Software Engineering Assessment 1

by

Daniel Dixon

DIX16602092

School of Computer Science

University of Lincoln

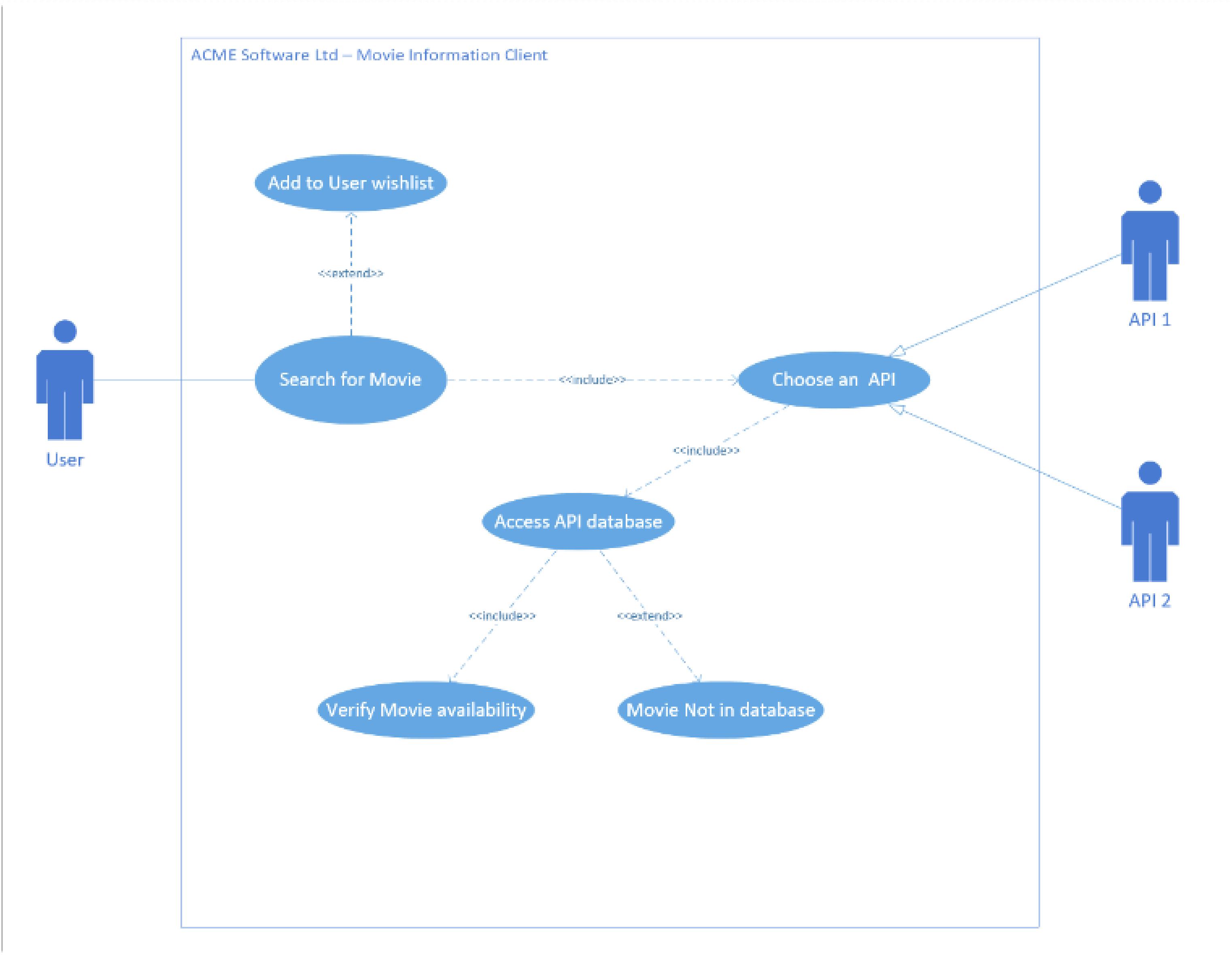
2018

Artefact Creation:

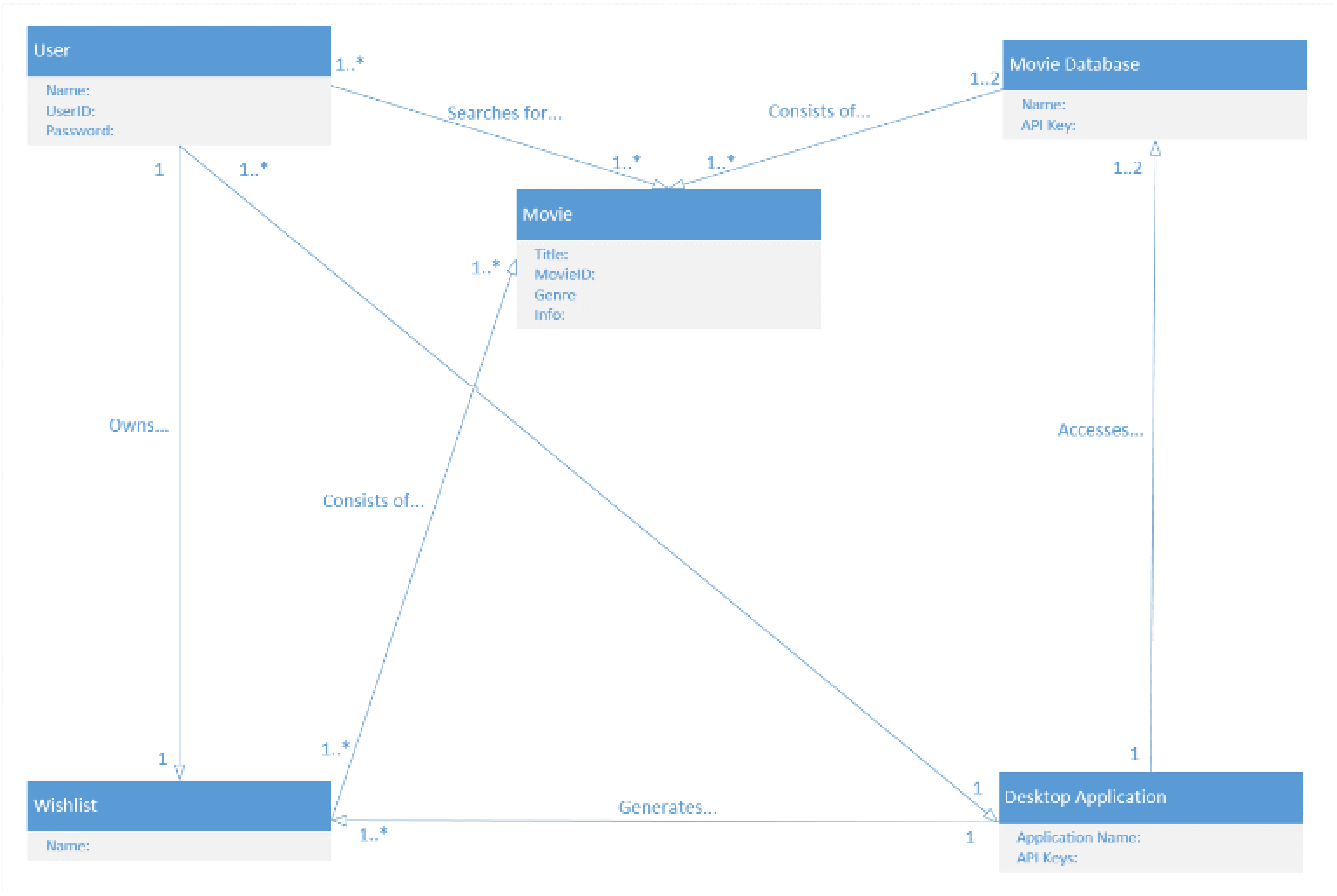
Artefact Code: https://github.com/cwells1298/Software-Engineering-C17

UML and Domain models:

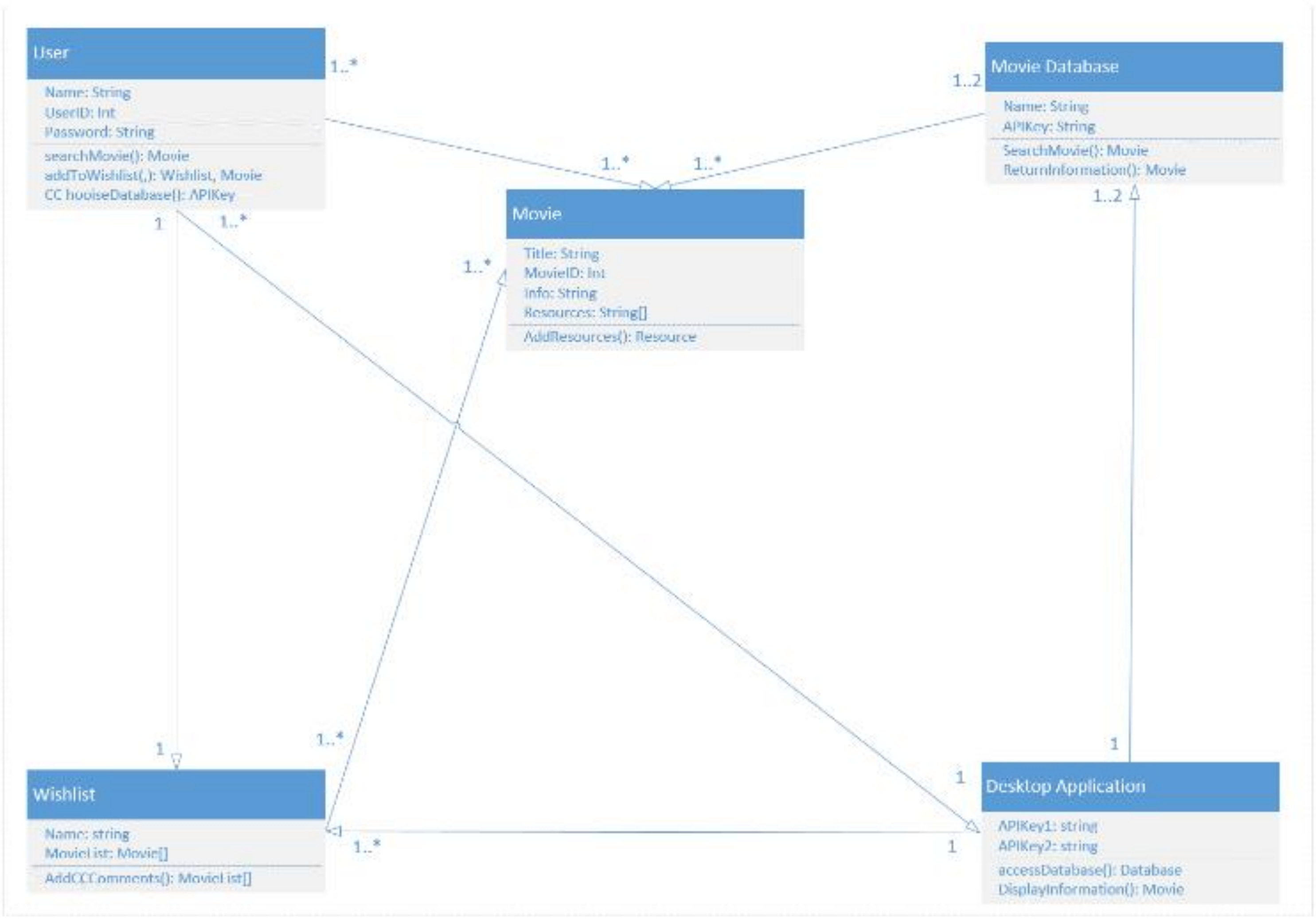
1. UML Case Diagram



2. UML Domain Model



3. UML Design Model



Product Backlog:

ID	Product Backlog Item	Story Points	Business Value	Priority	PBI Type	Effort Required
1	UML and Domain Modelling research	4	3	7	Knowledge Acquisition	1
2	UML and Domain Modelling creation	7	8	7	Knowledge Acquisition	3
3	Understand coding language differences	6	7	9	Knowledge Acquisition	1
4	Determine coding language and syntax	8	5	8	Knowledge Acquisition	1
5	Design a UI framework	7	8	5	Knowledge Acquisition	3
6	Choose and research coding libraries	8	5	8	Knowledge Acquisition	2
7	Research API and key acquisition	9	7	9	Knowledge Acquisition	1
8	Create initial prototype API interaction	9	8	8	Knowledge Acquisition	4
9	Create initial prototype GUI	7	8	6	Knowledge Acquisition	4
10	Merge prototypes to display results on GUI	7	7	7	Feature	3
11	Allow input from user	8	9	8	Feature	1
12	Query API with user input	8	9	9	Feature	1
13	Research Second API and key acquisition	9	7	9	Knowledge Acquisition	1
14	Allow user to choose between two databases	6	8	6	Change	3
15	Creation of wish list functionality	7	7	6	Feature	2
16	Allow randomisation functionality	6	5	4	Feature	2
17	Wish list commenting	6	5	5	Technical Improvement	2
18	Attempt different merging technique	7	7	7	Defect	2
19	Redesign UMLs to new specification	6	5	7	Change	1
20	Create menu bar	7	8	7	Technical Improvement	1

The Backlog was first created before initial development started but was edited throughout the process. This included interacting with a second API and adding a wish list which were both added to the Backlog to increase functionality.

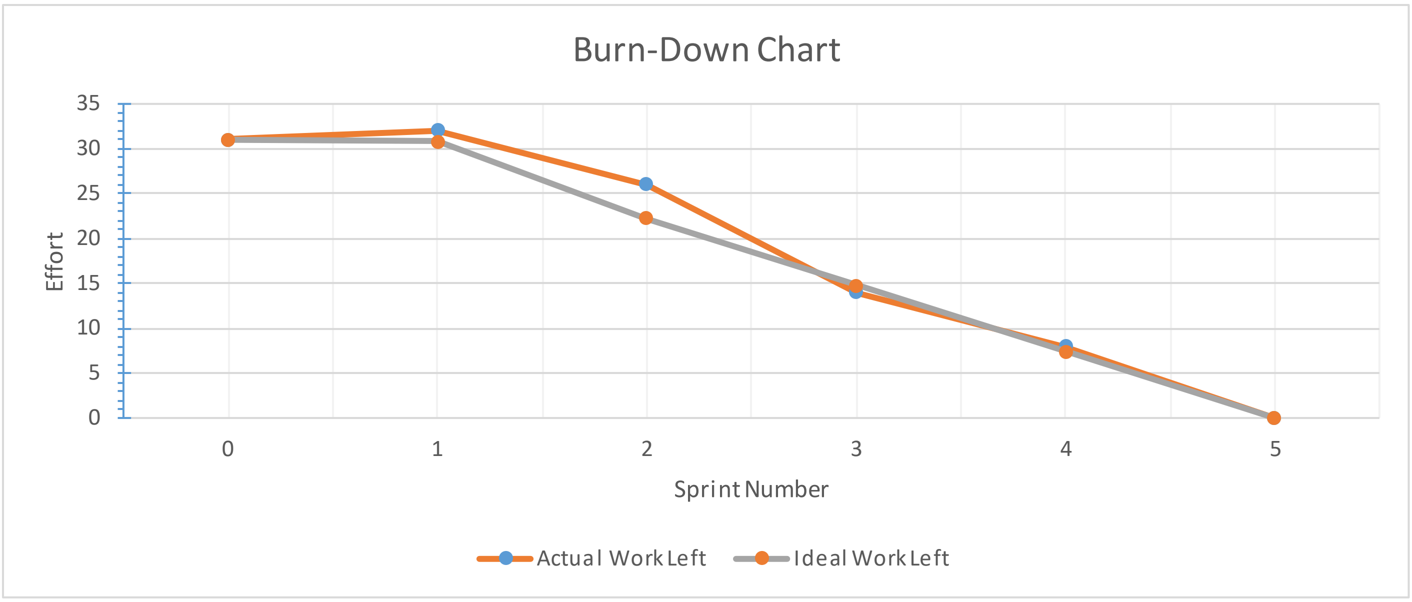
Sprint Logs:

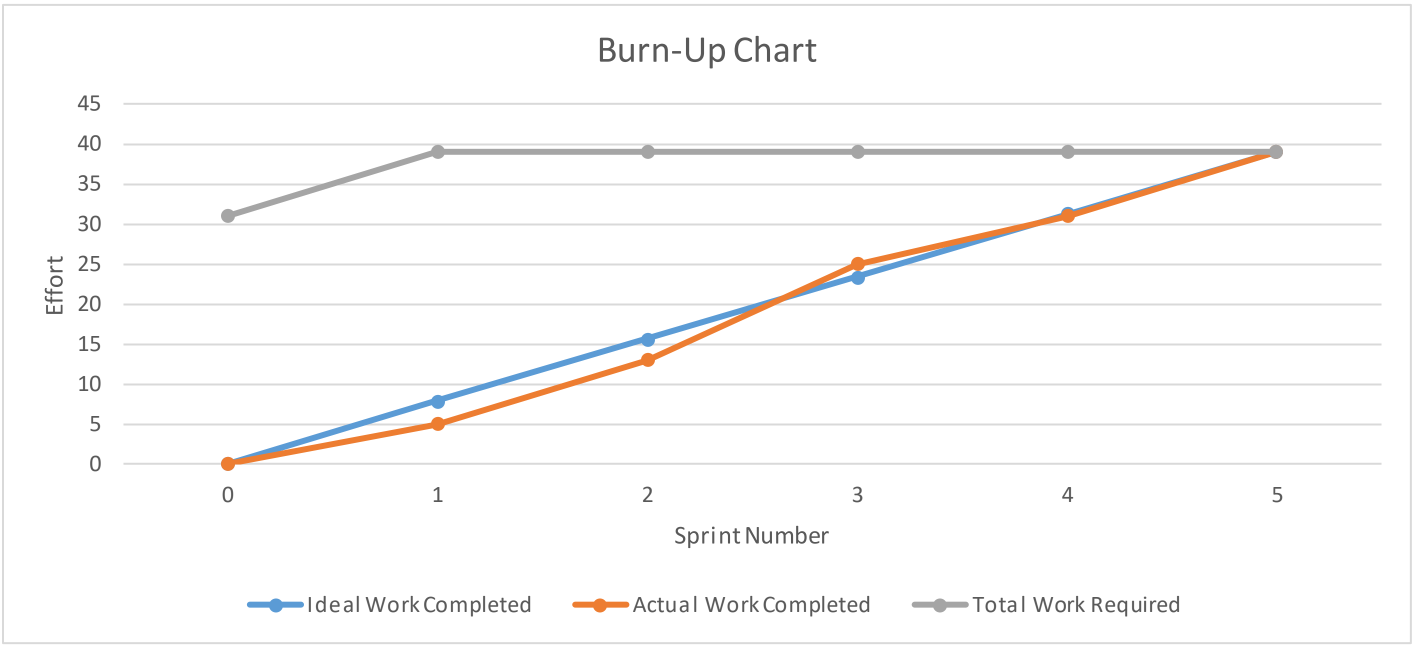
Sprint No	Start Date	End Date	Tasks	Completed
1	31/10/2018	05/11/2018	UML and Domain Modelling research, UML and Domain Modelling creation, Understand coding language differences	UML and Domain Modelling research, UML and Domain Modelling creation, Understand coding language differences
2	08/11/2018	13/11/2018	Determine coding language and syntax, Design a UI framework, Choose and research coding libraries, Create initial prototype API interaction, Allow input from user	Determine coding language and syntax, Choose and research coding libraries, Create initial prototype API interaction
3	15/11/2018	20/11/2018	Allow input from user, Design a UI framework, Create initial prototype GUI, Merge prototypes to display results on GUI	Allow input from user, Design a UI framework, Create initial prototype GUI
4	27/11/2018	02/12/2018	Redesign UMLs, Query API with user input, Merge prototypes to display results on GUI, Research second API and key acquisition, Allow user to choose between two databases,	Redesign UMLs, Query API with user input, Merge prototypes to display results on GUI, Research second API and key acquisition, Allow user to choose between two databases
5	02/12/2018	08/12/2018	Allow database choice, Create wish-list functionality, Allow randomisation functionality, Wish-list commenting, Create menu bar	Allow database choice, Create wish-list functionality, Allow randomisation functionality, Wish-list commenting, Create menu bar

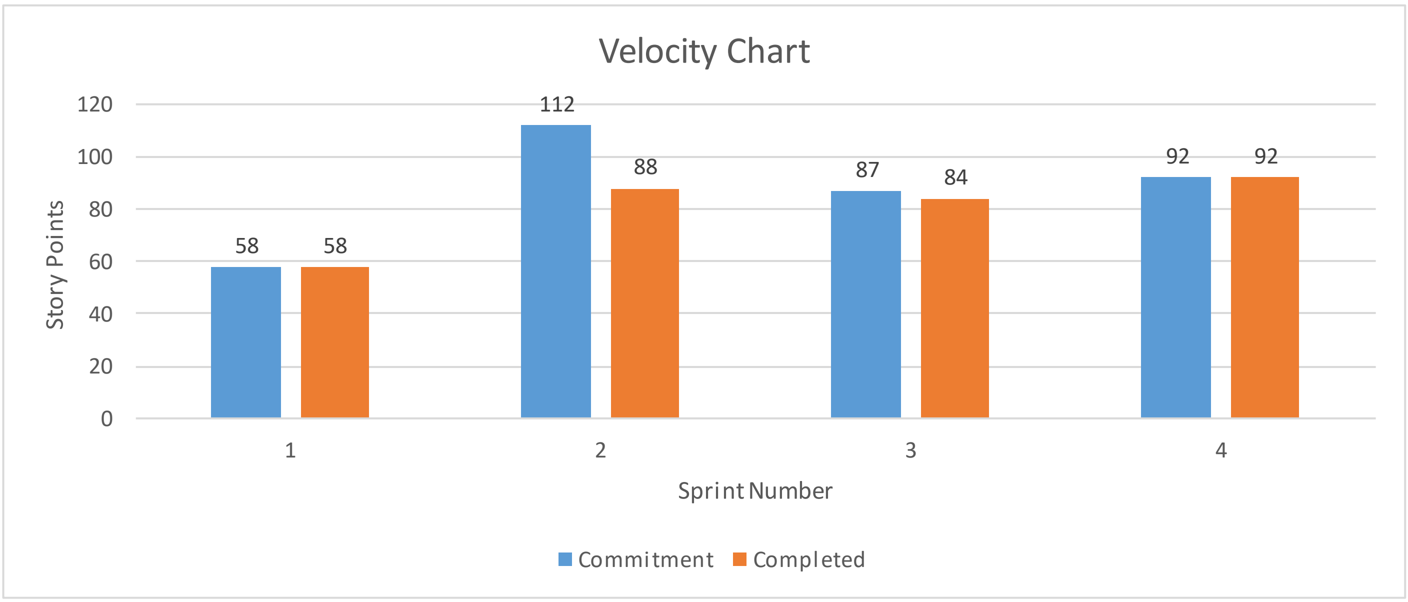
Sprint No	Sprint Review Comments
1	No issues with work completed, more work could be attempted for the next sprint.
2	No members absent for sprint initialisation but some members did not complete tasks/ turn up to sprint review. Pair Programming helped with coding issues. Started uploading sprints to GitHub under separate branches
3	Again, absent members made work difficult but were able to meet some of the goals set for ourselves. GitHub pull requests start to be used
4	Introduction of new needed functions, update to a lot of logs and backlog. Large amounts of work completed by the team.
5	Finishing up coding and logs, upload and commit changes to master for the final time.

Sprint Charts:

Sprint No.	**Story Points Value Forecast**	**Story Points Value Completed**	**Activity IDS Forecast**	**Activity IDS Completed**				
1	58	58	1,2,3	1,2,3				
2	112	88	4,5,6,8,11	4,6,7,8				
3	87	84	5,9,10,11	5,9,11,19				
4	92	92	12,13,14,18	12,13,14,18				
5	73	73	15,16,17,20	15,16,17,20				
Sprint No.	**Ideal Work Left**	**Actual Work Left**	**Tasks Completed**	**Tasks Added**	**Point Value Change**	**Ideal Work Completed**	**Actual Work Completed**	**Total Work Required**
0	31	31	N/A	N/A	0	0	0	31
1	31	32	1,2,3	13,14,19,20	1	8	5	39
2	22	26	4,6,7,8	18	-6	16	13	39
3	15	14	5,9,11,19	N/A	-13	23	25	39
4	7	8	12,18,14	N/A	-6	31	31	39
5	0	0	15,16,17,20	N/A	-8	39	39	39


Burn-Down Chart
Effort
35
30
25
20
15
10
5
0
0
1
2
3
4
5
Sprint Number
Actual Work Left
Ideal Work Left


Burn-Up Chart
Effort
45
40
35
30
25
20
15
10
5
0
0
1
2
3
4
5
Sprint Number
Ideal Work Completed
Actual Work Completed
Total Work Required


Velocity Chart
Story Points
120
100
80
60
40
20
0
58
58
112
88
87
84
92
92
1
2
3
4
Sprint Number
Commitment
Completed

Pair Programming Logs:

Date of Log	**Time Started**	**Driver**	**Observer**	**Lines of Code Written**	**Errors Spotted**	**Activity tested or compiled**
13/11/2018	13:18	Daniel	Conor	32	4 (2 Daniel, 2 Conor)	Setup of IDE using chosen libraries, Using libraries to interact with API
15/11/2018	13:10	Sam	Daniel, Conor	24	2 (1 Daniel, 1 Conor)	Create Initial UI, Create variables that can be merged
27/11/2018	13:06	Daniel	James, Conor	22	5 (2 Daniel, 1 James, 2 Conor)	Menu bar, Merge UI and Input, Allow Database Choice
02/12/2018	17:00	Daniel	James	37	3 (1 Daniel, 2 James)	Database Choice, Randomisation

Date of Log	**Errors/Comments**
13/11/2018	Errors: Coding language not placed within computer path (Daniel) Missed pass-through variable on function (Conor) No Semi-colon at end of line (Daniel) Use of an object that had not yet been created (Conor) Comments: Selenium must open a web browser (makes code look malicious), using requests library instead, may lose JSON but still able to get raw data
15/11/2018	Errors: UI Frame not linked to UI object (Daniel) Linking error, incompatible data types (Connor) Comments: Failed to merge UI and input, try again next time
27/11/2018	Errors: Wish list menu linked to incorrect function (Daniel) Choice of Database using same Database for both (Daniel) Input from users can accidentally have extra characters (James) User input must be sanitised for security (Conor) Menu bar not correctly linked to menu section correctly (Conor) Comments: Second API did not work well with the code but workarounds found
02/12/2018	Errors: Found that changing database before searching can cause errors(Daniel) Randomisation range could go out of bounds for IMDB ID (James) Hitting random too quickly can time out connection (James) Comments: Completed Second API connection, Randomisation needed testing of usable range

Design Patterns:

Facade Pattern

Name: User Interface (UI)
Problem Description: User must be presented retrieved data in an easily readable format, as well as making the ability of traversing functions easier
Solution: Design a graphical interface that the user can interact with, allowing mouse and keyboard as input and keeping user away from the base code. Will Group functionality into more convenient locations
Consequences: User loses ability to customise the program. All other patterns must be layered onto interface.

Builder Pattern

Name: Wish-list
Problem Description: User is presented with data and must take the same process to see data on each use
Solution: Introduce functionality to be able to store data sets for later retrieval either locally or on a network/server
Consequences: User requires data storage suitable and file I/O issues could arise.

Observer Pattern

Name: Search
Problem Description: User needs to be able to access data from the API using search terms that can be processed into query data.
Solution: Gain unique key for API access to allow interaction and process search term into a query format.
Consequences: Error Handling needed for invalid user input & loss of API functionality

Observer Pattern

Name: Randomize
Problem Description: User is restricted in how to interact with API, retrieving only data they always want.
Solution: Generate random ID and query API(s) using it
Consequences: Error Handling needed for invalid key creation & loss of API functionality

Builder Pattern

Name: User
Problem Description: The system has no way to uniquely identify the current user or what data they had previously saved
Solution: Allow the creation of unique users who will be able to have an individual experience when using the system
Consequences: Security required to ensure data is kept safe

Meeting Logs:

Meeting Date	Attended	Discussion Topics
02/10/2018	Daniel, Sam, Conor, James, Matt	Team formation, GitHub setup
09/10/2018	Daniel, Sam, Conor, James, Matt	UML Diagram Creation
16/10/2018	Daniel, Sam, Conor	Domain Modelling Creation
23/10/2018	Daniel, Sam, James	API Introduction
30/10/2018	Daniel, Conor, James	Sprint Planning, Start Sprint 1
05/11/2018	Daniel, James	Sprint 1, Create logs
06/11/2018	Daniel, Sam, Conor, James, Matt	Presentation, Sprint Review
08/11/2018	Daniel, Conor	Start Sprint 2, Update Logs
13/11/2018	Daniel, Conor	Sprint Review, Sprint Velocity
15/11/2018	Daniel, Conor, Sam	Start Sprint 3
20/11/2018	Daniel, James, Matt	Sprint Review
27/11/2018	Daniel, Sam, Conor, James	Start Sprint 4
02/12/2018	Daniel, Conor, James	Sprint Review, Start Sprint 5
08/12/2018	Daniel, Conor, James	Sprint Review, Scrum Review

Contribution:

Name	ID Number	Main Role/s	Contributions	Comments
Daniel Dixon	16602092	Scrum Master, Development Team	UML, Logging, API Interaction, Prototype GUI, Wishlist, randomisation	Pushed for completion. Started and helped update many of the logs. Assisted with coding.
Conor Wells	14589397	Product Manager, Scrum Support	UML, Logging, Sprint charts, Sprint management, GitHub Organising	Worked to create design elements and organisation of the work being completed.
James Horne	13456827	Development Team, Scrum Support	Coding choice, API Key, Design Patterns	Pushed in final weeks. Assisted with initial coding setup
Matt Woods	14472689	Development Team	UI Design, Allow user input	Assisted with design, issues with attendance
Samuel Cleathero	16603728	Lead Developer	Merge GUI and API	Assisted with coding during the middle of the Scrum

Agile Process Usage:

Process Name	**Comments on Usage**
Sit Together	The Sprint Review constitutes as using the Sit Together
Whole Team	Was used when cross referencing logs and making sprint charts
Informative Workspace	There was no defined team space and so this could not be used
Energized Work	Was not used as would step over the line of student to student interaction, attempts were made however to reduce workload for those contributing the most
Pair Programming	Was used to facilitate coding once underway, helped enormously when attempting to discover errors
Stories	Attempted to create a pseudo-story with the use case diagram but did not fully complete this due to the project being relatively simple and easy to understand what the user needed
Weekly Cycle	The combination of product backlog and sprint reviews created a process similar to the Weekly Cycle in which we could discuss what items had been completed and determine goals for the new sprint
Quarterly Cycle	Was deemed not useful due to the relatively short development window
Slack	Again because of a limited time and due to most items being necessary we never really used this
Ten-Minute Build	The code was generally not that long as to need ten-minutes to build it, once IDEs were set up running the code took seconds
Continuous Integration	Done all the way through development of the code, developers added all software together multiple times every coding session and then ran the code to catch errors
Simple Design	We followed this process to not overwork any member, we met "the current requirements and nothing more" (Sommerville, 2016, 78)

Evaluation:

Scrum artefact reflection:

Using Scrum to complete this artefact has had an enormous effect on how our work was completed and the efficiency of its completion. Using scrum definitely helped to easily discuss improvements and failures in Scrum meetings. Scrum itself is useful for developing a prototype that can be discussed while also still working on the project, offering convenience in a corporate setting by giving a full overview of what is being done. Scrum also allows for versatility as the objectives can be changed after each sprint depending on what was effective or useful. Scrum reviews also provide testing of the product and thus can reduce time needed at the end of development and reducing the cost for the company involved. Using Scrum also comes with some issues, such as the speed at which we were pushed and the participation of all members of the group; when working to create fast paced iterations, if people did not up to meetings (As shown in the meeting logs) or did not complete a task set to them those who were left were left to complete the tasks for the week, to solve this issue the main roles had to be laxed to allow those members who were contributing more to complete as many task as necessary to get the work done. We also had a small amount of scope creep as ACME asked midway through the project to add another API and wish list functionality to our backlog, this is a weakness of scrum as with no rigid scope to work from the product manager can consistently ask for more features, elongating and slowing development time.

Software development methodology comparison and evaluation:

Although Scrum was the main methodology used for this project, in this section we will discuss how it holds up in comparison to other methodologies. These include: XP, Waterfall, and DSDM.

1. Extreme Programming (XP)

   Like the name suggests Extreme programming works by taking tired and tested practices used in the development world and pushing them as far as possible. “In XP, requirements are expressed as scenarios (called user stories), which are implemented directly as a series of tasks” (Sommerville, 2016, 77), as discussed in the XP Processes above we did not create stories but rather a Use Case UML, this meant we didn’t have variation for different users and interactions with the software but we also felt it was unnecessary with how small the development needed was. We did end up using the iteration time frame of XP for sprints (one to two weeks rather than scrum’s two weeks to a month) and the processes specific to XP however which shows that it had merit, especially for shorter tasks like ours where the group also spent a lot of time together when working on the project.

2. Waterfall

   Waterfall is a non-agile method that works by following a sequence of tasks to complete the goal rather than using an iterative process like an agile method would. Because Waterfall is so rigid it allows developers to know what they must do straight away and allows them to plan the totality of their work, this can be seen as better than scrum and other agile methods as they are more focussed on short term goals. However, many companies can’t define the whole project straight away and agile methods allow them a little more choice to update as the development goes on.

3. Dynamic Systems Development Method (DSDM)

   DSDM is another agile framework based on responsible development that was developed to solve issues such as late delivery and high cost that other projects can cause by focussing on user involvement. In comparison to scrum, where features can be added to the backlog at any time and cause both the issues mentioned, this method could have been more useful to incorporate. However, because the project has no defined users and is "Not suitable for small organizations" (Jamsheer K, 2018) we would not get the full use out of this method.

Scrum tools evaluation:

1. Version Control and Data Collation (GitHub)

   GitHub was our main source of controlling both our group data collation and versioning of code, this was because it is widely used by development teams and is very easy to understand. "GitHub is usually the most popular site that hosts a large number of open source projects" (Joy et al 2018) and so it was highly appropriate to use the software. Although it took a few attempts to get working directories running, after this initial setup we had no issues and were able to more efficiently manage resources. GitHub allowed us to split our repository into branches, we used this to separate our sprints out and compartmentalise our completed work, giving us the ability to view our progress as we developed the artefact. Part of the software allows us to initially make changes to a branch and merge it with the master branch, this involved creating a pull request and having another user review the introduced data before aligning it with the master branch. This did slow us down when doing this for every updated item, because all people were working on separate tasks at the same time the pull requests took too long to verify. To solve this problem, we instead completed pull requests to merge the recently completed sprint and master branches at the time of the sprint review.

2. Coding Language

   When choosing how to code the artefact we researched into several coding languages including C++, Java and Python. Initially when discussing C++, we discovered that it has a large number of available libraries and a lot of control due to it being a low-level language. This control does mean that a lot of the security and compiling efficiency is down to the developer and so would drastically decrease development efficiency.

   Java is similar to C++, they both use similar syntax and are Object Oriented languages. Java is a higher-level language than C++ making it easier for the development team to create code quickly. If we are to present the artefact to the ACME, we must make it as functional as possible for as long as possible, Java has very little in the way of backwards compatibility when newer versions are released and so if ACME wishes to keep the code running they have to have further development carried out every year or so (which is costly and resource consuming) or stay with and older version which will become less secure or be unable to interface with newer devices.

   The last language to choose from is Python, this language is hailed as very easy to learn with a large amount of community support and libraries, this would have been needed for our group as none of us had produced anything significant before using the language. Like Java it is a high-level language and is cross compatible but the language itself can suffer from having too many libraries to choose from and being slow for large projects. The drawbacks were not seen as troubling for this project as we easily found the necessary libraries and the project wasn't too large and so Python was chosen as the language for the project.

3. Coding Libraries

   Part of developing the software included using specific libraries both to create a UI and to interact with the API.

   For UI development it was decided that a graphical user interface (GUI) should be implemented to show the information to users clearly and concisely. The three main GUI libraries for Python are Tkinter, Pyqt and WxPython. Although arguably the most aesthetically pleasing and functional of the three languages, Pyqt is known to be the most complicated to use initially, with the turnaround of the project it was considered too difficult. Other issues included complicated licensing to sort through which would need approval from ACME and so we decided against its usage. WxPython and Tkinter are quite similar with syntax and both are easy to use. Although Tkinter uses resources more efficiently and can do more with less code, WxPython has a GUI that is native to the environment whereas Tkinter looks similar across multiple platforms, making Tkinter the least aesthetically pleasing GUI without lots of added extra coding. WxPython was eventually chosen as the right library to fit the project's available time and resources.

   For API development there were also three library choices: Selenium, Requests and Beautiful Soup. Selenium was the initial library taught to the group by the university, it pulls JSON data from the website by opening a webpage. This was the library originally shown to us by the university and worked well to gain the data we needed but we discussed how the webpage opening looked like the program was malicious, we briefly looked into whether the webpage could be opened without displaying on the user's screen but had no success. We then researched using BeautifulSoup which interacts with Requests to show data, this would have allowed us to search specifically through the html to find tags related to the search terms used, unfortunately the Request to the first API only returned raw data and so could not be sifted through. Because Beautiful Soup was non-functional in this situation this led us to just use the Requests library to gain the information and then coding in our own searching function.

4. Messaging Software

   Originally our group attempted to use Discord as the primary messaging software to share information and ideas, we quickly found that using this would not work for a myriad of reasons. When using the software people assign themselves usernames instead of real names, this led to confusion as to who was talking and slowed efficiency. The setup also took a long time as many members of our group did not own the software originally, In the end these burdens led to the group migrating to Facebook messenger; All of the team h ad it and we knew each other by name, this increased efficiency as we could plan meetings much more effectively.

5. Organisation software

   To organise our work to a timetable Trello was used. This software helped us stay focussed on completion dates for our work and when they should be uploaded to GitHub but did not add anything in the way of organising files. GitHub already allowed us to keep everything in order and so Trello was not very useful in that regard. As shown in Appendix 1 Trello was used more in the early stages of development.

6. Office Tools

   The Scrum logs and other text data we created during the development process needed to be stored and updated throughout before their transition to this report, we discussed the use of office tools to complete this and found two main choices; Microsoft Office and

LibreOffice. Both tools offer a wide array of word processing options but Office has quite a few more all with heightened functionality.

7. Extension choice

   To allow for more efficient development, most word processing documents used the extension .odt (Open Text Document) which offers transportability and cross-compatibility, this removed the issues with our work group using different operating systems and office tools as it allowed all documents to be opened anywhere.

Contribution Reflection:

While working on the artefact all members of the group were given different key roles as shown in the Contribution logs. The work produced by each member and the attendance were all to different levels. I was given the position of Scrum master as I had pushed hard to complete work in the early stages before development began and continued to provide support and delegation throughout the project. By creating the log documents and incrementally updating them myself or giving other developers support to improve them I feel I completed the role to a satisfactory standard. Thankfully the project had clear incremental goals and task to complete as without them the project may have struggled, especially when using scrum that has been discussed previously as being bad without clear goals and motivation.

References:

Jackson, C. (2017) *What are the pros and cons vs. C++, Python and Java?*. Available from: https://www.quora.com/What-are-the-pros-and-cons-vs-C++-Python-and-Java [accessed 03 December 2018].

Jamsheer K (2018) *12 Best Software Development Methodologies with Pros & Cons* [blog]. 11 September. Available from: https://acodez.in/12-best-software-development-methodologies-pros-cons/ [accessed 11 December 2018].

Joy, A. Thangavelu, A. Jyotishi, A. (2018) Performance of GitHub Open-Source Software Project: An Empirical Analysis. In: *2018 Second International Conference on Advances in Electronics, Computers and Communications (ICAECC)*, Bangalore, India 9-10 February. New York, IEEE, 1-6.

Kirill Fakhroutdinov (undated) *The Unified Modeling Language*. Available from: https://www.uml-diagrams.org/ [accessed 30 November 2018].

Prechelt, L. (1999) Comparing Java vs. C/C++ Efficiency Differences to Interpersonal Differences. *Communications of the ACM*, 42(10), 109-114.

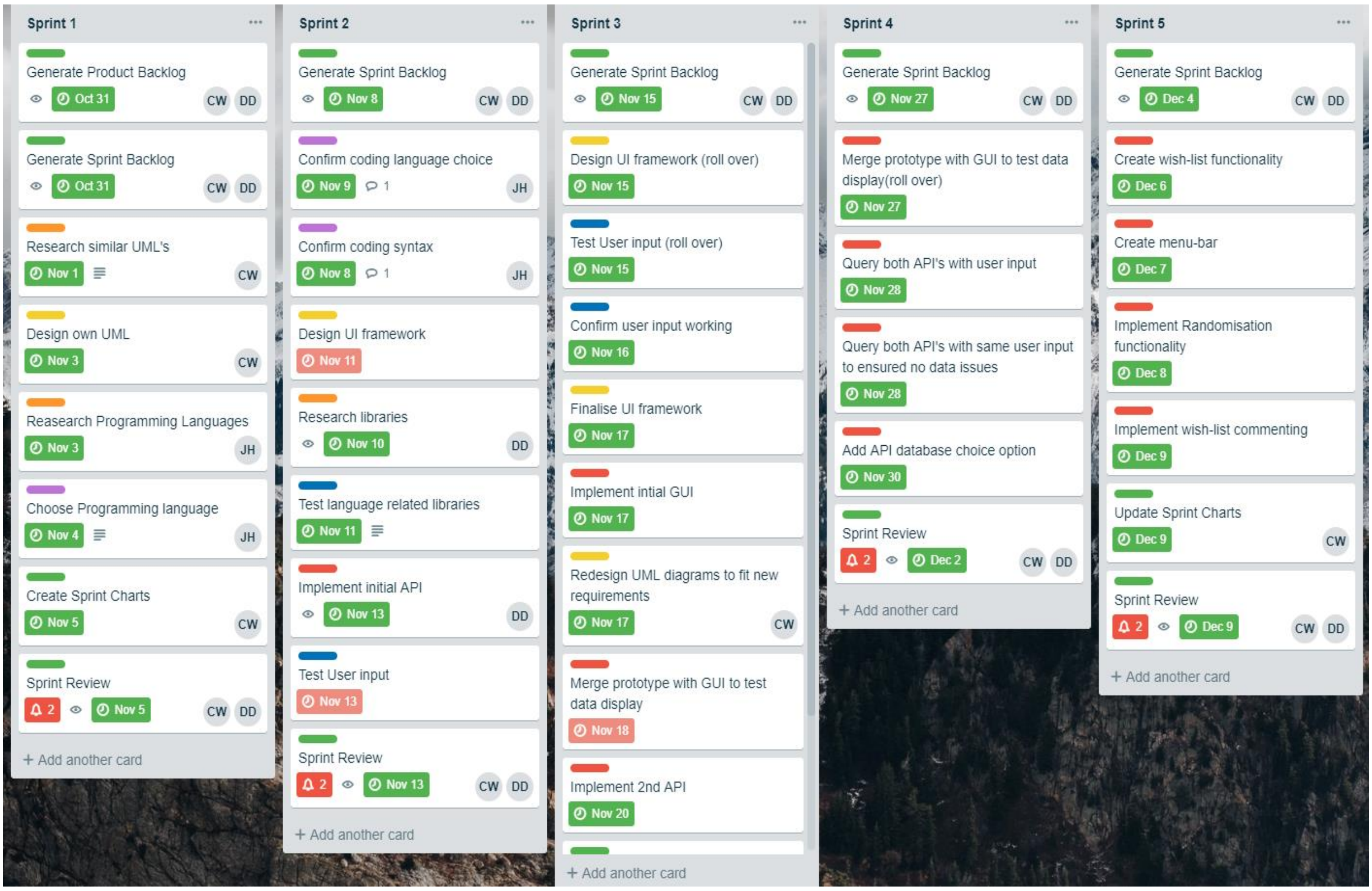
Riyu, B. and Anuja, A. (2018) Influence Indexing of Developers, Repositories, Technologies and Programming Languages on Social Coding Community GitHub. In: *2018 Eleventh International Conference on Contemporary Computing (IC3)*, Noida, India, 2-4 August. New York, USA: IEEE.

Rubin, K.S. (2013) *Essential Scrum: a practical guide to the most popular Agile process*, Boston: Addison-Wesley.

Sommerville, I. (2016) *Software Engineering*. Boston: Pearson Education Limited.

Appendices:

Appendix 1: Trello main page



Sprint 1

- Generate Product Backlog — Oct 31 — CW DD
- Generate Sprint Backlog — Oct 31 — CW DD
- Research similar UML's — Nov 1 — CW
- Design own UML — Nov 3 — CW
- Reasearch Programming Languages — Nov 3 — JH
- Choose Programming language — Nov 4 — JH
- Create Sprint Charts — Nov 5 — CW
- Sprint Review — 2 — Nov 5 — CW DD

+ Add another card

Sprint 2

- Generate Sprint Backlog — Nov 8 — CW DD
- Confirm coding language choice — Nov 9 — 1 — JH
- Confirm coding syntax — Nov 8 — 1 — JH
- Design UI framework — Nov 11
- Research libraries — Nov 10 — DD
- Test language related libraries — Nov 11
- Implement initial API — Nov 13 — DD
- Test User input — Nov 13
- Sprint Review — 2 — Nov 13 — CW DD

+ Add another card

Sprint 3

- Generate Sprint Backlog — Nov 15 — CW DD
- Design UI framework (roll over) — Nov 15
- Test User input (roll over) — Nov 15
- Confirm user input working — Nov 16
- Finalise UI framework — Nov 17
- Implement intial GUI — Nov 17
- Redesign UML diagrams to fit new requirements — Nov 17 — CW
- Merge prototype with GUI to test data display — Nov 18
- Implement 2nd API — Nov 20

+ Add another card

Sprint 4

- Generate Sprint Backlog — Nov 27 — CW DD
- Merge prototype with GUI to test data display(roll over) — Nov 27
- Query both API's with user input — Nov 28
- Query both API's with same user input to ensured no data issues — Nov 28
- Add API database choice option — Nov 30
- Sprint Review — 2 — Dec 2 — CW DD

+ Add another card

Sprint 5

- Generate Sprint Backlog — Dec 4 — CW DD
- Create wish-list functionality — Dec 6
- Create menu-bar — Dec 7
- Implement Randomisation functionality — Dec 8
- Implement wish-list commenting — Dec 9
- Update Sprint Charts — Dec 9 — CW
- Sprint Review — 2 — Dec 9 — CW DD

+ Add another card