not actually endless!) which allows the users to scroll vertically through the many articles in our main feed. On the left hand side we also have a filtering feature, which allows users to select articles by their news category. In the bottom right, we have also included a weather widget. When a live update comes in, a notification appears in the top right-hand corner of the UI, and the newly added articles are marked with a “new” tag. Our application aims to load quickly and also scale well to being used by multiple clients at the same time.

Throughout all of this we made a rigorous effort both to abide by a formal Scrum development framework, and also to document us doing so, by taking minutes of all our meetings when possible (including one audio recording of a sprint retrospective) and also taking snapshots of our Trello workflow board at various intervals. We carried out all of the requisite meetings during our two sprint cycles, and assigned different team members the roles of Scrum Master and Product Owner for each sprint.

Specification Tasks

Below is a summary of how we completed various elements of the practical with regards to the requirements listed in the specification. We list each requirement and our approach to fulfilling it:

a) Architectural / high-level design of the software

Our news aggregator, *Hot Off The Press*, collects a large number of RSS feeds from a variety of news sources. We used a Java Glassfish server backend to process the feeds and pass appropriate data to a HTML+CSS+JS client.

Using Glassfish allowed us to make use of Websockets to handle messages between client and server. This means our page updates with live news as it gets published to RSS feeds.

b) Team structure and role assignment

There was a clear divide in our architecture between client and server work so we split the development along these lines. Two team members focused on the client web page and the other 3 worked on the server back end.

c) Features included in the final application, and those considered but not included

Our product brings news headlines and descriptions from RSS feeds and displays them category by category (categories are determined from the feed).
Images (if available) are displayed alongside this information.

The user can filter visible stories by category, choosing only to read what they want e.g. UK news. We also have a weather widget included.

We considered having some way of identifying and promoting “popular” articles, as well as location-based services, but we decided both of these tasks were beyond the scope of our current application.

d) How features were broken down into tasks and assigned to team members

We used Trello for the bulk of our planning
(https://trello.com/b/iLb3ZABK/scrum-news-aggregator).
Cards were created in the product backlog list and moved into the sprint backlog at each sprint planning meeting. Team members would add themselves as “members” of the card to claim responsibility for that task (these were decided at each sprint meeting). Cards were then moved into the “in progress” list upon a listed member beginning work on that card’s feature. When a feature was completed i.e. it's checklist is complete it would be moved to the “completed” list.

e) The process used for, and the results of, the team's task estimation

As not everyone in the group was familiar with the architecture, it was difficult to find good time estimates. At first we used the estimates given by the most experienced members. These estimates were not always true in the end, as many tasks ended up taking a lot longer than expected. In the later sprint, we adopted 'planning poker', which gave us more realistic estimates. Although these estimates were more conservative, they were more reasonable as some members took a lot longer to complete tasks compared to others as they were learning new concepts.

f) The tools used, web-based or otherwise

As previously mentioned, Trello was our main organisational tool. A group chat on Facebook provided us with out of office communication. We found Trello to be very effective for managing the Scrum development process due being able to easily move our tasks around from backlog to in progress to completed. For our server we used Glassfish, for our frontend we used Javascript/HTML/CSS and JQuery and to deploy it we used AWS.

g) Meeting outcomes: sprint planning, sprint review, sprint retrospective, and an overview of stand-up meetings (i.e. daily meetings need not be documented individually)

We had a few problems organising regular stand up meetings due to clashing time schedules (3 joint honours students and 2 CS all with different modules), but managed to conduct a few in person standups. On a couple of occasions we did have to do Facebook standup meetings, with every member giving brief status updates on what they'd been working on.

On our first (pre-sprint) meeting we came up with the large list of features that we might like to have in the product. This was the basis for our product backlog. We then spent a few days individually writing extra user stories before we had another meeting. In this "user story consolidation meeting", we looked through the product backlog and eliminated some "duplicate" stories before finalising our product backlog.

In our first sprint planning meeting we also began moving product backlog into the sprint backlog. The first features we wanted to work on were getting the RSS feeds

collected and parsed by the Java backend while the web page developer(s) worked on a prototype UI. These features were mostly finished by the time of the second sprint at which we decided to add extra features such as images and category filtering.

h) Reflections on the Scrum process, and whether you would choose to use it again

We see the benefits of Scrum in getting a small group to work on large tasks. As a result of this, many of our members have set up Scrum-like workflows for their JH projects. We feel Scrum would work much better for a group of people that all share the same schedule e.g. working at the same company on the same shift. Most of our Scrum problems came from scheduling conflicts due to being a group of mostly joint honours students with different subjects. Accountability is important for Scrum and working in the same office at the same time increases it. Developers will be more inclined to finish the work they told their team they would do when the team is doing all of the work in the same physical space.

Design & Implementation

Our design aims when it came to the front-end were to keep things as simple as possible. From a UI perspective, this meant a simple (although still highly functional) interface, which was easy to use and implement. Our design, with the infinite scroll capability, was inspired by social media sites such as Facebook and Twitter.

We chose to keep the backend *logic* simple, so that most of the work (e.g. polling & parsing RSS feeds and managing the resulting data) could be done on the server. This was because the client's performance would be severely degraded by having to poll all of our RSS feeds (50+), transform the data and then insert them into the DOM tree.

To further improve the speed at which we can relay news to users, we also didn't want clients to have to poll our server for updates; we wanted a full-duplex communication channel in which the server could send clients updates when updates were available. This led us to choose websockets as our method of client-server communication. Once updates are received by our client, it is a simple matter of creating the relevant HTML structures and also keeping track of the category filtering logic.

However, using websockets also had its potential drawbacks. We wanted to use java on the server-side, and the only way of using websockets on the client-side any of us

were familiar with was node.js. Thus, before we began the first sprint, we went away and researched our options for java servers which could support websockets. Initially we thought that Apache Tomcat (partial Java EE server) would satisfy our needs, but in the end we weren't convinced it would suffice, and instead opted for a full Java EE Application server, in the form of Oracle's Glassfish server.

This server technology took us quite a considerable time to get used to and understand, but in the end it offered us some substantial advantages. Using the Java EE JavaBean service allows us to run our RSS feed parsing as a constant background service (by annotating it as a singleton session bean, running on startup), while Glassfish's support for websocket endpoints allowed us to service multiple user requests, all at the same time.

The first time our feed polling service runs, it populates a data structure which stores the last five items from each feed in a queue. Then, it checks all these feeds every 30 seconds (using Java EE's asynchronous Timer service) and updates the internal data structure if there are any articles that weren't there before. If this is the case, it also sends out this new data to users. It does this by first, converting the new data to JSON, and secondly, by iterating over a data structure (stored in our websocket endpoint class) which holds all the connected user sessions, and posting a message to each one in turn.

When a new user connects to our server, because it's internal state at any given point holds the 5 latest articles for each feed, we are able to very quickly turn the current state into a JSON object and send it to the client straight away. This is done by using the websocket's "onOpen()" method to make a call to our singleton session bean running in the background, which generates the JSON representation of our current state. This means that when a UI connects to our websocket endpoint, we can generate and send a long list of stories for them to view, without the client having to wait long at all.

The result of this architecture is a scalable, robust and surprisingly simple application, which is able to reliably give users updates within 30 seconds of a story appearing on an RSS feed. Furthermore, because so much of this logic is on the server, the client runs on a modern browser with no noticeable lag at any point, without having to wait long at all for the page to load either.

Once the application was written and user-tested, we then took steps to deploy it at a publicly accessible location. This proved to be much more challenging than we anticipated (we didn't include it as a Scrum task - but maybe we ought to have done)! We encountered some unusual and problems attempting to deploy it on the school host servers relating to running out of memory, so we decided to switch to AWS.

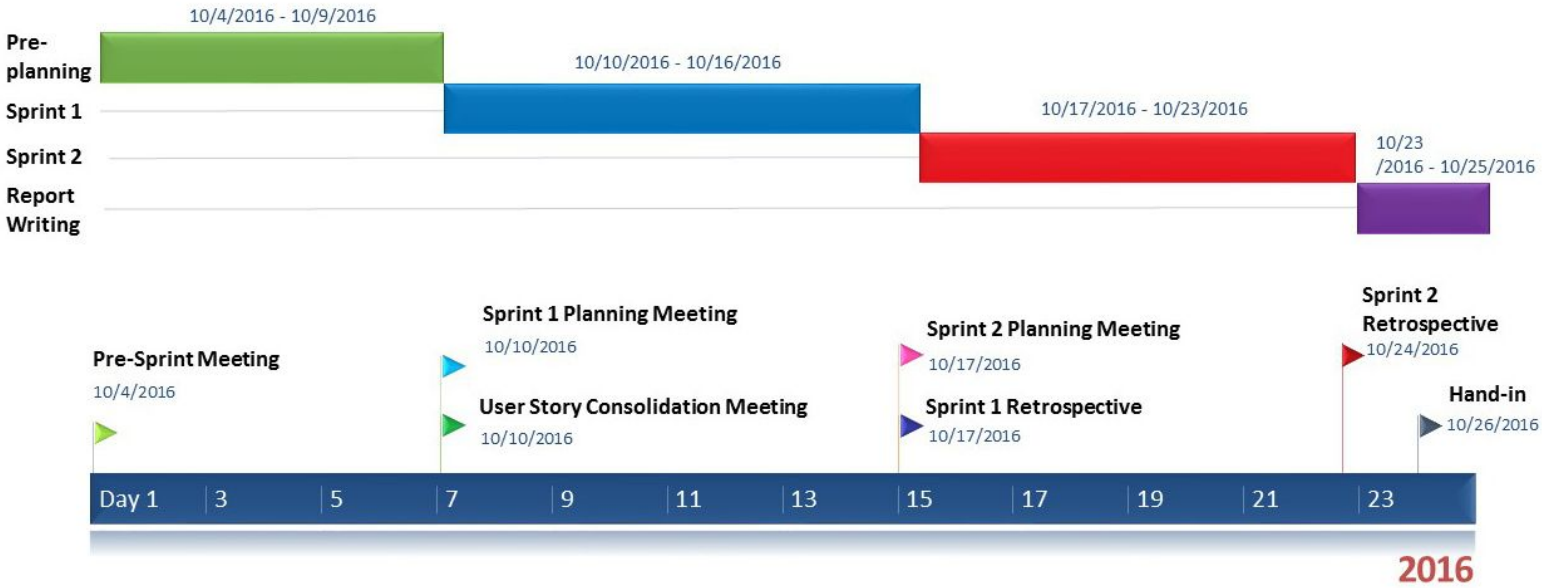
Our initial attempt to run it using a Glassfish container hosted on AWS's Elastic Beanstalk service failed, due to what we think was a lack of support for websocket functionality (we may be wrong here, but the error was proving hard to detect). Our backup plan was to install our own Glassfish server on a basic EC2 Ubuntu server instance. We did this, and eventually got it to work, but only running on port 8080. To redirect our traffic to port 80 we used EC2's Elastic Load Balancer feature, in the capacity of a port-forwarding service. The final deployment step was to associate the AWS url our instance had been auto-assigned, with a pre-purchased domain (owned by one of our team members: "thecharlesblake.co.uk"). We did this using AWS's Route 53 DNS service.

If you wish to compile and deploy your own version of our code, what we have submitted is our project folder from Eclipse. This can be compiled into a WAR (although I'm not sure how simple this will be outside of the IDE) and dropped into a Glassfish 4 Server. One small line may need to be changed if you wish to run this on localhost, which is the websocket connection url in our index.js file.

Scrum Documentation & Chronology

Timeline

At our pre-sprint meeting where we discussed how we would approach the project in general terms, we drew up the following timeline diagram which we then stuck to as the project commenced:



We also conducted our sprint reviews at the same time as the retrospectives, meaning that we did not have to keep having lots of small meetings at different times.

Pre Sprint Meeting

We decided on the project architecture very early on (see section **a** for details) and planned at this point that we would split the team between front and back end due to the whole team being proficient in Java but only two of us having solid web technology knowledge.

Some of the “obvious” user stories were written collaboratively at this meeting (“should be able to see news headlines” etc) and then went away to write more individually. We merged our individual stories (there were some duplicates) at the next meeting.

Ideas were floated about how to make the UI “interesting”. A graphical/interactive UI was mentioned as a possibility as an attempt to move away from a traditional list based news aggregator. Unfortunately this didn’t make it into development due to time constraints.

After the pre-sprint meeting there was in initial period where our various group members were finishing off other deadlines, and we were able to begin preparing for the first sprint on the 10th.

User Story Consolidation Meeting

After our initial Pre-Sprint Meeting, we all agreed to take some time away to think about what possible user stories to include in the Product Backlog, which we each added to the Trello Scrum Board for discussion at the User Story Consolidation Meeting. At this meeting we looked at all the proposed stories and discussed as a group which we thought were most appropriate/desirable for our product. We noted that many of the stories we'd created individually were similar to each other, apart from perhaps a few small differences. Because of this, we were able to merge many of the user stories, keeping the fundamental idea the same but adding novel features from one idea where appropriate. We also thought about what was going to be realistic as an end goal, given the time constraints imposed by the deadline, as well as the varying abilities of the group members. After discussing the stories, taking into consideration all of the above, we settled on a final product backlog, with the stories outlined in it becoming the basis for our sprints.

--- Sprint 1 ---

Sprint 1 Planning Meeting

In the sprint planning meeting we first decided that we would focus more on the aggregation of the RSS feeds in the first sprint, as this was the main goal of the practical. With this in mind we then selected user stories for the sprint backlog. As many of the team had other deadlines in the week, we kept the sprint backlog small and so only completed the main tasks, e.g. displaying the titles and links of articles on the web page.

These stories then had to be broken down into tasks, which was displayed on the Trello by the use of a checklist on the cards. We discussed time estimates for each task, but as some members of the team were unfamiliar with aspects of the architecture this was quite difficult. The estimates were therefore decided by the member with the most knowledge of the architecture.

Sprint 1 Timeline

10th Oct: Sprint start

11th Oct: Stand-up meeting (in person)

12th Oct: Started work on researching RSS feeds and storing RSS feed data

Wrote basic HTML page for UI **(task complete)**

13th Oct: A list of potentially useful RSS feeds was compiled **(task complete)**
Managed to read data in some form from BBC feed into Java code **(task complete)**

Managed to extract title from RSS feed **(task complete)**

14th Oct: Stand-up meeting (in person)
A parser to obtain basic information from RSS feeds was completed **(task complete)**

Created simple initial data structure to store Feed data **(task complete)**

Set up and tested a Glassfish server **(task complete)**

15th Oct: Stand-up meeting (in person)

Basic websocket communication between client and server **(task complete)**

16th Oct: A prototype of the webpage was made

When the user connects to the webpage, some unformatted information from the RSS feeds is sent to the page **(task complete)**

Included a hyperlink in the sent data so users could go to the story **(task complete)**

Sprint end

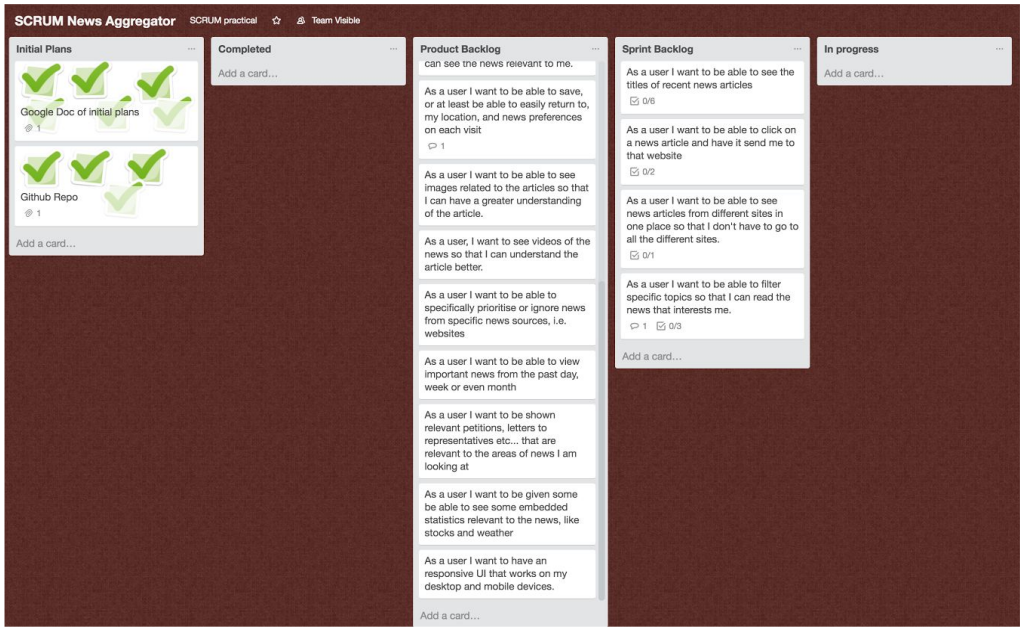
Had to move our one incomplete story in the sprint backlog back into the product backlog.

By the end of Sprint 1 we had a fairly basic, but still certainly shippable product.

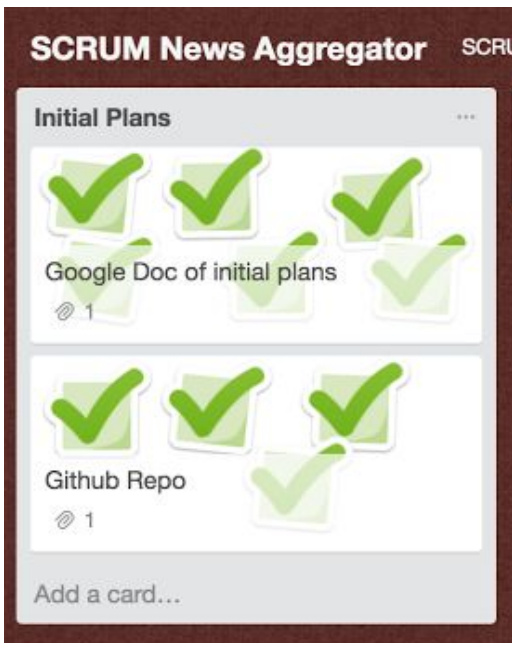
Sprint 1 Trello Screenshots

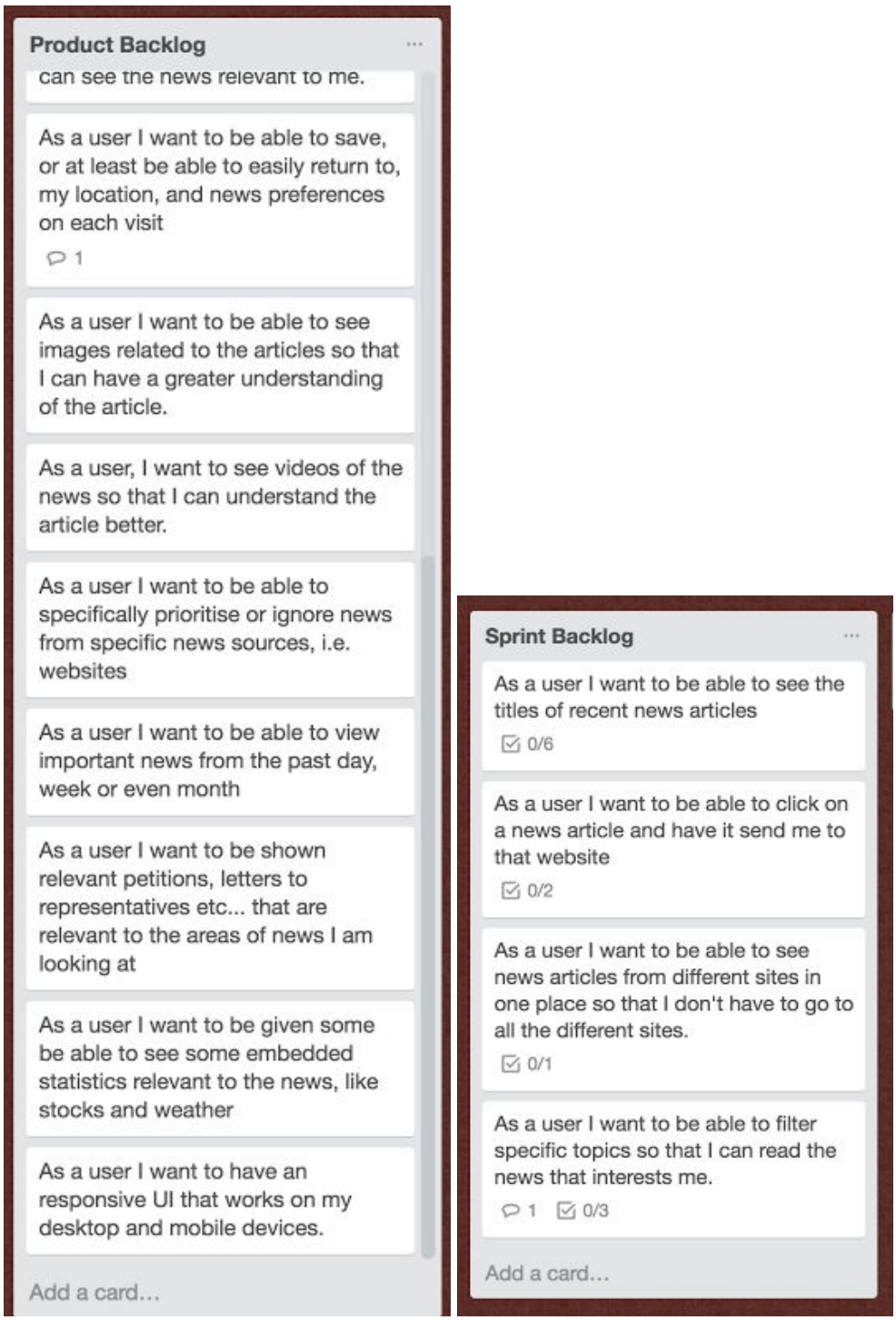
The following images display our Trello board at the start of our first sprint:

Overview:

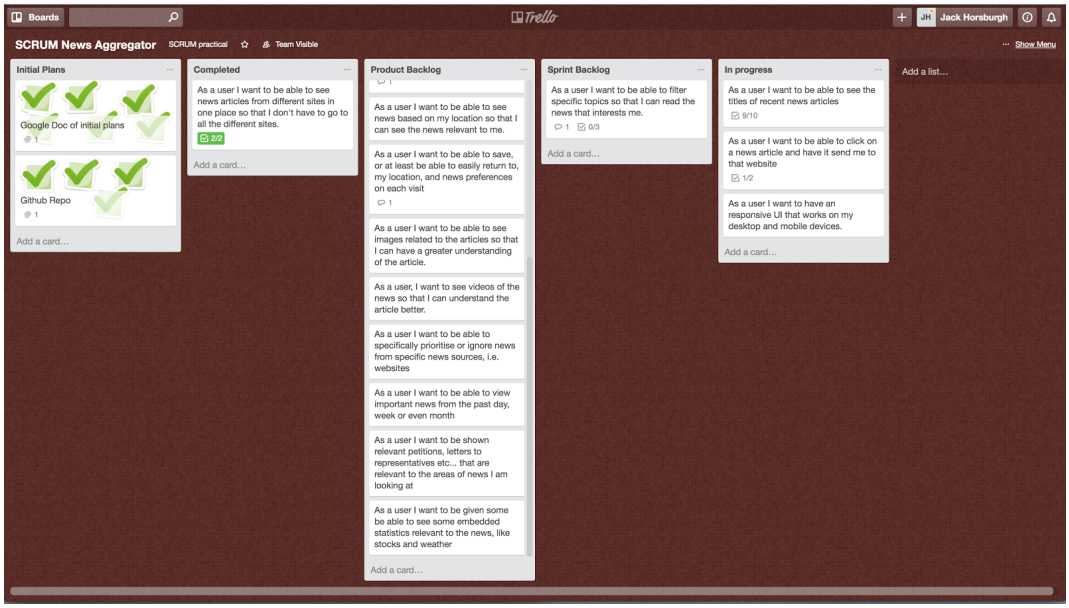


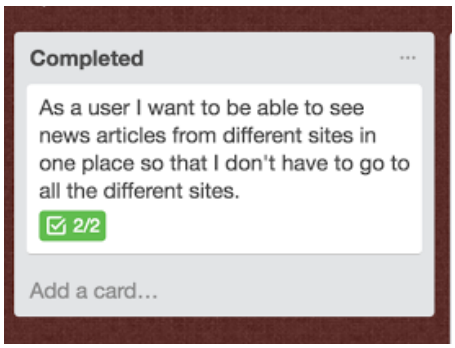
Zoomed-in Sections:

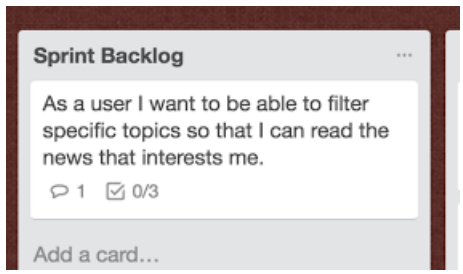


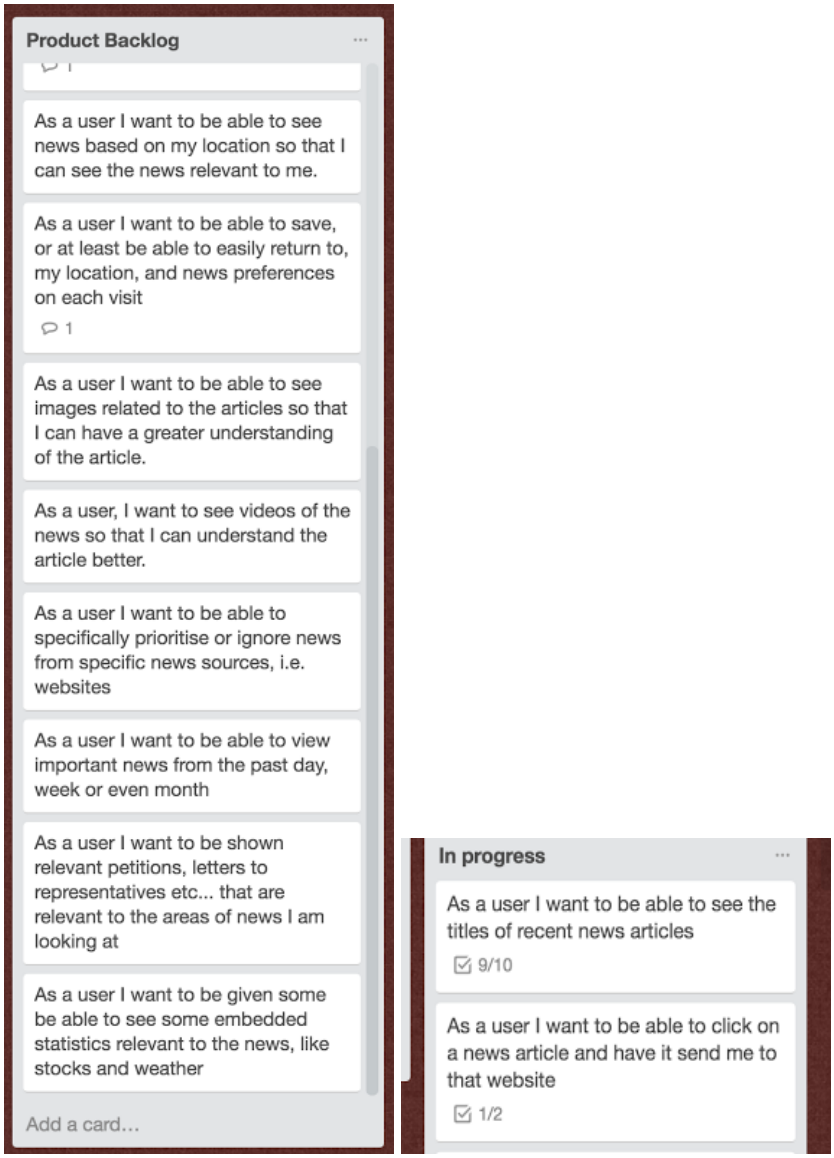


And then at the end of the first sprint:


Boards
Trello
Jack Horsburgh
SCRUM News Aggregator
SCRUM-practical
Team Visible
Show Menu
Initial Plans
Google Doc of initial plans
Github Repo
Add a card...
Completed
As a user I want to be able to see news articles from different sites in one place so that I don't have to go to all the different sites.
2/2
Add a card...
Product Backlog
As a user I want to be able to see news based on my location so that I can see the news relevant to me.
As a user I want to be able to save, or at least be able to easily return to, my location, and news preferences on each visit
As a user I want to be able to see images related to the articles so that I can have a greater understanding of the article.
As a user, I want to see videos of the news so that I can understand the article better.
As a user I want to be able to specifically prioritise or ignore news from specific news sources, i.e. websites
As a user I want to be able to view important news from the past day, week or even month
As a user I want to be shown relevant petitions, letters to representatives etc... that are relevant to the areas of news I am looking at
As a user I want to be given some be able to see some embedded statistics relevant to the news, like stocks and weather
Add a card...
Sprint Backlog
As a user I want to be able to filter specific topics so that I can read the news that interests me.
0/3
Add a card...
In progress
As a user I want to be able to see the titles of recent news articles
9/10
As a user I want to be able to click on a news article and have it send me to that website
1/2
As a user I want to have an responsive UI that works on my desktop and mobile devices.
Add a card...
Add a list...


Completed
As a user I want to be able to see news articles from different sites in one place so that I don't have to go to all the different sites.
2/2
Add a card...


Sprint Backlog
As a user I want to be able to filter specific topics so that I can read the news that interests me.
1
0/3
Add a card...



Although we were only able to move one full user story into the completed section, we had 11 tasks completed whose cards were in the “in progress” section, all of which had one task left before the whole story was complete. Thus we were able to move these stories into the “complete” section quickly after starting the second sprint, by swiftly finishing off those final tasks.

Sprint 1 Stand-up Summary

At our first stand-up we discussed our choice of server and tools to implement server client communication, including discussions of Apache Tomcat and Glassfish. There was further discussion and clarification of our choices for the basic architecture and design. Also we had a short discussion that involved feedback and potential improvements with regards to the trello page.

At our second stand-up we briefly discussed particular RSS feeds. We had important discussion of our methods and time allocations for setting up the Glassfish servers across the lab machines and laptops of the group. We also had some discussion of our passing methods.

At our third stand-up we discussed our basic page layout and design, use of tables for example, as we had completed some of the initial stages of engineering the frontend. We also discussed the specifics and details that had arisen as issues with regards to the respective protocols and roles between, and of, the frontend and the backend. Also there was more discussion on developments with regards to the parser.

Sprint 1 Review

We made a conscious effort to have a working product made for the end of the first sprint so that this could be demo'd to the Product Owner. At the Sprint Review we displayed the prototype of our UI, as well explaining how the RSS feeds are parsed and stored. The Product Owner gave their feedback, saying that the initial prototype was good, but that we should now focus on categorising the feeds, as well as getting more useful information onto the web page.

Sprint 1 Retrospective

The Sprint Retrospective meetings allowed us to reflect on the sprint we had just finished, highlighting what aspects of it we thought went well, what went wrong, and what could be improved for the next sprint. For our first retrospective meeting we recorded the audio of our conversation (included in our upload), but have also outlined the points raised during it below.

What went well?

- The process of Scrum itself allowed us to be more focused on specific goals, through user stories etc., making it easier to work on things bit by bit.

- There was a good level of productivity, and we managed to complete a lot of the user stories from the Sprint Backlog.

- Creating our Scrum Board on Trello, where everyone always had access to it, allowed us to see and track our progress as it happened.

- There was a good division of people working on the areas that suited them best, i.e. back-end server or front-end UI.

- Splitting up the tasks amongst the group meant that we were able to start of working on all aspects of the project without having to wait for another to be finished, e.g. building the UI while the server and code for the parsing of RSS feeds was still being worked on, as its basic functionality wasn't heavily dependent on them.

What went wrong?

- We didn't quite complete our sprint backlog: there was still one task relating to categories which we had to put back into the product backlog.

- We could have managed the Trello Scrum Board better beyond the first few days, ensuring we actually recorded our progress when it happened.

- Stand-up meetings could have been more consistent time-wise, i.e. they could have taken place at the same time each day.

- It was difficult to work around the various different deadlines everyone had; the Scrum process might have felt more suitable if we were solely focussed on the Scrum-based project.

- Our time estimates for some of the user stories were a little ambitious; we could have taken more time to actually think about which areas were going to hold us up the most and allocated them more time compared to the rest.

- More specifically in the back-end development, there were quite a few areas that couldn't be worked on until another was completed, meaning the project didn't advance as quickly as it could have done.

- Ideally, we wouldn't have had to change which server we were using to host our site after we had already started development.

- Some of the server-based problems were hard to delegate amongst the group, as most of the group had very little/no prior experience working with servers.

Could anything be improved?

- There was quite a bit of inertia towards the start of the project, largely because of the server issues, but there is now some momentum, hopefully making it easier to continue now there is a basis to work from.

- We could maybe assign tasks to people better based on their ability and experience with a specific area, i.e. servers, Java, HTML etc., as well as what they most preferred to work on.

- Ensure that the Trello board is being kept up-to-date by ticking items of the checklist of a user story, moving a user story card once everything for it is completed etc.

Scrum Master Report

Scrum Master: Charlie Blake

As Scrum Master for the first sprint, I saw it as my role to facilitate the team's pursuit of our sprint goals, and to remove any impediments along the way. This was a challenging task for a group of people with little experience of Scrum, and with uncertain expectations regarding what a sprint might include and how long tasks would take to complete.

The first stage of this was getting our Scrum-related tools setup. As a team, we started a Trello page to log all of our progress which we felt was a better alternative to using a whiteboard, as often we would be working with each other remotely. I also took responsibility for our Sprint 1 Planning Meeting, at which I managed the process of moving stories from the Product Backlog into the Sprint Backlog, which we did as a team. We also broke these stories down into tasks which could be completed in a relatively short amount of time, and we also made time estimates for each task as a group.

In a more general sense, I also communicated with everyone over Facebook to make sure at different times everybody was up to date and understood what was required of them. Generally, we all felt very pleased with how we had taken to the Scrum framework and immediately saw the dividends that using this method could pay in terms of managing our work. If there was one area which I could have managed better it would have been the handling of obstructions to our workflow: the initial setup phase of the project involved a few tasks had to be completed before others could begin. Had I foreseen this more clearly, perhaps I could have taken steps to mitigate this effect. As a result we had a few user stories whose tasks were not quite complete by the end of the first sprint, and were only completed at the very beginning of Sprint 2.

Product Owner Report

Product Owner: Taylor Alexander

As the Product Owner, I was responsible for ensuring that the product met the standards of the specification. I did this by setting a priority for this sprint - the product must collect news articles and display the links in one place. I felt that by prioritising this, the news aggregator would at least have some basic functionality by the end of the first sprint. At the end of the sprint I gave my feedback on the product that was made. I felt that we had achieved what I had set as a priority and so we could then look at categorising feeds for extra functionality.

--- Sprint 2 ---

Sprint 2 Planning Meeting

In the next sprint we took a different approach to the planning meeting. As we already had a basic news aggregator, the stories placed in the backlog focused more on the user interface and adding extra features. As we had gained momentum from the first sprint and understood how the Scrum process worked, we were more confident in adding extra user stories to the sprint backlog. The stories chosen were filtering categories and retrieving images for the articles, as well as implementing live updates.

The way the stories were broken down into tasks was done in the same way, but ‘planning poker’ was used as a way of determining the time estimates. This allowed for a better estimate for the length of the tasks due to the experience that each team member had. From the previous sprint retro it can be seen that we did not always know what tasks each member was completing. To combat this, we made use of the Trello to add members to a card, which greatly helped with this matter. From this meeting we had a clearer goal of what we were to achieve in the second sprint.

Sprint 2 Timeline

17th Oct: Sprint start

Formatted UI to look like a very basic news site **(task complete)**

Refactored backend code to improve structure **(task complete)**

18th Oct: Stand-up meeting (on Facebook)

Formalised our method of reading in RSS feed URLs into an explicit class **(task complete)**

19th Oct: Changed backend code to hold last 5 items from each feed in queue **(task complete)**

Wrote code to turn item queues into JSON **(task complete)**

Sent out update JSON to all connected clients when new item was encountered in the RSS feeds **(task complete)**

The RSS feeds were categorised into separate data structures **(task complete)**

20th Oct: Added UI buttons to let users select categories **(task complete)**

Wrote javascript code to implement actual filtering **(task complete)**

21st Oct: Wrote methods to send all existing feed info to new connecting clients **(task complete)**

Experimented with how often we could/should poll RSS feeds **(task complete)**

Added weather widget **(task complete)**

22nd Oct: Stand-up meeting (in person)

Debugged and fixed quite a few small errors with our server-side code.

22nd Oct: Added image links from the RSS XML to the JSON sent out by the server **(task complete)**

.

Added small improvements to the UI just to give it a “polished” feel.

Parsed the image on the client side and rendered it on the page **(task complete)**

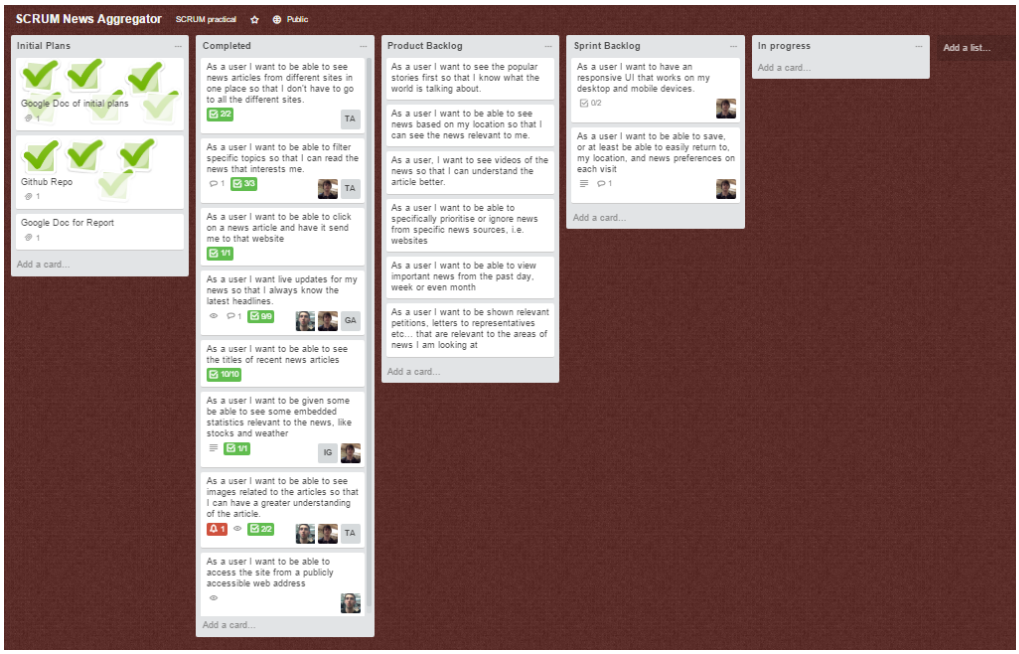
23rd Oct: Stand-up meeting (in person)

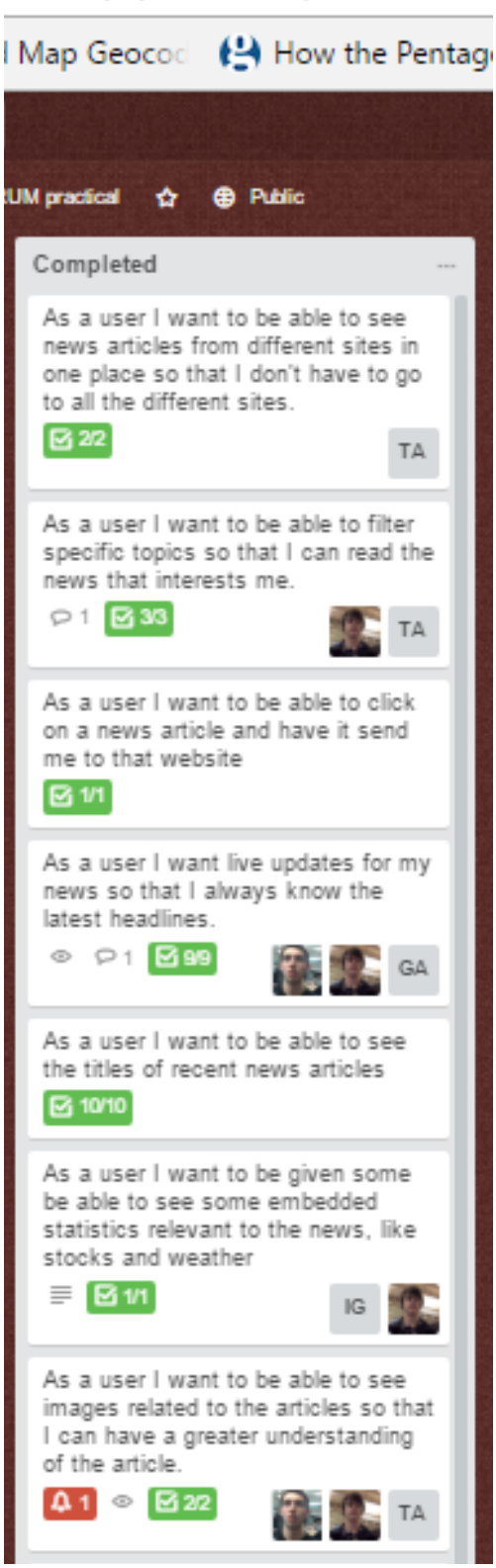
Brushed up the webpage further by optimising code which slowed the page down and adding small css changes, plus a loading icon.

Sprint end

Sprint 2 Trello Screenshots

The following screenshots are from the end of our second sprint:




Map Geococ
How the Pentag
UM practical
Public
Completed
As a user I want to be able to see news articles from different sites in one place so that I don't have to go to all the different sites.
2/2
TA
As a user I want to be able to filter specific topics so that I can read the news that interests me.
1
3/3
TA
As a user I want to be able to click on a news article and have it send me to that website
1/1
As a user I want live updates for my news so that I always know the latest headlines.
1
9/9
GA
As a user I want to be able to see the titles of recent news articles
10/10
As a user I want to be given some be able to see some embedded statistics relevant to the news, like stocks and weather
1/1
IG
As a user I want to be able to see images related to the articles so that I can have a greater understanding of the article.
1
2/2
TA


gon pu
How immigration cam
Product Backlog
As a user I want to see the popular stories first so that I know what the world is talking about.
As a user I want to be able to see news based on my location so that I can see the news relevant to me.
As a user, I want to see videos of the news so that I can understand the article better.
As a user I want to be able to specifically prioritise or ignore news from specific news sources, i.e. websites
As a user I want to be able to view important news from the past day, week or even month

Sprint 2 Stand-up Summary

In our first stand-up for the second sprint we discussed backend architecture improvements and things we should prioritise with regards to the data collection from the RSS feeds. As we were using JSON at this point we had a discussion of the data types and the structure of information that would have to be used. There was also a brief discussion and dismissal of alternatives to JSON.

At our second stand-up for the second sprint we discussed some of the more difficult bugs that we were encountering with the software and tried to find potential reasons for, and solutions to, these issues. We had a discussion of the final webpage design, and a brief discussion of image selection.

Sprint 2 Review

Our second sprint review session involved us all grouping together in the Jack Cole lab and putting our final webpage on one of the large screens. We went through each feature of the UI and assessed its functionality and how our general application compared to our initial expectations.

We again tried to hold this meeting in such a way that it felt like the team were presenting to our Product Manager, who had no knowledge of the internal workings of the app. In a way this was also a rehearsal for our project presentation. Although we still had a few items in the product backlog by the end of the project, our final product was definitely a viable and quite effective application.

Sprint 2 Retrospective

As with the first Sprint Retrospective meeting, our second allowed us to reflect on the sprint we had just completed, not only looking at what went well, but whether we were able to improve upon any of the points raised in the first retrospective meeting. It also allowed us to reflect on the project as a whole, now that we had completed our final sprint. We did not record this meeting, but the topics discussed are given below.

What went well? What did we improve?

- Our stand-up meetings were more focussed, with a better division of work amongst the team.

- We were able to use Trello more effectively, adding group-members to specific user story cards, making it easy to see who was working on what, and interacting with and updating the board more as user stories were completed.

- More work was done generally, helped by the fact that it was Independent Learning Week and there were fewer deadlines for other modules to think about.

- Already having the basic architecture set up made it easier to pick up the momentum from the previous sprint and continue developing the product.

- There was a good, modular division between elements of the architecture, meaning components could be modified and upgraded independently.

- Our time estimates for task completion were improved and more realistic.

What went wrong?

- We were sometimes too ambitious with the Sprint Backlog, both with the number of user stories in it, as well as the complexity of the individual stories.

- More time could have been spent working on the product in the same physical location, allowing us to carry out stand-up meetings more easily, help each other when a problem was encountered, make decisions as a group etc.

- On occasion, the distinction between what was finished code and what was still a draft version wasn't very clear.

- Debugging and testing methods could have been more robust, especially for the back-end parsing of RSS feeds etc.

Could anything be improved? General reflections on the project?

- The roles of Scrum Master and Product Owner could have been more visible during the sprints themselves.

- Eclipse was not the best choice of IDE for Java EE purposes; we could and should have spent longer researching other options.

- Creating an architecture diagram at the start of development would have helped to visualise the desired infrastructure.

Scrum Master Report

Scrum Master: Jack Horsburgh

As Scrum Master I was responsible for ensuring the team had no major obstacles to their progress. I facilitated several discussions on our facebook group chat about fixing problems with our server architecture which were preventing the team from working. I organised our meetings for the week (although most of these ended up being on facebook) and was actively managing the trello board. I also managed the documentation of the second sprint by creating report templates and made a start to the group report.

Product Owner Report

Product Owner: George Alexander

I was the product owner for the second sprint, in the second sprint we had quite a variety of goals in the sprint backlog, including live updates, embedded weather, images and so on. This was in contrast with the first sprint where basic functionality was a very focused goal. As such feedback was closely integrated with the development (especially since this was our final sprint), in order to ensure the different parts of the project cohered to our combined vision. In the second sprint we succeeded in adding a good number of features and finished with a very polished final product with features that worked and complemented each other.

Summary

To conclude, our team fulfilled the specification by creating a working news aggregator, which uses a large number of RSS feeds to create an “ultra-live” service. These feeds can also be categorised, allowing the user to only see the news they want.

We worked well as a team to complete the tasks set while using the Scrum agile process. We used Trello to effectively display our Scrum board while also communicating through Facebook when stand-up meetings were not possible. We feel that we may end up using some aspects of Scrum in our year-long projects as it allows us to organise the team a lot better.

Overall we are very happy with the product that we have created. Many of the team actually use the news aggregator regularly as it has a very easy to use interface. We

also feel that, if developed further, *Hot Off The Press* has the potential to become a commercial product that could be widely used.