2019

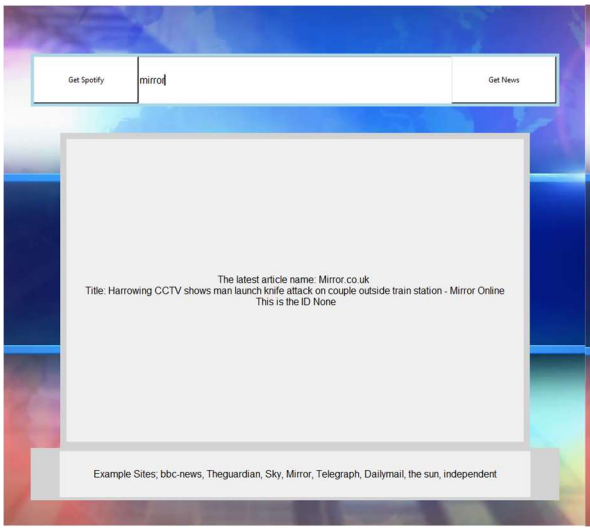
Software Engineering: SCRUM Report

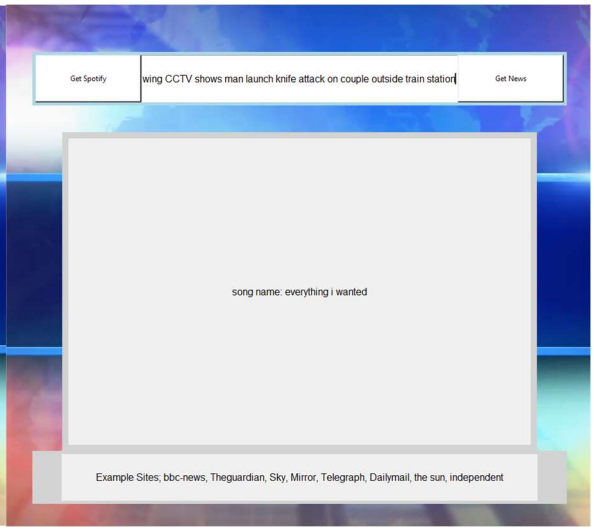
GROUP D4: GABRIELLA DI GREGORIO, ROWAN REED, ALEX WOOD, JONNY WRIGHT
BY GABRIELLA DI GREGORIO 15624188

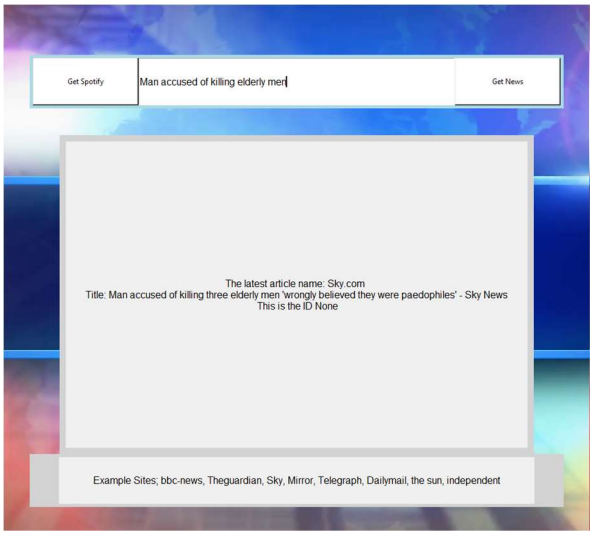
Contents

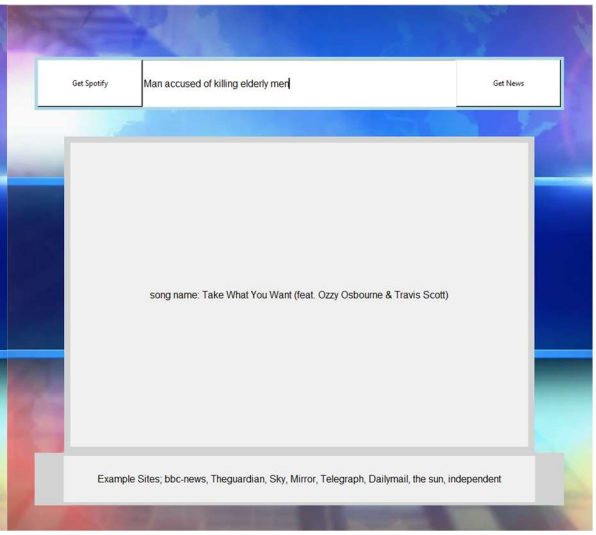
The Artefact

We chose the project to create an “application to generate track recommendations from Spotify based that day’s news”.









Our program reads in the articles in the current news from which the user can search the GUI for a news source to generate a headline from it, such as BBC News. Then, the headline or part of the headline can then be used to generate a random song recommendation from Spotify based on terms within the article.

Development Log

Product Backlogs:

Initial Product Backlog:

ID	Item/Story	Type	Story Points	Business Value	Priority
0	As a user I would like to open the application and get Spotify song recommendations related to today's news so I can listen to new music.	Feature	9	9	9
1	As a developer I want access to API's to be able to get data.	Feature	2	1	8
2	As a user I would like to have an intuitive user interface for checking my recommendations so the application is easy to use.	Feature	8	8	7
4	As a user, I want to choose which news sources will generate my recommendation so I get a variety of songs.	Feature	3	5	6
5	As a developer I would like to be able to test the program when new versions are released to make sure there are no bugs in the application.	Knowledge Acquisition	7	4	5
6	As a user I want the application to be quick and responsive to get recommendations quickly to avoid user frustrations.	Feature	2	8	3
7	As a user, I want to pick between different regions and categories so that my choices are relevant to wherever I might be interested in.	Feature	3	8	2
8	As a fan of rock music, I want to get more personalised recommendations based on my music taste	Feature	2	7	2

Item/Story: Requirements or features often in the form of user stories from stakeholders such as customers. Items of functionality that will have tangible value to the user and customer.

Type: Feature, Change, Defect, Technical Improvement, or Knowledge Acquisition.

Story Points: An estimation of how long it would take to complete the task or the size of it. 1-9

Business Value: How important it is to the client/customer to achieve in order to meet the requirements or to the completeness of the system. 1-9

Priority: The order in which the tasks will be developed, likely to be similar to the Business Value. 1-9

A Product Backlog is an ordered list of everything that is known to be needed in the product. It is the single source of requirements for any changes to be made to the product. (Scrum.org, n.d.) We decided to create a Product Backlog in order to document all the requirements for the task as well as roughly planning the order in which to carry out subtasks and the relative timeframe. To ensure that we covered a wide range of perspectives, understood the task at hand and did not miss out any details, we decided that each member would individually create their own draft of a Product Backlog. In the next meeting, the Product Owner (the role played by myself at the time), discussed each members' Product Backlog ideas and used each one to collate them into a single Initial Product Backlog that would be our reference point for the first Sprint. For example, the first item 'as a user I would like to open the application and get Spotify song recommendations related to today's news so I can listen to new music' is the core point of the task. Since nothing else can be done without this being completed, it scored the highest business value of 9 and since it is the largest task and would most likely take the longest time, it scored the highest Story Points of 9, these factors therefore made this task the highest priority.

A well-prioritized agile backlog not only makes release and iteration planning easier, it also broadcasts all the things your team intends to spend time on—including internal work that the customer will never notice. This helps set expectations with stakeholders and other teams, especially when they bring additional work, and makes engineering time a fixed asset. (Radigan, n.d.) We recognised the Product Backlog as an important part of Scrum and it aided in ensuring that all team members understood the task and the subtasks that needed to be carried out, especially since it could be shared online so that everyone had access anywhere at all times. However, whilst working with a list is helpful when the focus is on adding functionality, on writing and prioritising user stories, creating a great product requires more than just user stories. The user journeys, the visual design, and the non-functional properties have to be considered too but they unfortunately don't fit into a

list. As a consequence, agile teams may either forget about capturing the user experience, or they keep the UX artefacts separately, for instance, in a project management tool. While the former can result in a product with a poor user experience, the latter isn't great either: information that belongs together is stored separately. This makes it more difficult to keep the various artefacts in sync, and it can cause inconsistencies and errors. (Pichler, 2013) Throughout the process, we realised the difficulty of coming up with a definitive list of requirements before the development process had begun, of course, this simply highlights the need and importance of a Groomed Product Backlog.

Groomed Product Backlog:

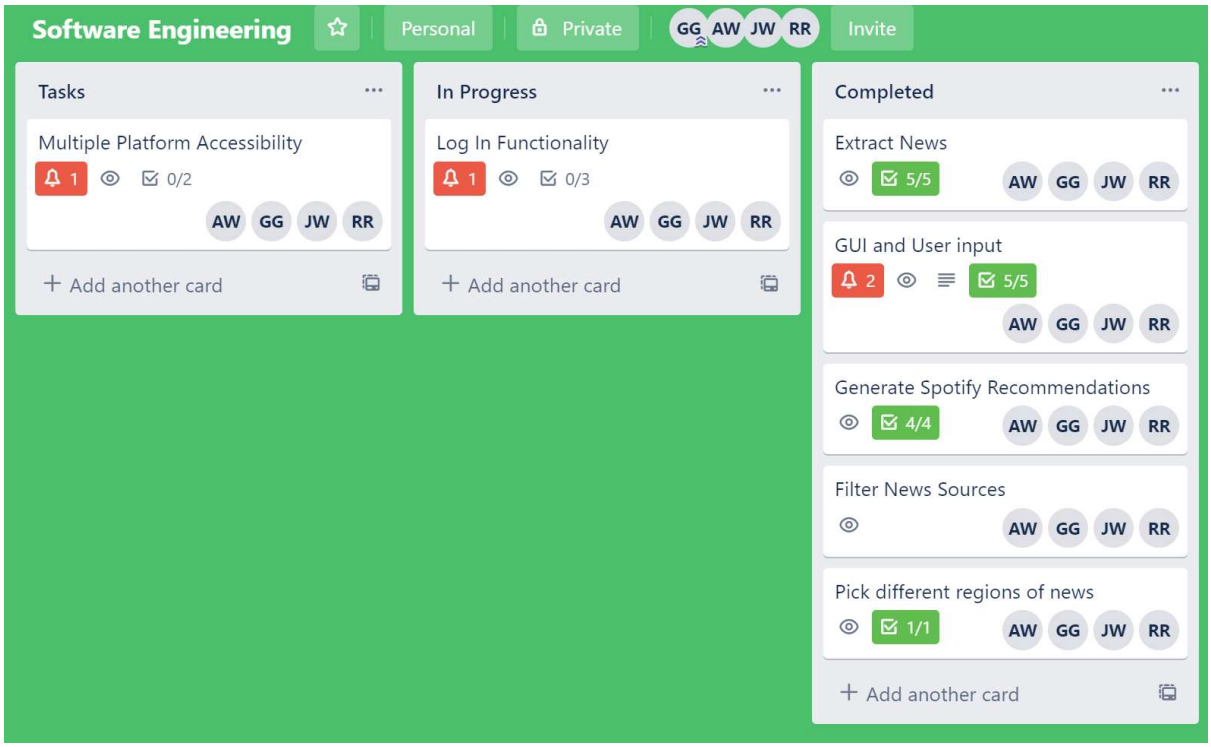
Backlog grooming, also referred to as backlog refinement or story time, is a recurring event for agile product development teams. The primary purpose of a backlog grooming session is to ensure the next few sprints worth of user stories in the product backlog are prepared for sprint planning. Regular backlog grooming sessions also help ensure the right stories are prioritised (Productplan.com, n.d.), irrelevant user stories are removed, and new user stories are added in response to newly discovered needs.

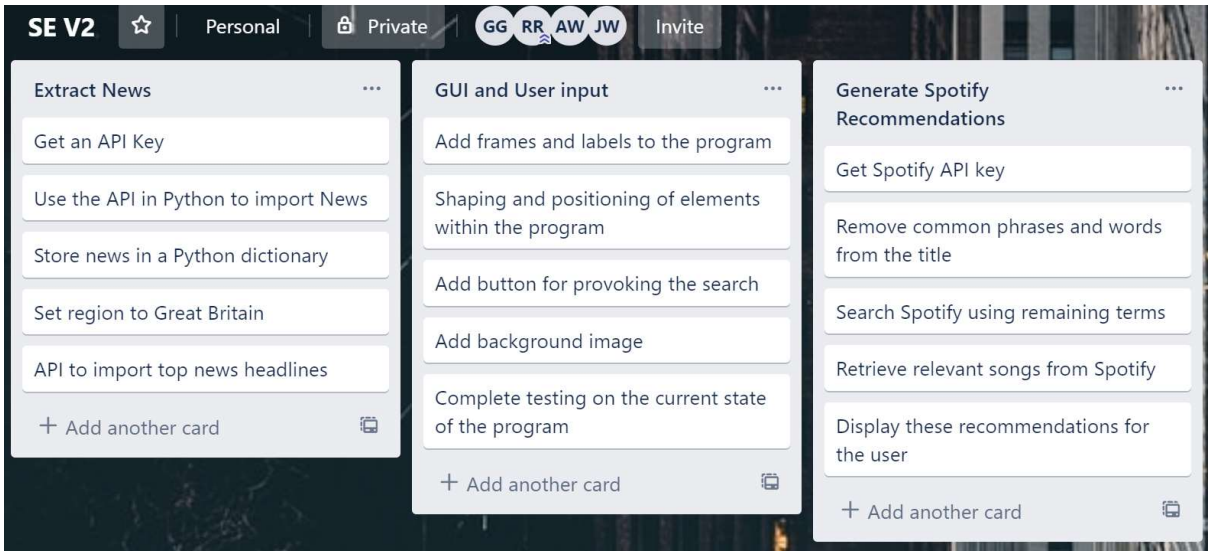
ID	Item/Story	Type	Story Points	Business Value	Priority
0	As a user I would like to open the application and get Spotify song recommendations related to today's news so I can listen to new music.	Feature	9	9	9
1	As a developer, I want access to API's to be able to get data.	Feature	2	1	8
2	As a user I would like to have an intuitive user interface for checking my recommendations so the application is easy to use.	Feature	8	8	7
3	As a developer, I had issue getting terms from the API's.	Defect	1	7	7
4	As a user, I want to choose which news sources will generate my recommendation so I get a variety of songs.	Feature	3	5	6
5	As a developer I would like to be able to test the program when new versions are released to make sure there are no bugs in the application.	Knowledge Acquisition	7	4	5
6	As a user I want the application to be quick and responsive to get recommendations quickly to avoid user frustrations.	Feature	2	8	3
7	As a user I would like to be able to sign in with my Spotify account so songs can be saved as a playlist to my account.	Technical Improvement	2	7	1
8	As a user I want application to work on multiple platforms so I can get daily recommendations anywhere.	Technical Improvement	9	6	1
9	As a user, I want to be able to save recommendations so I can see them again even when the news changes.	Technical Improvement	3	6	1
10	As a user, I want to be able to log in to the application for my personalised recommendations.	Technical Improvement	3	6	1

After each Sprint, at least one team member groomed the Product Backlog to ensure its relevance, and to prepare for the next sprint. As can be seen by comparing the Initial Product Backlog and the Groomed Product Backlog, several changes were made and whilst it became longer after grooming, other stories were removed that were in the initial version. For example, item 8 in the initial product backlog, 'as a fan of rock music, I want to get more personalised recommendations based on my music taste' was removed during a grooming session as it was decided that this was beyond the scope of the task. However, several additions such as 'as a user I would like to be able to sign in with my Spotify account so songs can be saved as a playlist to my account' were made during grooming sessions as new requirements were discovered that had perhaps not been considered before the process begun. These additions were mostly made as 'technical improvements' as they are not key features of the product and it could still be considered complete without them.

Sprints:

Sprint Planning:





Sprint planning is an event in the Scrum framework where the team determines the product backlog items they will work on during a sprint and discusses their initial plan for completing those product backlog items. Teams may find it helpful to establish a sprint goal and use that as the basis by which they determine which product backlog items they work on during that sprint. (Agile Alliance, n.d.) Once we had created our initial product backlog, we were able to begin Sprint planning, and since we knew which tasks to prioritise, we knew which task to tackle in our first Sprint. All that was left to do was to break the Story Point(s) being tackled down into manageable subtasks. We initially did these as checklists in Trello as shown by the first board, but since it was not easy to see all the tasks and subtasks at once, an additional board was created using a different layout which broke each task down into subtasks and all subtasks could be viewed at once. We found that the second board was useful at a glance, but our initial sprint board was far useful for management since tasks could be

ticked off and dragged between 'in process' and 'completed'. This was extremely useful for agile development since completed tasks may return to 'in progress' in order to carry out further testing or to make technical improvements. We intended to use the first board for our sprint logs, but since Trello does not provide an easy or clear way to represent predicted and actual times taken for each subtask, it was necessary to also create tables for each sprint displaying this key information.

The main benefit of sprint planning is that it allows a team to start a new sprint with a shared understanding of what they will work on for that sprint, as well as an initial plan for how they approach that work. (Agile Alliance, n.d.) We found our sprint plans to be essential in ensuring that all team members understood the task, what they needed to do, and what is yet to do. This was particularly helpful since due to the short timeframe, development may have occasionally taken place without all team members present and some work may have been carried out individually. Since our sprint plans were shared online and were updated live, all team members could access the plans anywhere anytime in order to check our progress or update the rest of the team regarding any changes or progress that had been made. However, sprint planning can become ineffective when the team does not have a properly refined product backlog from which to draw product backlog items. If this situation arises, the team will inevitably have to spend time during sprint planning developing a better understanding of the product backlog items, which may include splitting stories, and estimating. (Agile Alliance, n.d.) This can be an inefficient use of time, especially when coincided with the time spent creating sprint plans in various formats to ensure understandability. This issue was largely avoided due to regular product backlog grooming sessions, before new sprint plans were made.

Sprint 1:

Sprint Log:

For each sprint, we thought about the main user story from the Product Backlog being tackled, and the subtasks identified in the sprint plans. However, reflection after each Sprint revealed that multiple user stories related to a single sprint, this is likely to be due to the agile nature of the project since tasks are not always fully completed before another begins, so this was documented in the sprint retrospectives. Our sprint tables show an estimate of how long we predicted each subtask would take, and how long we actually spent carrying out these subtasks across the development period. Whilst our tables show the total time spent on each subtask throughout development, upon reflection, it would have been useful to show the total estimated time and the total time spent for each task/user story, as it takes a moment of calculation to know whether we were behind or ahead of schedule. Despite this, they were useful overall in keeping track of our schedule and relative success, however it did sometimes prove to be difficult in accurately remembering/recording how long each subtask took, especially when they were carried out consecutively.

User Story - As a user I would like to open the application and get Spotify song recommendations related to today's news so I can listen to new music.

Sprint 1 – Extract News

Tasks	Estimated time to complete	W1 Hours	W2 Hours	W3 Hours	W4 Hours	Actual time to complete
Get an API Key	30 minutes	10 minutes	0	0	0	10 minutes
Use the API in Python to import news	2 hours	2 hours	0	0	0	2 hours
Store news in a Python dictionary	1 hours	1 hour	0	0	0	1 hour
Set region to Great Britain	30 minutes	30 minutes	0	0	0	30 minutes
Filter by News Source	30 minutes	10 minutes				10 minutes
API to import top news headlines	1 hour	1 hour	0	0	0	1 hour

Sprint Retrospective/Meeting Log:

Goal: Extract News

Relevant User Stories:

As a user I would like to open the application and get Spotify song recommendations related to today's news so I can listen to new music.

As a developer I want access to API's to be able to get data.

As a user, I want to choose which news sources will generate my recommendations, so I get a variety of songs.

As a developer I would like to be able to test the program when new versions are released to make sure there are no bugs in the application.

What each member did:

Gabby: Planned the task in the Product Backlog and assigned it the highest Story Points and therefore the highest Priority

Rowan: Created the Sprint Backlog (and the template for all future Sprints) and broke the task down into subtasks

Alex: Obtained an API Key and began programming

Jonny: Groomed the Product Backlog

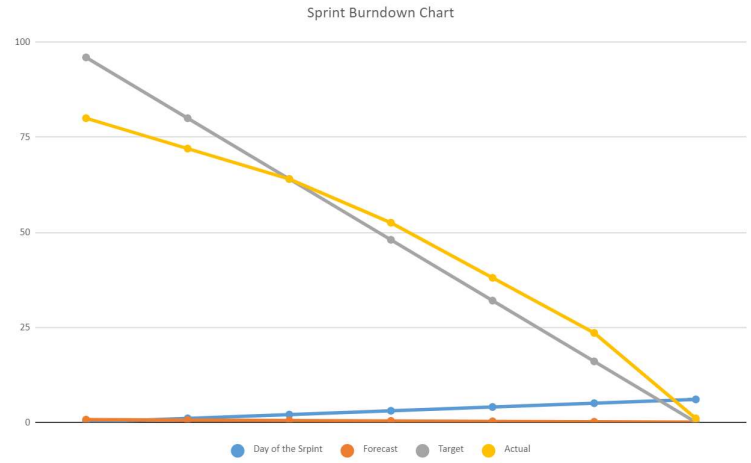
What went well?

Obtained API Key easily and quickly. Was able to extract news from different sources and remained on schedule. Testing did not show any errors.

What could be improved?

Problems were encountered relating to permissions on Lab PCs so time could have been saved by members bringing their own devices.

Sprint Burndown Chart:



Sprint 2:

Sprint Log:

User Story - As a user, I would like to have an intuitive user interface for checking my recommendations so the application is easy to use.

Sprint 2 – GUI and User input

Tasks	Estimated time to complete	W1 Hours	W2 Hours	W3 Hours	W4 Hours	W5 Hours	Actual time to complete
Add frames and labels to the program	3 hours	0	2 hours	0	1 hours	1 hours	4 hours
Shaping and positioning of the elements in the program	4 hours	0	2 hours	0	1 hour	0	3 hours
Add button for provoking the search	1 hour	0	30 minutes	0	0	0	30 minutes
Add background image	30 minutes	0	30 minutes	0	0	0	30 minutes
Complete testing on the current state of the program	4 hours	0	2 hours	2 hours	2 hours	2 hours	6 hours

Sprint Retrospective/Meeting Log:

Goal: Create GUI and User Input

Relevant User Stories:

As a user I would like to have an intuitive user interface for checking my recommendations so the application is easy to use.

As a developer I would like to be able to test the program when new versions are released to make sure there are no bugs in the application.

What each member did:

Gabby: Wrote up the Sprint Retrospective (and a template for all Sprints) and Updated the Sprint Plan in Trello

Rowan: Pair Programmed with Jonny to create the Interface

Alex: Groomed the Product Backlog and filled out the Sprint Backlog

Jonny: Designed and Implemented the interface

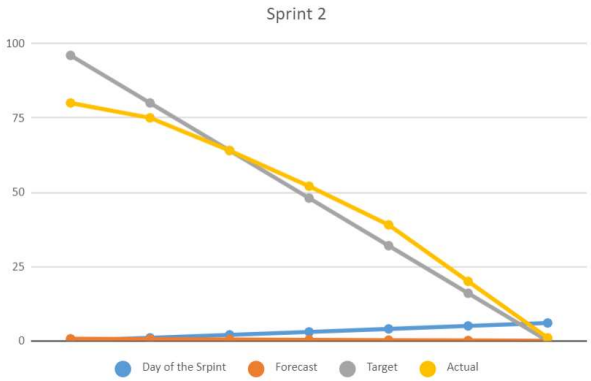
What went well?

No Issues arose in implementation or testing

What could be improved?

Took longer than anticipated to implement so meant that some Technical Improvements were not able to be made in the timeframe, therefore the Product Backlog required significantly updating. Could not finish the GUI since it was started before the Spotify API was implemented so had to be returned to at a later date, it may have been better to do this in a different order.

Sprint Burndown Chart:



Sprint 3:

Sprint Log:

User Story - As a user I would like to open the application and get Spotify song recommendations related to today's news so I can listen to new music.

Sprint 3 - Get songs from spotify

Tasks	Estimated time to complete	W1 Hours	W2 Hours	W3 Hours	W4 Hours	Actual time to complete
Get Spotify API key	30 minutes	0	0	10 minutes	0	10 minutes
Remove common phrases and words from the title	2 hours	0	1 hour	2 hours	0	3 hours
Merge news and spotify branches	2 hours	0	0	2 hours	0	2 hours
Search Spotify using remaining terms and phrases	2 hours	0	0	3.5 hours	0	3.5 hours
Retrieve relevant songs from Spotify	1 hours	0	0	1 hour	0	1 hour
Display these recommendations for the user	30 minutes	0	0	30 minutes	0	30 minutes

Sprint Retrospective/Meeting Log:

Goal: Get songs from Spotify

Relevant User Stories:

As a user I would like to open the application and get Spotify song recommendations related to today's news so I can listen to new music.

As a developer I want access to API's to be able to get data.

As a developer I would like to be able to test the program when new versions are released to make sure there are no bugs in the application.

What each member did:

Gabby: Participated in Pair Programming with Alex to implement the API

Rowan: Updated the Sprint Plan in Trello and created the Sprint Backlog

Alex: Obtained an API Key and began programming

Jonny: Groomed the Product Backlog

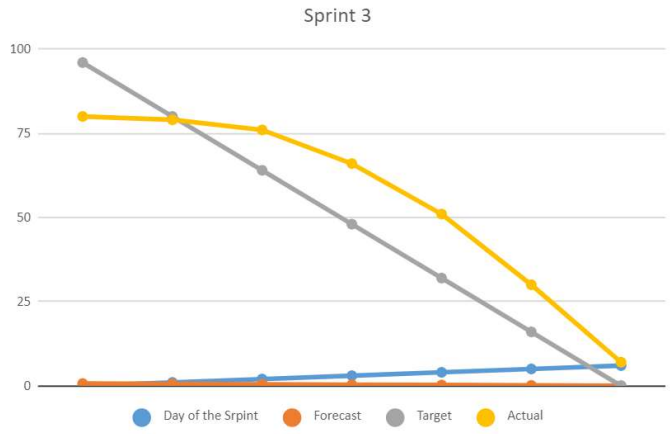
What went well?

Obtained the API Key and found documentation to make the implementation process smoother.

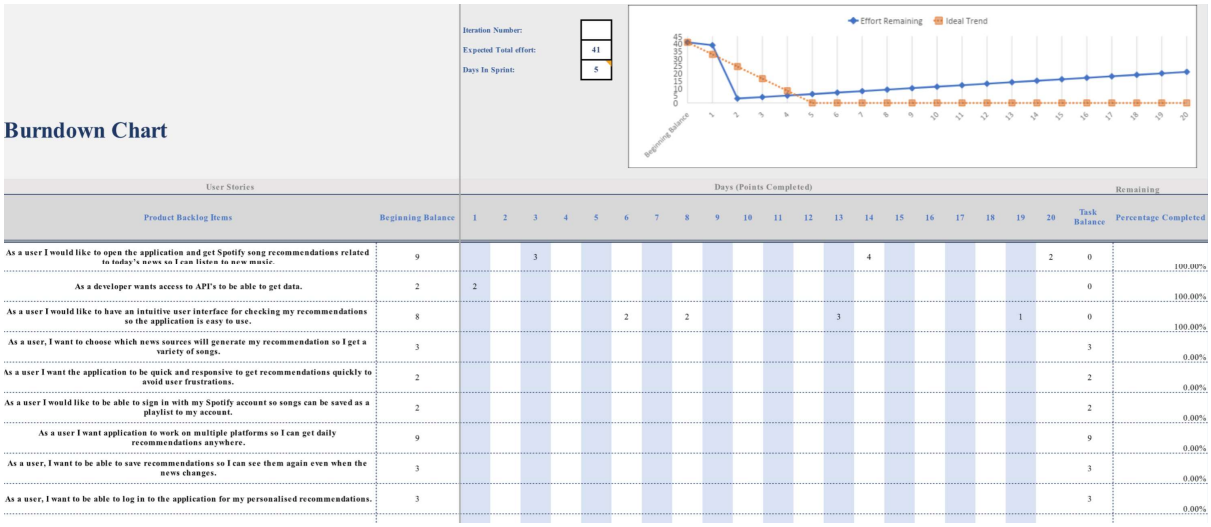
What could be improved?

Some of the documentation was outdated so had to take time to search elsewhere, it may have been useful to do this in advance. A Spotify account was also necessary to proceed so took some time to get a member logged in and would also 'sacrifice' their recommendations, could have made a new dummy account instead.

Sprint Burndown Chart:



Product Burndown Chart:



Burndown Chart

Iteration Number:

Expected Total effort: 41

Days In Sprint: 5

User Stories		Days (Points Completed)																		Remaining	
Product Backlog Items	Beginning Balance	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Task Balance	Percentage Completed
As a user I would like to open the application and get Spotify song recommendations related to today's news so I can listen to new music.	9			3											4						2	0	100.00%
As a developer wants access to API's to be able to get data.	2	2																				0	100.00%
As a user I would like to have an intuitive user interface for checking my recommendations so the application is easy to use.	8						2		2					3						1		0	100.00%
As a user, I want to choose which news sources will generate my recommendation so I get a variety of songs.	3																					3	0.00%
As a user I want the application to be quick and responsive to get recommendations quickly to avoid user frustrations.	2																					2	0.00%
As a user I would like to be able to sign in with my Spotify account so songs can be saved as a playlist to my account.	2																					2	0.00%
As a user I want application to work on multiple platforms so I can get daily recommendations anywhere.	9																					9	0.00%
As a user, I want to be able to save recommendations so I can see them again even when the news changes.	3																					3	0.00%
As a user, I want to be able to log in to the application for my personalised recommendations.	3																					3	0.00%

A burndown chart is a graphic representation of how quickly the team is working through user stories and shows the total effort against the amount of work for each iteration. The obvious benefit of a burndown chart is that it provides an updated status report on the progress of the project. Having a visual representation of this most important data keeps everyone on the same page. (Blackburn, 2019) Whilst its relevance to SCRUM development is clear, the team did unfortunately not reap the benefits of this tool for this task since they were created as part of reflection after the product had been completed, rather than updated iteratively. Nevertheless, it is a good way of representing how closely our work matched the predicted, or ideal trend. The chart shows that after staying on or slightly ahead of target at the very beginning of the project, effort/speed dropped early on meaning that extra work was necessary to ensure tasks were completed in time. However, the burndown chart doesn't reveal everything, for example, it only shows the number of story points that have been completed. The burndown chart doesn't show any changes, such as, in the scope of work as measured by the total points in the backlog. Therefore, it can be hard to tell if changes in the burndown chart are because of the backlog items having been completed or because of an increase or decrease in story points. (Blackburn, 2019)

Pair Programming Logs:

Week	Time Spent	Task	Programmer A	Programmer B	Problems Encountered	How Problems were resolved
1	5 hours	Get program to get news from newsapi.org	Alex Wood		Needed an api key to do GET requests	Acquired api key.
2 & 4	14 hours	Create GUI	Jonny Wright	Rowan Reed	Program searches for news instead of setting source	Wrong parameter being assigned from GUI.
3	2 hours	Get program to search Spotify songs using an API	Alex Wood	Gabriella Di Gregorio	1. Could not proceed without a Spotify account 2. Some Syntax errors went unnoticed since Jupyter doesn't identify errors until the code is ran 3. Documentation for library was outdated 4. On search request get a 401 No token provided	1. A group member logged in so that we could use Spotify to develop the program 2. Some were spotted by the passenger programmer and others were resolved after being identified by being ran in Jupyter 3. Had to spend time searching elsewhere for guidance on how to use the library 4. Spotipy library documentation out of date. had to add Client credentials
3 & 4	2 hours	Finalise program by merging code for both News and Spotify	Alex Wood	Jonny Wright	1. News output was an incorrect type 2. No result songs found for result	1. Added code to change how the output of the news is stored 2. Split title into terms to search separately

Pair programming is where two developers work using only one machine. One programmer acts as the driver who codes while the other will serve as the observer who will check the code being written, proofread and spell check it, while also figuring out where to go next. (Stackify, 2017) For each sprint, we attempted to tackle them in pairs. The idea of pair programming is to resolve coding problems more quickly since 'two heads are better than one' – two people will be looking at the code to spot a problem or come up with a solution, and problems may be avoided if the passenger programmer spots an error being typed by the driver immediately. A study showed that it results in 15% fewer bugs than code written by solo programmers and two programmers working on the same program are only 15% slower than when these programmers work independently. (Stackify, 2017) However, overall, we felt that this was an inefficient use of time for the task at hand. Since the programming task itself was relatively short and simple and the Scrum process requires so much documentation, we often found that the passenger programmer had nothing to do and focused on documenting the process instead of keeping an eye on the code. Furthermore, meetings were often spent planning and documenting so code was sometimes carried out independently outside of meetings.

My Contribution:

Whilst we tried to change up the roles throughout the project, my most predominant role turned out to be the Scrum Master. The scrum master is responsible for ensuring a true scrum process over the course of a project. They hold together the scrum framework, facilitating the process for the organization, product owner and scrum team. (Malsam, 2018) I felt that I fit this role quite well due to my time management and leadership skills and my determination to succeed. Throughout the project, I sometimes felt a lack of motivation from my team so I decided to take it upon myself to make sure members were on-target and to encourage them to attend meetings. I also frequently proposed plans and ideas of what to tackle in each meeting to avoid any concerns that attending would not be a productive use of time. However, I had difficulty reminding my team of the importance of the scrum framework since members often got too focused on completing the code independently rather than working collaboratively and truly experiencing the framework. For example, whilst I had hoped most of the programming would be done in pairs in meeting sessions, it was frequently worked on by members independently outside of meetings. Whilst I expressed my frustrations and urged work to be done together and shared, I was frequently ignored. Since several people were working on the same project, it was important to keep the code version-controlled and shared, but despite my frequent reminders, the team often neglected this. Although I succeeded in ensuring the required work was done in time and was eventually documented (even though some documentation was made much later than it should have been), my role as a scrum master was unsuccessful in other ways. We may have made the right choices in best-using each members' skillsets, but there was only so much I could do to encourage others to work within the guidelines of the framework. This could have been improved or perhaps avoided by having recognised official roles at the start of the project rather than fitting a role to each member post-development and not swapping them so much. I also think some team members lacked a clear understanding of the scrum methodology, so it may have been beneficial to spend some time revising the framework and planning in advance how we would maintain the process before beginning development.

I also gained experience in the role of a Product Owner throughout the project. A Scrum Product Owner is responsible for maximizing the value of the product resulting from the work of the Development Team. (Scrum.org, n.d.) Part of the product owner's responsibilities is to have a vision of what they wish to build and convey that vision to the scrum team. This is key to successfully starting any agile software development project and the agile product owner does this in part through the product backlog. (Mountain Goat Software, n.d.) I ensured that I had a thorough understanding of what is required by carefully reading the task description and asking clients for further requirements. I also proposed the idea that each team member drafts a version of a Product Backlog to ensure all members were actively engaged and understanding the task, as well as sending reminders to complete this before the following meeting. I then led the meeting in which we discussed our ideas for the product backlog and used my draft as a template to collaborate all our ideas into an Initial Product Backlog which the team agreed upon. After this initial planning phase, I largely distributed the tasks and responsibilities of the Product Owner. Whilst I carried out some backlog grooming and oversaw all grooming sessions, I ensured that as many members of the team got a chance to carry out backlog grooming as possible. As well as this, I ensured that the correct backlog items were being focused on in each sprint and the tasks were broken down appropriately.

Unfortunately, due to my relatively lacking programming ability, I did not get as much experience as part of the development team as would have been ideal. Because of the short timeframe and relatively simple programming task, as the scrum master I decided it would be the most efficient use of time if the members with the most programming experience worked on the code more than

others. Whilst this was successful in getting both the program and scrum documentation done in time, it was not ideal for scrum as pair programming was often imbalanced. If we had more time on the project or the programming task was larger/more difficult, we would have ensured that programming was distributed fairly. However, since scrum involves rigorous documentation and we did not have much time, I believe that it was the right choice and the best use of my time and skills for myself to focus more on the planning, management, and documentation rather than the programming.

Overall, I feel that I worked to the best of my abilities throughout this project. Due to my roles as the scrum master and product owner, I ensured that the task was completed in time, tasks were distributed based on each members' unique abilities, and that all documentation was made following the scrum framework. With more time for the project, I would have liked to have had more hands-on experience as part of the development team in order to improve my programming skills and gain a better insight into the progress of the project. Furthermore, to avoid problems and conflicts such as members failing to share the work they have done with the rest of the team, I could have worked on some code myself so that I could be responsible for sharing and version-controlling it correctly. Nevertheless, I believe my time and skills were used as efficiently as possible.

Critical Reflection on SCRUM

Scrum is an agile project management methodology or framework used primarily for software development projects with the goal of delivering new software capability at regular intervals. It is one of the approaches that influenced the Agile Manifesto, which articulates a set of values and principles to guide decisions on how to develop higher-quality software faster. (Resources.collab.net, n.d.) Agile software development refers to a group of software development methodologies based on iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams. (Cprime, n.d.) Scrum is just one example of an agile methodology, another being DSDM (Dynamic Systems Development Method) which is an iterative, incremental approach based on the Rapid Application Development (RAD) methodology. Unlike Scrum, DSDM is structured in four strict phases; feasibility and business study, functional model/prototype iteration, design and build iteration, and implementation. (SolutionsIQ, n.d.) In contrast to agile methodologies, the waterfall model is a software development process which emphasises that a logical progression of steps be taken throughout the software development life cycle. While the popularity of the waterfall model has decreased over recent years in favour of more agile methodologies, the logical nature of the sequential process used in the waterfall method cannot be denied, and it remains a common design process in the industry. (Powell-Morse, 2016)

The key difference of the scrum framework to most other frameworks is its distinct roles and events/documentation. The product owner is responsible for creating the product backlog and ensuring the product is of maximum value. On one hand this role is useful to ensure one person is responsible for the team stays on-target however on the other hand, in a small team of 4 it was not efficient to have one less member as part of the development team to do this relatively simple task. Furthermore, to ensure all team members had a good understanding of the task and were all in agreement regarding which tasks to prioritise, we were all involved with creating and maintaining the product backlog. Despite not benefitting from having one person in charge of the product backlog, the product backlog itself was a useful asset in this project. Unlike a simple list of requirements, a product backlog ensured that the team carefully thought about the importance and difficulty of each user story, making prioritising tasks to focus on much easier. Another role unique to the scrum framework is the scrum master, who is responsible for keeping the team focused and

motivated. This role was very useful in this process because it was like having a project manager and it avoided a 'too many cooks' situation caused by too many people working on the coding of a relatively short and simple program. Without this role, it would have been easy for the team to become unfocused and may have led to members having little to do without taking part in programming. For these reasons, I believe the scrum master role was extremely beneficial in the development of our artefact. Lastly, like all frameworks, scrum needs a development team. However, a scrum development team is usually 5-9 programmers, and with just 4 team members and two other roles to play, we were left with just 2 development team members at any one time. Despite this, the development team were not overworked due to the relatively simple nature of the task at hand. For this reason, it may be evident that the scrum framework was not the best choice for this project since even with far less people than recommended, it was difficult to give every member a useful task at all times. In fact, since the scrum framework requires large amounts of documentation, team members were more occupied with creating and filling these out correctly than working on the program itself. This may also suggest that scrum was not necessarily appropriate for this task since more time needed to be spent adhering to the several processes involved in the scrum framework rather than actually developing a product. The first thing the development team must do is to conduct a sprint planning session in which the product backlog items are broken down into subtasks that can be easily translated into code and managed. This was a useful stage in this project since it may have been too difficult to begin programming based on the product backlog alone as some of these items may require various pieces of code and others may have no impact on the program at all. This then produces a sprint backlog which shows which tasks are being worked on and completed, which is handy to update the rest of the team on what is being done and is left to do. However, it was time consuming to update with specific details that may be hard to remember such as how long a task took to be completed. I found this rather contradictory to the agile idea since not every task must be completed in one go before moving onto the next one, so the time spent may be an accumulation that must be calculated. As well as this, the team must undergo regular sprint meetings in which they discuss their progress. During our project, I found these meetings counter-productive as they would often disturb the workflow and serve the same purpose of updating the sprint backlog. In a real scenario, a company might not be impressed to learn that the developers spent more time creating charts and diagrams rather than actually programming since without the extensive documentation, the program may have been able to be completed more quickly.

Instead of using an agile methodology, particularly scrum, we could have used the waterfall method instead. The waterfall method is a linear approach to software development that has distinct phases which are fully completed before the next begins. These stages are; requirements elicitation, analysis, design and prototyping, implementation, system testing, user testing, deployment, and maintenance. If this task was carried out using the waterfall method, our first task would have been to write a complete list of requirements based on the brief and a single session of asking the clients for any additional needs. This may be beneficial in having a clear idea of the goal and finished product before development begins and can save time by not revisiting this initial planning phase at a later date. It may also mean that the team stays focused because they can learn the requirements and once they have a good understanding of them, there is no concern of having to regularly check or attend meetings to see if the requirements have changed. However, it does make the program very inflexible since it relies on all ideas to be presented at the start and is difficult to adapt if either the client or developers come up with a new idea later on. This may put unnecessary pressure on the client since they may not have a clear vision of what they want and may need to see something implemented/drafted before new ideas can spark. The next task would be to design the system, this

could involve things like sketches and prototypes. As well as this, technical details such as the programming language may be decided at this stage. This would have been a good opportunity to divide the workload based on team members' skills, as the more creative could work on designs, and the more advanced programmers could work on the technical details. However, many stages of the waterfall method do not use a team so efficiently, especially since there are no set roles in this methodology like there are in Scrum. The next step in the waterfall method would be implementation. This would mean that either all four members would be actively involved in the coding at all times, which is unnecessary, inefficient, and may cause more problems than having a smaller development team, or, while some members worked on the implementation, others would be left with nothing to do because tasks do not overlap in this methodology. This could have potentially led to lots of time-wasting waiting around for the program to be complete before moving on to the next stage. Once the development is complete and the programmers have ensured there are no errors, the completed system would then be given to users to test. This is an excellent way to gather feedback since the user would be experiencing a finished product and time in the development process has been dedicated to thorough testing. However, any feedback given from the users may be much more difficult to go back and change or implement on a finished product than it would be for a partially-complete product. Furthermore, in our scenario it may have been difficult to get thorough feedback from users and clients within the timeframe. Overall this project could have been less stressful and more organised if it was undertaken using the waterfall method since the phased development cycle enforces discipline. Each step has a clearly defined starting point and conclusion, which makes progress easy to monitor. (Kienitz, 2017) This helps meet agreed upon timescales. On the other hand, the time frame may have simply been too short to use this methodology since projects generally take longer to complete than they would using an agile process. This is because each stage is imperative and thorough, and there is no overlap. Furthermore, the lack of defined roles may have led to too many people working on the fairly short piece of code, or others being unable to engage effectively.

To summarise the advantages of Scrum, it divides even large projects into manageable sprints, whereas the 'implementation' phase of the waterfall method is somewhat vague and overwhelming. Even though the agile nature of Scrum can result in more flexible timescales, the project is likely to be completed more quickly overall since there are no rigorous set phases such as planning and testing. Since each sprint delivers something of tangible value, improvements can be made iteratively which avoids the completed program having to undergo major changes after user feedback. This benefit is not unique to Scrum but is also evident in other agile methodologies such as DSDM. These benefits meant that we were able to complete the project in the short timeframe. Moreover, due to scrum meetings, the effort of each team member is highlighted regularly which meant that it was easy to ensure that everyone was contributing fairly which may have been difficult to do without defined roles such as in the waterfall method.

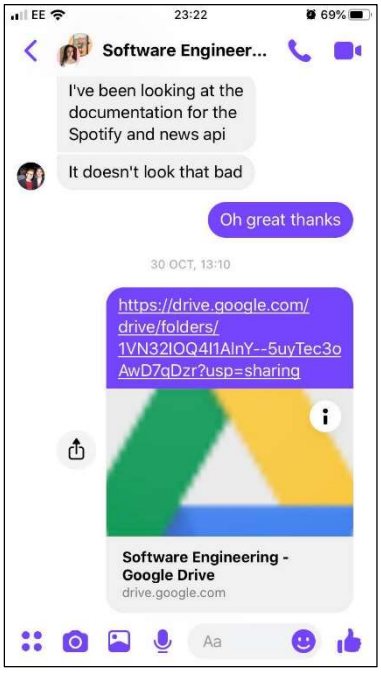
Disadvantages of Scrum have also been highlighted throughout such as how it can lead to 'scope creep' since the end-product and requirements can be unclear and changed or added at any time. The waterfall obviously combats this since the requirements are carefully decided at the start. However, other agile methodologies can also avoid this problem such as the stricter principles of the DSDM framework. Furthermore, other issues were experienced such as that Scrum works better with more experienced, committed teams, and that regular meetings can frustrate team members. This could have been avoided by using the waterfall method since it is a reliable framework that is easier to understand and avoids the need to regularly keep the team up to date with meetings. Despite this, Scrum is likely one of the better agile methods to minimise problems of this nature

since DSDM is not suitable for small organisations or one-time projects, and since it is a much newer model, it is even more difficult to understand and adapt to.

Overall, I think Scrum was a good choice of framework for this project. Even though we worked with less people than a traditional scrum team, it scaled down fairly well and the variety of roles to play ensured that all members were more likely to participate. As well as this, breaking the task down into manageable sprints, regularly keeping the team updated and the management-style role of the scrum master helped to ensure the project could be completed in time. Whilst some of the documentation involved in the scrum framework did not always feel like an efficient use of time, it still could have taken longer if a non-agile method such as waterfall was adopted due to its strict, lengthy phases that do not overlap. Nevertheless, I believe the project could have still been successful under the waterfall model, the benefits and drawbacks experienced would have simply been different. If the project had been larger and more complex, therefore benefitting from more people engaging in programming, and if the timescale was longer, waterfall could have been the better choice. If I were to tackle this project again, I would ensure a more thorough understanding of the scrum framework and its required documentation both for myself and the entire team before starting development. This would have resulted in all documentation being created at the correct intervals, rather than the lag we experienced. As well as this, I think it would have been better to decide the roles at the start of the project and not swap them. This would have undoubtedly led to the team being more focused and organised, and would have also ensured the best use of each members' unique skillsets.

Tools Utilised

Facebook Messenger:

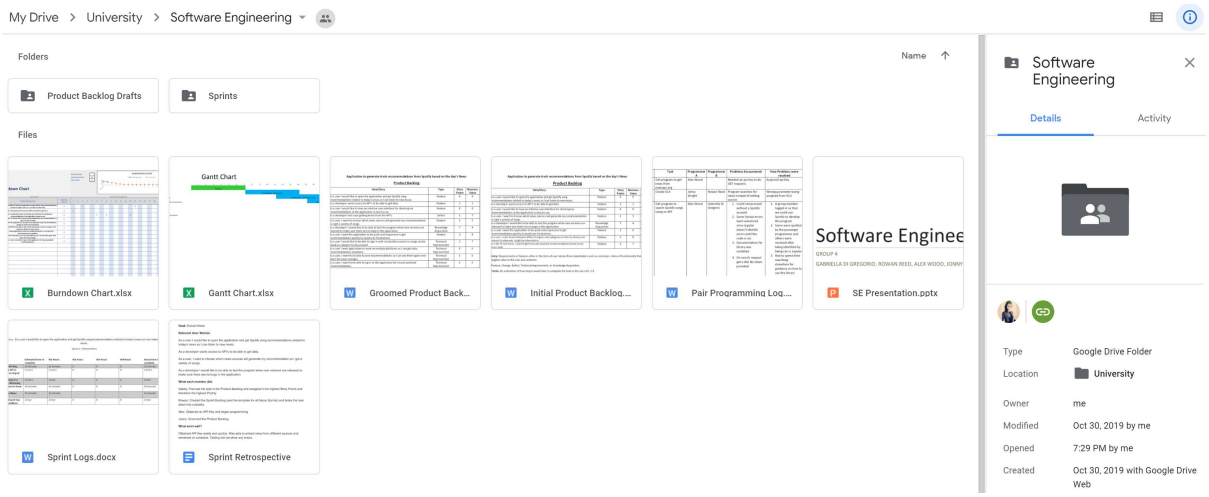


The first tool we utilised when we were informed of the team members was Facebook Messenger. We used this in order create a group chat to start initial conversation and planning, by familiarising ourselves with each other and discussing what must be done and when to meet. Scrum development teams can only be built with a culture of open, respectful, and honest communication. It is important that there is good communication between the team and to remember that even too much communication is better than too little. (Davies, n.d.) For this reason, we wanted to ensure communication was available and easy at all times. Facebook Messenger proved to be an extremely useful tool form arranging meetings, discussing progress, and sharing links to other tools and resources.

A major advantage of this tool was that all team members had access to the app on our mobile devices at all times, which meant we were able to communicate very easily and at any time. This was extremely convenient for sending quick updates such as being on our way to meetings or having completed a task. Furthermore, Facebook Messenger can be used for collaboration in the business world as it allows users to send files and links with ease. If one is telecommuting or working at a mobile office site, it can keep employees in touch with each other if they are working on a project together. (Gaille, 2016) Therefore, we were using an effective business tool to aid our collaborative working.

However, Facebook Messenger is by no means an ‘all-in-one’ tool, we had to use many other tools alongside it due to lacking features. For example, whilst you can share files through Facebook Messenger, we did not do this because there is no way of organising them and the only way to access files previously shared is to scroll back through the conversation which is not convenient. As well as this drawback, a potential disadvantage of over-utilisation of this tool is that group members may have been less inclined to attend regular meetings since updates could be sent in the group chat rather than spoken in person.

Google Drive:



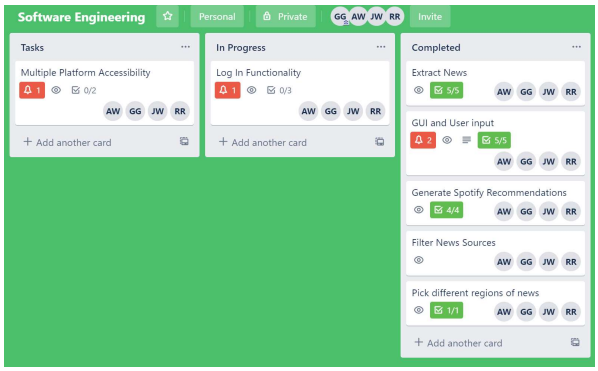
Since the first task was to create a Product Backlog, each team member created a draft from different perspectives. We wanted to share our ideas so I created a Google Drive folder in which we could share documents such as these. In the next meeting, it was very easy to put our ideas together into one Product Backlog since all of our drafts were easily accessible in the same place. I also utilised this space to upload the presentation I created in a meeting to show our progress and results after our first sprint. This meant that any group member could add things such as screenshots and speaker notes from any location before presenting it to our clients. This Google Drive space can be used and edited by any team member and can be organised appropriately. This tool was used effectively throughout the entire project since every resource created other than the code itself was uploaded here.

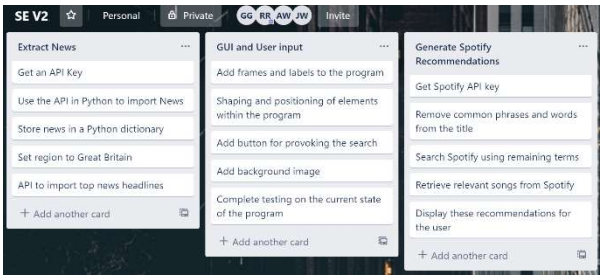
The advantage of using cloud storage is that our work can be accessed at any time, in any place, and on any device as long as an internet connection and browser is available. This meant that collaborative working could easily take place at any time from anywhere rather than being restricted by time restraints of meetings. Google Drive is recognised as one of the best collaboration tools for productive teams in business since Google’s collaboration tools include its Docs and Sheets services, which are designed to allow teams to edit files at the same time and save all their changes automatically. (Bika, n.d.) Furthermore, since Scrum is an agile method meaning that planning and the artefact are continuously updated, version control is important. Google Docs have a built-in version history tool so that changes can easily be identified and reverted. So evidently, this was another effective tool that enabled us to collaborate successfully.

However, although almost any file type can be uploaded to Google Drive to be downloaded by anyone given access, the benefits of online working and version control are restricted to documents only. Therefore, it was not appropriate to use Google Drive for code as we would have not been able to work on it collaboratively or have the safety of version control. As well as this, even though Google Drive is very accessible and widely available – it still relies on a stable internet connection

and comes with online risks such as hackers. For important work it was crucial not to solely rely on Google Drive to store or work, backups on physical devices were a necessary precaution to take in order to avoid the risk of not being able to access our work.

Trello:





Trello has an intriguing user-friendly interface, it is easy to learn and works well for monitoring projects and assigning tasks. Trello also makes using Agile, Scrum and other project management frameworks easy. (Bika, n.d.) Our next task was to carry out Sprint Planning in order to create a Sprint Backlog. Whilst post-it notes on a whiteboard might come to mind, we wanted something more permanent that can be accessed and updated from anywhere. Because it's so easy to use, Trello is an amazing Scrum and Agile solution and Trello's features made it the perfect tool for representing Sprint Backlogs. It works like a traditional whiteboard but in a digital form and the flexibility of Trello boards is perfectly aligned with the Scrum framework, as it gives you full visibility into project stages, roles, and deadlines. (Nevogt, 2019) We used this tool to create Sprint boards, containing Product Backlog items broken down into small, specific tasks that we were able to drag across the board from 'To Do' to 'In Progress' to 'Completed' as well as assigning team members to each tasks and a due date.

Similarly to Google Drive, an advantage of Trello is that we were able to access our boards any time, any place and on any device and work on them collaboratively and remotely. As well as this, Trello has a notification system so each team member would be notified of any changes or updates made which was extremely useful since we would be informed immediately if our tasks changed. Another advantage of Trello is its ease of use – although none of us had experienced using it before, it was extremely intuitive and easy to learn.

Despite its numerous advantages, Trello is not the perfect tool either. Whilst each task can be assigned a due date, there is no built-in way of adding an estimation of how long they would/did take. This is a crucial part of a Sprint Backlog, so our options were to improvise by adding the time estimation to the title of the task, or by using our Trello Board to recreate a table elsewhere with these missing features. Furthermore, planning is further restricted in Trello due to its lack of a Gantt Chart Feature or Calendar. Additionally, you cannot review Sprint Iterations through Trello. Working in iterations can be successful only when retrospective is possible. Performance of work will improve when it is possible to set a regular interval to review the work done and chart out a plan to review iterations. (Oamkumar, 2018) Therefore, regular meetings were crucial to review each Sprint and logging/documentation elsewhere was required.

GitHub:

RowanReed98 / CMP3111M Private

Watch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Actions Projects 0 Security Insights

Software Engineering

5 commits 5 branches 0 packages 0 releases

Branch: master New pull request Create new file Upload files Find file Clone or download

alexwood99 Rename README to master Latest commit ab8d646 9 days ago

master	Rename README to master	9 days ago
testupload.txt	Add files via upload	13 days ago

Default branch

master Updated 9 days ago by alexwood99 Default

Active branches

news-search-spotify	Updated 8 days ago by alexwood99	1 | 5
spotipy	Updated 13 days ago by alexwood99	1 | 2
README.md	Updated 13 days ago by alexwood99	3 | 1
Newsapi	Updated 13 days ago by jonny1245	1 | 2

6 commits 5 branches 0 packages 0 releases

Branch: Newsapi New pull request Create new file Upload files Find file Clone or download

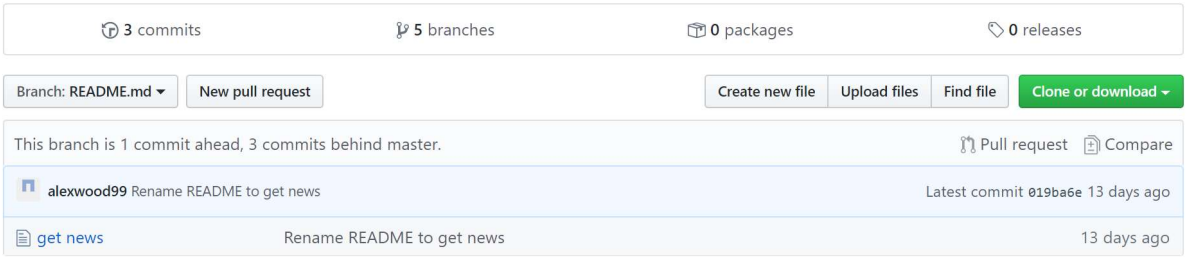
This branch is 2 commits ahead, 1 commit behind master. Pull request Compare

jonny1245 Add files via upload Latest commit 26f3dc7 13 days ago

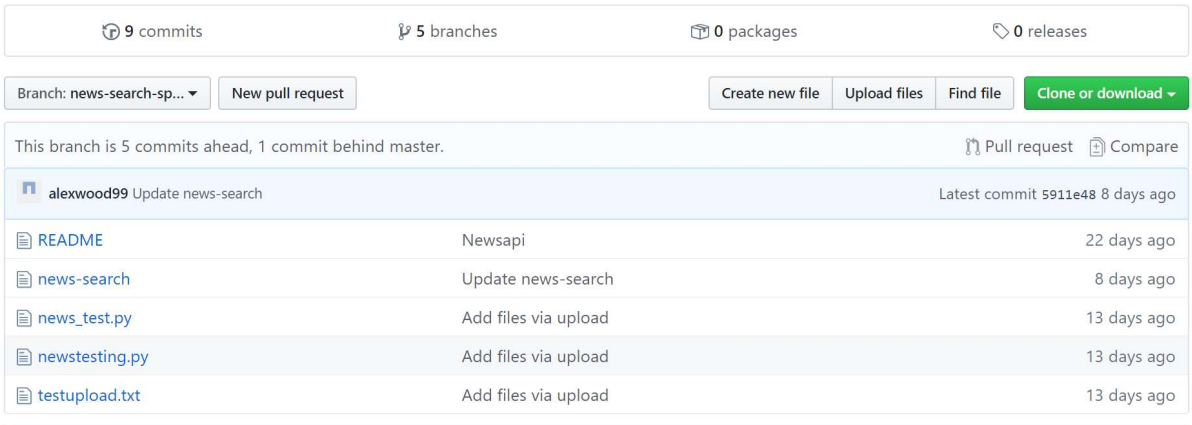
README	Newsapi	22 days ago
news_test.py	Add files via upload	13 days ago
newstesting.py	Add files via upload	13 days ago
testupload.txt	Add files via upload	13 days ago

README

```
from newsapi import NewsApiClient
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

stop_words = set(stopwords.words('english'))

#init
newsapi = NewsApiClient(api_key='6a4d8068c73d4b6e8dc5f9ed70d34beb')

data = newsapi.get_top_headlines(sources='bbc-news')

articles = data['articles']
for w in articles:
    word_tokens = word_tokenize(w["title"])
    filtered_sentence = [w for w in word_tokens if not w in stop_words]
    filtered_sentence = []
    for w in word_tokens:
        if w not in stop_words:
            filtered_sentence.append(w)

    print(word_tokens)
    print(filtered_sentence)
```

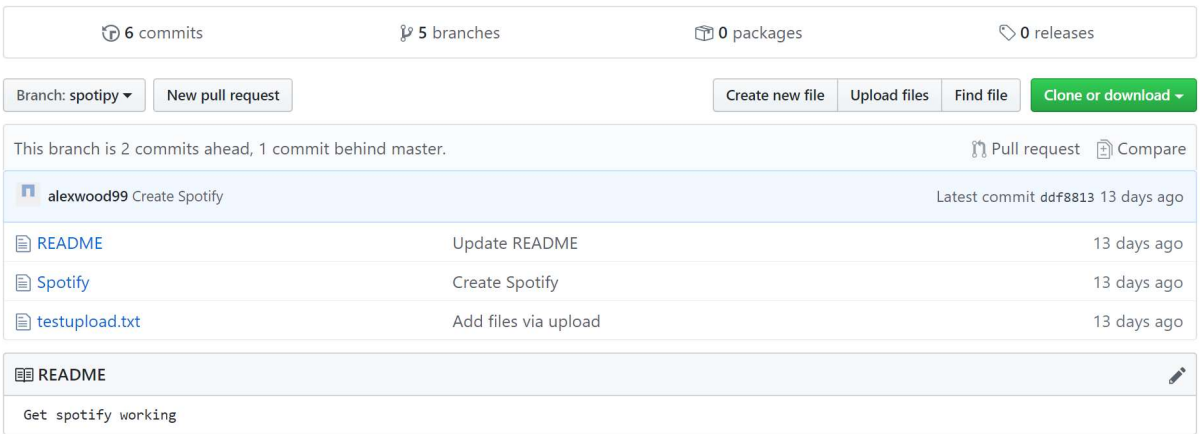


Software Engineering



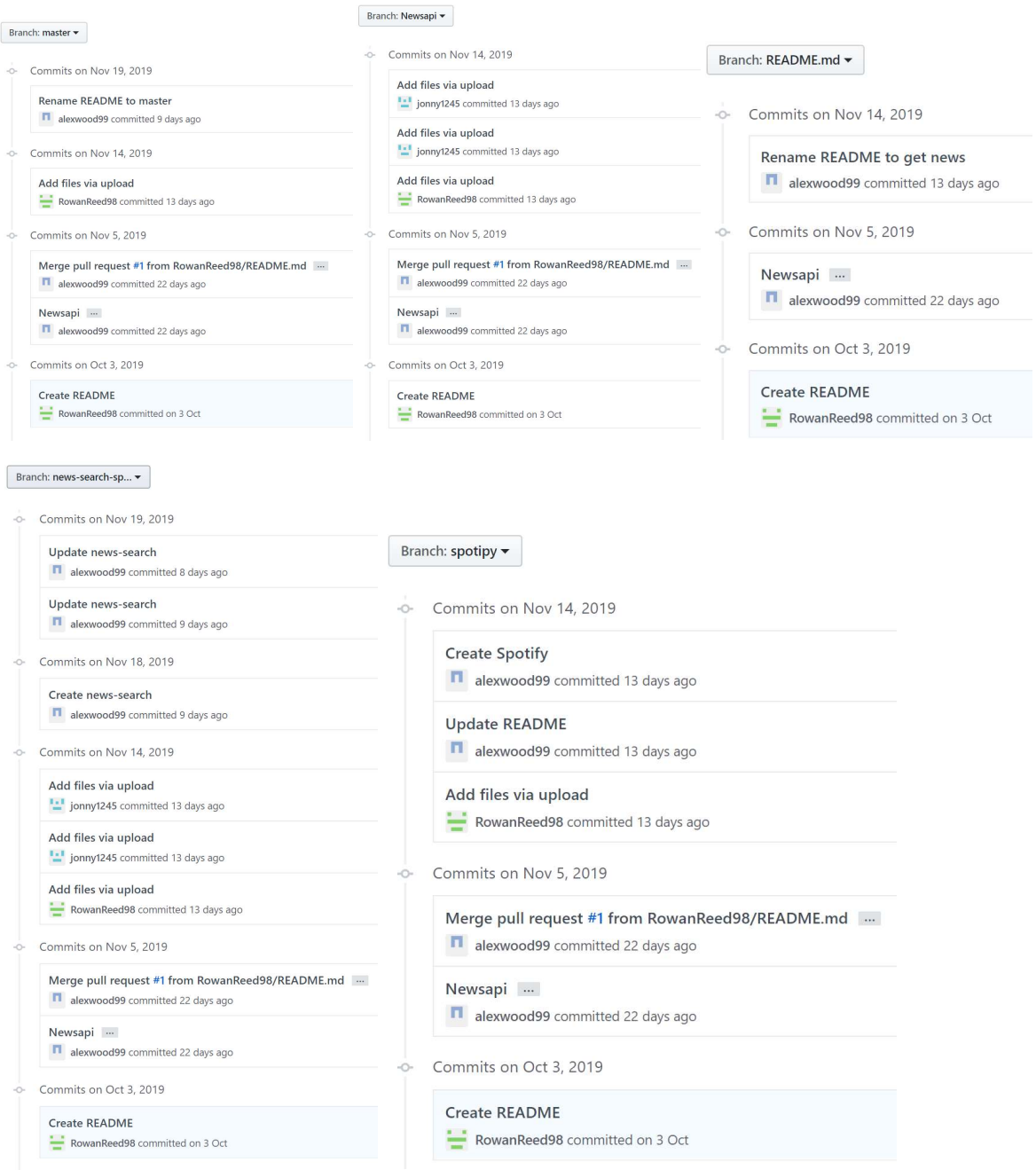
README

```
from newsapi import NewsApiClient
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

stop_words = set(stopwords.words('english'))

#init
newsapi = NewsApiClient(api_key='6a4d8068c73d4b6e8dc5f9ed70d34beb')

data = newsapi.get_top_headlines(sources='bbc-news')

articles = data['articles']
for w in articles:
    word_tokens = word_tokenize(w["title"])
    filtered_sentence = [w for w in word_tokens if not w in stop_words]
    filtered_sentence = []
    for w in word_tokens:
        if w not in stop_words:
            filtered_sentence.append(w)

    print(word_tokens)
    print(filtered_sentence)
```



README

```
Get spotify working
```



Since we would be collaboratively working on code, a version-controlled sharing tool was essential. Since none of us had much prior experience with using these types of tools, we decided to go for the most obvious choice, GitHub. At the very start of the project, a member of the team created a GitHub repository which he shared with all other members so that we would be able to share and maintain the code throughout. Once the development process began, the code was put on GitHub at regular intervals, particularly after each Sprint. Due to the small development team and the use of Pair Programming, there was less need for every member to access it on their own devices as there may have been in some other frameworks, but this was still a handy feature. However, the main purpose of doing this is to have version-control so that a working piece of code is not completely overwritten. This meant that if further development was not successful, an old version could be easily reverted to without having to go through the time-consuming process of carefully undoing changes. As well as this, Scrum ensures that each part of the project is broken down into manageable chunks, so our programming was done in these chunks too. So, the first sprint resulted

in a branch with working code for implementing the News API and filtering terms, and another resulted in a dedicated branch for searching terms in Spotify, and so on. These individual working pieces of code could then be merged together in a new branch, as the finished product.

There are many advantages to using GitHub, such as how it acts as a backup of your work. Although using an online repository should never be considered infallible, it provides a nice and simple way to have code and version history available online, regardless of what happens to the local machine. (Clancy, 2018) As well as this, GitHub is a Git repository hosting service, but it adds many of its own features. While Git is a command line tool, GitHub provides a Web-based graphical interface. It also provides access control and several collaboration features, such as a wikis and basic task management tools for every project. (Finley, 2012) Although we did not take advantage of all the unique features GitHub offers since Scrum requires specific and unique documentation that we decided to create elsewhere, it is a clear benefit and can come close to being an all-in one tool for shared projects. Moreover, GitHub can serve as a tool to showcase your work, in one easily distributable link. As young developers, this may give us opportunity for our work to be seen by potential recruiters and can effortlessly build up a portfolio.

However, despite its overwhelming popularity amongst developers, GitHub has some drawbacks too – some of which we experienced during our project. In my opinion, the most noticeable weakness of GitHub is its extremely unintuitive and unappealing interface. There is definitely a learning process involved with using GitHub, since it is not immediately obvious at first use. We found that our unfamiliarity with GitHub led to time being wasted trying to understand how it works, and also led to using it fairly inefficiently since we were not aware of its full capabilities and we may have been less inclined to use it as regularly as we should have. Furthermore, the main focus of this project was the planning and documentation, rather than the code itself. GitHub may not be the best tool for capturing the creative process or for recording ideas. Although it is very good for tracking code, it's not the best for tracking design. (Qayyum, n.d.) For this reason, many more tools were needed to aid the processes involved in this framework since GitHub was not sufficient to accurately represent the Scrum processes.

Python and APIs:

In order to tackle our task, we first had to choose a programming language. I started by asking my teammates if they had any preferred languages or if anyone was particularly advanced in one in case we could use expertise to our advantage. However, we all agreed that none of us had a preference or were particularly strong programmers. Therefore, we wanted to ensure we used a language that is easy to learn and use to avoid any unnecessary complications, difficulty or wasting time on learning the fundamentals. Our next consideration was ensuring compatibility and that we used the most efficient tools available. We found that the News API (Newsapi.org) and the Spotify Web API (Developer.spotify.com, n.d.) were both supported in Python and offered us the fastest and simplest way to both extract news articles and retrieve Spotify content such as songs, and these are the two key parts to the task at hand. As well as supporting these handy APIs, Python is a relatively simple and easy to use programming language so was therefore the obvious choice.

Compared to other programming languages Python is the most broadly applied by the developers lately. The main advantage of Python is that it is easy to read and easy to learn. It is easier to write a program in Python than in C or C++, and it is easier to read which also makes the code easier to sustain and reduces the cost for maintenance. (Webcase Studio, 2018) This advantage was very useful and relevant to our team since we lacked expertise in programming, and it is important that code is readable and easy to understand when numerous people are working on it. Its simplistic

nature avoided the need to spend large amounts of time learning the basics or trying to figure out what another team member had written. Another advantage of Python is its extensive support Libraries. It provides large standard libraries that include the areas like string operations, Internet, web service tools, operating system interfaces and protocols. Most of the highly used programming tasks are already scripted into it that limits the length of the codes to be written in Python. This and clean object-oriented designs that increase two to ten-fold of programmer's productivity compared with using the languages like Java, C++ and C#. (Mindfire Solutions, 2017) This was very beneficial to this task since we had a very short amount of time to complete the task, so time-saving libraries were essential to complete it in time since it avoided having to learn how to code large amounts of code manually.

Like all programming languages, Python also has its drawbacks. Whilst it was very appropriate for our task and was undoubtably the best choice, we did experience one of Python's most known disadvantages, its speed. Python uses an interpreter that loads it line by line instead of a compiler that executes the whole file at once. This makes compilation slower and tends to perform slowly. This is the major reason competitive programmers don't use python, C++ provides more computation per seconds instead of python. Moreover, this is why Python is not extensively used in application development. (Edureka, 2019) We noticed this since our program does not execute immediately, it does take a short while to generate song recommendations. However, fast execution was not a requirement nor priority for this task therefore it was still a suitable solution in this case.

Group Evaluation

Group Member	**Average Contribution**
Gabriella Di Gregorio	100%
Rowan Reed	100%
Alex Wood	100%
Jonny Wright	100%

References

Edureka. (2019). *Advantages and Disadvantages of Python*. [online] Available at: https://www.edureka.co/blog/advantages-and-disadvantages-of-python/ [Accessed 19 Nov. 2019].

Mindfire Solutions (2017). *Advantages and Disadvantages of Python Programming Language*. [online] Medium. Available at: https://medium.com/@mindfiresolutions.usa/advantages-and-disadvantages-of-python-programming-language-fd0b394f2121 [Accessed 19 Nov. 2019].

Webcase Studio. (2018). *ADVANTAGES AND DISADVANTAGES OF PYTHON PROGRAMMING LANGUAGE*. [online] Available at: https://webcase.studio/blog/advantages-and-disadvantages-python-programming-language/ [Accessed 19 Nov. 2019].

Bika, N. (n.d.). *The 14 best collaboration tools for productive teams*. [online] Workable. Available at: https://resources.workable.com/tutorial/collaboration-tools [Accessed 8 Nov. 2019].

Blackburn, M. (2019). *Burndown Chart: What Is It & How Do I Use It?* [online] ProjectManager.com. Available at: https://www.projectmanager.com/blog/burndown-chart-what-is-it [Accessed 24 Nov. 2019].

Clancy, J. (2018). *The Pros and Cons of Using GitHub for Repository Management*. [online] CodeClouds. Available at: https://www.codeclouds.com/blog/advantages-disadvantages-using-github/ [Accessed 27 Nov. 2019].

Davies, A. (n.d.). *How to Build a Scrum Development Team?* [online] DevTeam.Space. Available at: https://www.devteam.space/blog/how-to-build-a-scrum-development-team/ [Accessed 8 Nov. 2019]

SolutionsIQ. (n.d.). *Dynamic Systems Development Method (DSDM)*. [online] Available at: https://www.solutionsiq.com/agile-glossary/dynamic-systems-development-method-dsdm/#:~:targetText=DSDM%20is%20an%20agile%20software,Functional%20model%20%2F%20prototype%20iteration [Accessed 25 Nov. 2019].

Finley, K. (2012). *What Exactly Is GitHub Anyway?*. [online] TechCrunch. Available at: https://techcrunch.com/2012/07/14/what-exactly-is-github-anyway/ [Accessed 27 Nov. 2019].

Gaille, B. (2016). *12 Facebook Messenger Pros and Cons*. [online] BrandonGaille.com. Available at: https://brandongaille.com/12-facebook-messenger-pros-and-cons/ [Accessed 8 Nov. 2019].

Kienitz, P. (2017). *Pros and Cons of Waterfall Software Development*. [online] DCSL Software Ltd. Available at: https://www.dcslsoftware.com/pros-cons-waterfall-software-development/ [Accessed 26 Nov. 2019].

Malsam, W. (2018). *What Is a Scrum Master? Here's Everything You Need to Know*. [online] ProjectManager.com. Available at: https://www.projectmanager.com/blog/what-is-a-scrum-master-everything-you-need [Accessed 25 Nov. 2019].

Nevogt, D. (2019). *How Trello Can Be Your Secret Weapon for Agile Work*. [online] Hubstaff. Available at: https://blog.hubstaff.com/agile-trello/#why-use-trello-for-scrum-and-agile [Accessed 8 Nov. 2019].

Newsapi.org. (n.d.). *News API - A JSON API for live news and blog articles*. [online] Available at: https://newsapi.org/ [Accessed 19 Nov. 2019].

Oamkumar, R. (2018). *Advantages and Disadvantages of Trello*. [online] Software Developer India. Available at: https://www.software-developer-india.com/advantages-and-disadvantages-of-trello/ [Accessed 8 Nov. 2019].

Pichler, R. (2013). *The Product Backlog's Strengths and Limitations*. [online] Roman Pichler. Available at: https://www.romanpichler.com/blog/the-product-backlogs-strengths-and-limitations/ [Accessed 24 Nov. 2019].

Powell-Morse, A. (2016). Waterfall Model: What Is It and When Should You Use It?. [online] Airbrake. Available at: https://airbrake.io/blog/sdlc/waterfall-model [Accessed 25 Nov. 2019].

Qayyum, A. (n.d.). *Understand the pros and cons of using GitHub*. [online] Smashinghub.com. Available at: http://smashinghub.com/understand-the-pros-and-cons-of-using-github.htm [Accessed 27 Nov. 2019].

Radigan, D. (n.d.). *The product backlog: your ultimate to-do list*. [online] Atlassian. Available at: https://www.atlassian.com/agile/scrum/backlogs [Accessed 24 Nov. 2019].

Mountain Goat Software. (n.d.). *Scrum Product Owner: The Agile Product Owner's Role*. [online] Available at: https://www.mountaingoatsoftware.com/agile/scrum/roles/product-owner [Accessed 25 Nov. 2019].

Agile Alliance. (n.d.). *Sprint Planning*. [online] Available at: https://www.agilealliance.org/glossary/sprint-planning/#q=~(infinite~false~filters~(postType~(~'page~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video)~tags~(~'sprint*20planning))~searchTerm~'~sort~false~sortDirection~'asc~page~1) [Accessed 24 Nov. 2019].

Developer.spotify.com. (n.d.). *Web API | Spotify for Developers*. [online] Available at: https://developer.spotify.com/documentation/web-api/ [Accessed 19 Nov. 2019].

Productplan.com. (n.d.). *What is Backlog Grooming? | Definition and Overview*. [online] Available at: https://www.productplan.com/glossary/backlog-grooming/ [Accessed 24 Nov. 2019].

Scrum.org. (n.d.). *What is a Product Backlog?* [online] Available at: https://www.scrum.org/resources/what-is-a-product-backlog [Accessed 24 Nov. 2019].

Scrum.org. (n.d.). *What is a Product Owner?* [online] Available at: https://www.scrum.org/resources/what-is-a-product-owner [Accessed 25 Nov. 2019].

Stackify. (2017). *What is Pair Programming? Advantages, Challenges, Tutorials & More*. [online] Available at: https://stackify.com/pair-programming-advantages/ [Accessed 24 Nov. 2019].

Resources.collab.net. (n.d.). *What Is Scrum Methodology?* [online] Available at: https://resources.collab.net/agile-101/what-is-scrum [Accessed 25 Nov. 2019].