Abstract

Documented evidence of Arbnor Aljilji implementing and demonstrating the KSBs.

Dev Ops Engineer Apprenticeship

T. Rowe Price Portfolio

Arbnor Aljilji

Contents

[**Introduction to myself, team, and project** 4](#_Toc93411653)

[Health and Safety 5](#_Toc93411654)

[Data Protection 5](#_Toc93411655)

[Knowledge 6](#_Toc93411656)

[**K1**: Continuous Integration - the benefits of frequent merging of code, the creation of build artefacts and ensuring all tests pass, with automation throughout - including common tooling. 6](#_Toc93411657)

[**K2**: The principles of distributed Source Control, including how to exploit the features of the tool, such as branching 6](#_Toc93411658)

[**K3**: How to use data ethically and the implications for wider society, with respect to the use of data, automation and artificial intelligence within the context of relevant data protection policy and legalisation 7](#_Toc93411659)

[**K4**: The business value of DevOps in terms of Time, Cost, Quality, with an emphasis on building in internal Quality throughout the lifetime of the product. 7](#_Toc93411660)

[**K5**: A range of modern security tools and techniques – e.g. threat modelling, vulnerability scanning and dependency checking with a general awareness of penetration testing – in order to deal with threats and attack vectors within code and across cyber domain 7](#_Toc93411661)

[**K6**: A range of problem-solving techniques appropriate to the task at hand, such as affinity mapping, impact maps, plan-do-check-act/Deming 7](#_Toc93411662)

[**K7**: General purpose programming and infrastructure-as-code. 9](#_Toc93411663)

[**K8**: Immutable infrastructure and how it enables continuous refreshing of software, namely updating of the operating system, container and security patching 9](#_Toc93411664)

[**K9**: Different organisational cultures, the development frameworks utilized and how they can both complement each other and introduce constraints on delivery 9](#_Toc93411665)

[**K10**: How the user experience sits at the heart of modern development practices in terms of strategies to understand diverse user needs, accessibility and how to drive adoption 9](#_Toc93411666)

[**K11:** Monitoring and alerting technologies and an awareness of the insights that can be derived from the infrastructure and applications – collecting logs and metrics, configuring alerting thresholds, firing alerts and visualizing data 9](#_Toc93411667)

[**K12**: The persistence/data layer, including which database/storage technologies are appropriate to each platform type and application when considering non-functional and functional needs; e.g. monolith, microservice, read heavy, write heavy, recovery plans 10](#_Toc93411668)

[**K13:** Automation techniques, such as scripting and use of APIs 10](#_Toc93411669)

[**K14:** Test Driven Development and the Test Pyramid. How the practice is underpinned by unit testing, the importance of automation, appropriate use of test doubles and mocking strategies, reducing a reliance on end-to-end testing 10](#_Toc93411670)

[**K15**: The principles and application of Continuous Integration, Continuous Delivery and Continuous Deployment, including the differences between them 10](#_Toc93411671)

[**K16:** How best to secure data; e.g. encryption in transit, encryption at rest and access control list (ACL) 10](#_Toc93411672)

[**K17:** What an API is, how to find them and interpret the accompanying documentation 10](#_Toc93411673)

[**K18:** Roles within a multidisciplinary team and interfaces with other areas of an organisation 10](#_Toc93411674)

[**K19:** Different methods of communication and choosing the appropriate one – e.g. face-to-face (synchronous, high bandwidth), instant messaging, email (asynchronous, low bandwidth), visualisations vs. words 10](#_Toc93411675)

[**K20:** Pair/mob programming techniques and when to use each technique 11](#_Toc93411676)

[**K21:** Architecture principles, common patterns and common strategies for translating user needs into both cloud infrastructure and application code 11](#_Toc93411677)

[**K22:** How their occupation fits into the wilder digital landscape and any current or future regulatory requirements 11](#_Toc93411678)

[**K23:** The importance of continual improvement within a blameless culture 11](#_Toc93411679)

[**K24:** The difference between Software-as-a-Service (SaaS) v bespoke v enterprise tooling and how to make an informed choice that suits each use case 11](#_Toc93411680)

[**K25:** Maintain an awareness of cloud certification requirements 11](#_Toc93411681)

[**Skills** 12](#_Toc93411682)

[S1 – Communicate credibly with technical and non-technical people at all levels, using a range of methods; e.g “Show and Tell” and “Demonstrations” 12](#_Toc93411683)

[S2 – Work with different organisational cultures with both internal and external parties 12](#_Toc93411684)

[S3 – Translate user needs into deliverable tasks, writing clear, concise and unambiguous user stories that the whole team can understand 14](#_Toc93411685)

[S4 – Initiate and facilitate knowledge sharing and technical collaboration 14](#_Toc93411686)

[S6 – Install, manage and troubleshoot monitoring tools 16](#_Toc93411687)

[S11 – Employ a systematic approach to solving problems, using logic and hypotheses / experimentation to identify the source of issues 17](#_Toc93411688)

[S12 – Automate tasks where it introduces improvements to the efficiency of business processes and reduces waste, considering the effort and cost of automation 18](#_Toc93411689)

[S13 – Engage in productive pair/mob programming 19](#_Toc93411690)

[S14 – Write tests and follow Test Driven Development discipline in various contexts 20](#_Toc93411691)

[S16 – Invest in continuous learning, both your own development and others, ensuring learning activities dovetail with changing job requirements. Keeping up with cutting edge 22](#_Toc93411692)

[S17 – Code in a general-purpose programming language 23](#_Toc93411693)

[**Behaviours** 25](#_Toc93411694)

[B1 – Exhibits enthusiasm, openness and an aptitude for working as part of a collaborative community; e.g. sharing best practice, pairing with team members, learning from others and engaging in peer view practices 25](#_Toc93411695)

[B3 – Displays a commitment to the mantra “You build it, you run it”, taking ownership of deployed code and being accountable for its continual improvement, learning from experience and taking collective responsibility when things fall 25](#_Toc93411696)

# **Introduction to myself, team, and project**

My name is Arbnor Aljilji and I’m part of the DevOps Engineering Apprenticeship at T. Rowe Price. I’m 28 years old and chosen to switch careers from Human Resources to Tech. I first studied BTEC IT in college and graduated in BA Hons Human Resources but I always had an interest in tech; AI, to electric vehicles, to gaming development, and my interest in technology increased heavily throughout the pandemic. I had a lot of free time to learn programming languages and creating very small math quizzes with codecademy etc. However, I didn’t want to tackle software engineering directly as technology itself is too big to know what I want to progress into so I researched the different paths a new ‘self-taught’ developer can go into. There were different options from Cybersecurity, Software Engineer, Cloud Engineer (DevOps), Data Analyst etc. After doing research and comparing the pros and cons between the different roles, DevOps itself stood out more to me as I’m also fascinated with Cloud Computing and the use of cloud computing is still increasing.

I am part of the T. Rowe Price Developer Services Group in London. We are a team of 9 Software Engineers, and 1 Product Manager that have different specialty skills as developers, SDETS and SREs. Our team looks after the developers who design apps, and our mission is to enable the developers to move as quickly as possible and enable them to push cleaner code at great frequency, and higher volume. Through automation around development tool chain, release management, software quality, builds, and developments, we aim to achieve these goals. Unity is an in-house product designed and looked after by the DSG developers to allow other developers to deploy their application to the cloud, at a much quicker pace.

The project I am currently working on is a chatbot designed to automate any toil that our developers may experience. It will be available to only the DSG London and US team, which is a group of 20 engineers until further expansion. The chatbot itself will handle messages and requests from clients and respond back with an answer or resource. Many of my clients have repeated manual tasks such as opening a browser, writing a link to check a specific merge requests, whereas the bot can notify the user with the merge request link upon request and the status of the merge request. There are other designed functions that myself and a project lead are working on for the bot to complete, as this will remove any toil the team undergoes. The value this chatbot project will bring to the team is by saving themselves couple minutes a day, to several hours a week in total of typing and clicking in which the bot can perform at a higher rate with a simple question. As well as this project, the team and myself also provide Linux Desktop Support for developers.

A normal week at TRP for myself would be defined as such:

* Every Monday morning at 9.30am I have an hour meeting with my line manager. We plan the week on what tasks I am working on, and what I did in the previous week
* Every Friday morning at 10am I have a team meeting with the Developer Services Group (my team’s department) to discuss what support issues we have faced in Linux Developer Desktop for the week, and any other topic (extra responsibility the team does)
* Tuesday and Wednesday I set up an hour or more meeting my tech lead, to discuss the UnityBot project. We plan on what to achieve next, what we have done so far, and what blockers we are on. I write this down as comments in my Jira tickets and note pad.
* I also study AWS SAA using AClougGuru
* Study on Udemy/YouTube for NodeJS, JS, Bash, Dev Tools, and Linux
* Meetings with other departments in Global Tech (Enterprise Security, Cyber Security, Tech Engineering, Demo Days, Cohort Social, Code Club, Town Halls)

We have work assigned via tickets in which the team works through, for example, if a client has an idea for a product we own, they would create a question/user request on AskTrusty (similar StackOverflow concept) and then a ‘feature’ ticket is raised for the team of the app to implement a new feature to the application. Another way of receiving tickets is when our clients have issues with their LDD, and the DSG team supports them from 9am-2pm, in which support is then handed over to the US team. Unity Folio, which is an internal app designed to make the SDLC a smoother operation for many developers. This app allows developers to upload their app into their cloud by themselves and can be done via pipeline or the UI. The focus is to improve the automation of the procedure, to minimize the work time to upload your work and automate any procedures into deploying their apps and maintaining them. Lastly, my project sits under one main ticket ‘Unity Boy’ with sub-tickets underneath to define each task I will be working on monthly. This is how I and the team operate.

# Health and Safety

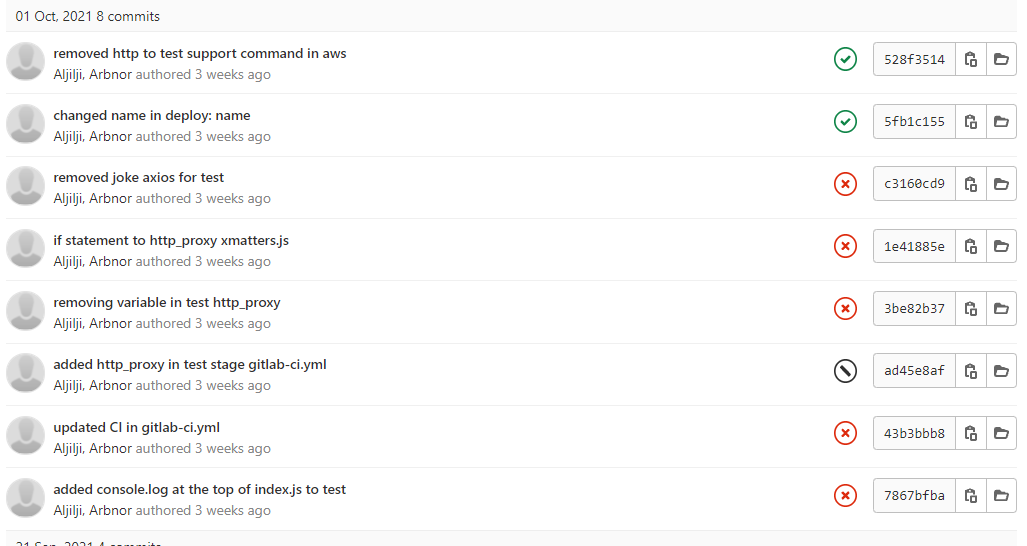
Have done the health and safety training at TRP and signed off.

# Data Protection

There are confidentiality regulations and policies I would have to follow before uploading my work

# Knowledge

## **K1**: Continuous Integration - the benefits of frequent merging of code, the creation of build artefacts and ensuring all tests pass, with automation throughout - including common tooling.



Above is a screenshot of frequent merging of code. The benefit of doing this is that with smaller code changes, it will be simpler and easier to handle rather than huge amount of code, and as such, will have fewer issues that may need to be fixed in the later stage. With smaller merged code, you could test as soon as they are integrated, which will highlight any errors before too much work is completed after. More points cover such as: fault isolations, Faster Mean Time to Resolution, More Test Reliability, Quicker Release, Less Backlog, Customer Satisfaction, Reduce Cost, Easy Maintenance and updates.

## **K2**: The principles of distributed Source Control, including how to exploit the features of the tool, such as branching

Source Control has many advantages: history, traceability, versioning, collaborative and parallel development to name a few. Having a centralized project repo, people can collaborate in the project and contribute to it, by creating their own branch and merging into main. Git itself has many commands such as pull, push, merge, clone, checkout etc. By having merge requests, users can review code change before it gets pushed into the main code base. With many people working on a project, people can look back at the history of the project to see how it has developed from 1.0/0.1 (depending how your team version it). In UnityBot project, it starts at version 1.0.0 and updates with every new production change it will go through. If there are any issues in the project which are caused in a newer version, teams would be able to go back and see what the code change difference is to see where it is broken. Nonetheless, you could also trace back on who made any code changes and merged onto main. Feature tools such as branching is a feature available in most modern version control systems. In Git, this is part of the everyday development process. Branching helps with taking a snapshot of the current code and applying additional code without changing the main code. For example, when working on a new feature or fixing a bug – no matter how big or small – you would make a branch to encapsulate your changes. This makes it safer for unstable code to get merged into the main code base, and I would be able to clean up any code before merging it into the main branch.

## **K3**: How to use data ethically and the implications for wider society, with respect to the use of data, automation and artificial intelligence within the context of relevant data protection policy and legalisation

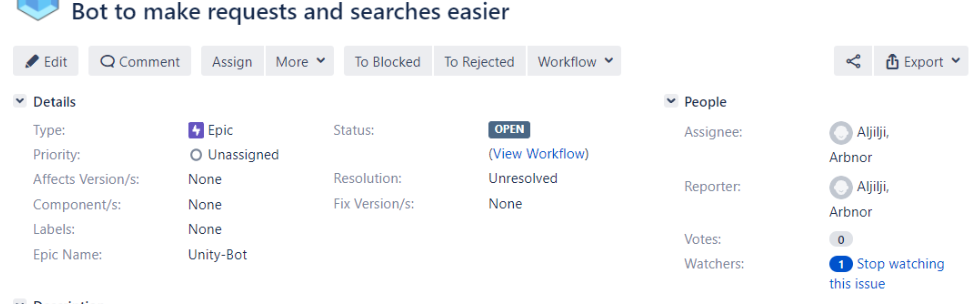
## **K4**: The business value of DevOps in terms of Time, Cost, Quality, with an emphasis on building in internal Quality throughout the lifetime of the product.

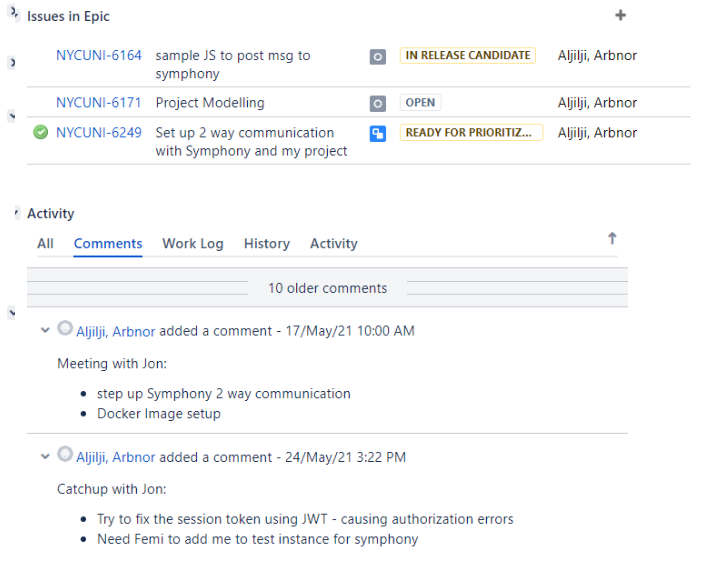
As mentioned in K1, I mentioned further importance of business value of DevOps in terms of time, Cost, and Quality. There several remaining business values could be such as it can reduce cost and easy to maintain and update. For example, automation in the CI/CD pipeline reduces the number of errors that can take place in many repetitive steps. This gives the coder free time to spend it on product development, as there isn’t much code change errors, nor the tests failing or picking up any errors quickly. The quality of the product also improves as testing on small code changes can pick up any further bugs found, thus, making it more of a flawless product to meet the customer satisfaction.

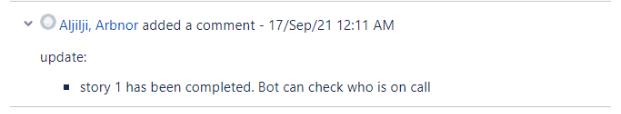
## **K5**: A range of modern security tools and techniques – e.g. threat modelling, vulnerability scanning and dependency checking with a general awareness of penetration testing – in order to deal with threats and attack vectors within code and across cyber domain

## **K6**: A range of problem-solving techniques appropriate to the task at hand, such as affinity mapping, impact maps, plan-do-check-act/Deming

Even though I understand the use of affinity mapping, I take a different approach. In my project, we use Jira and create tasks which handles the task needed to be completed, the planning and issues facing. For example, this Epic story below contains other tasks within the main task itself ‘bot to make requests and searches easier’ – so my team know it is a bot design task. However, within the Jira ticket, we can assign new issues within the Epic story, such as setting up the 2- way communication. Unlike using affinity mapping, I write all the completed tasks, and difficult tasks in the comment sections within the main epic story, as seen below. This only gives a clear outline that part of my project is completed, and I can now move into creating a new sub-task within that main task. This method of problem-solving helps me understand what I need to do in my project, what I am missing and what I have completed.

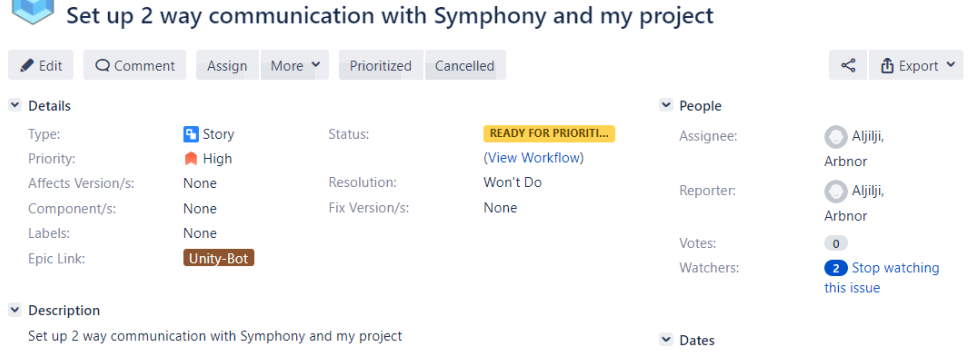




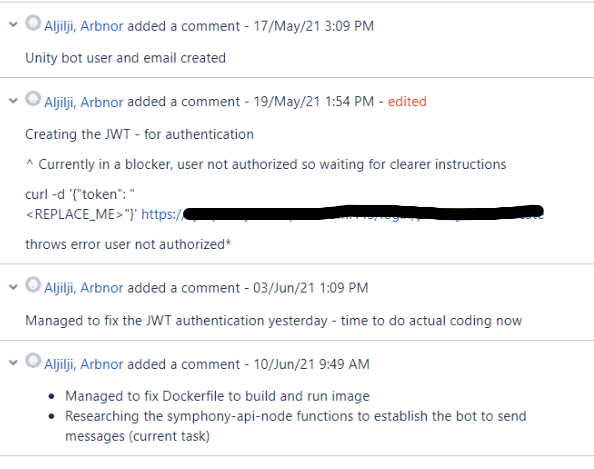


(Below is continued K6)

The screenshot below will show me clicking into NYCUNI-6249 which is an issue within the Epic story.



Now within this task, I know what I must do as I have created a task for it, and the issues I am currently facing, as I have inserted it in the comment section underneath.



## **K7**: General purpose programming and infrastructure-as-code.

## **K8**: Immutable infrastructure and how it enables continuous refreshing of software, namely updating of the operating system, container and security patching

## **K9**: Different organisational cultures, the development frameworks utilized and how they can both complement each other and introduce constraints on delivery

## **K10**: How the user experience sits at the heart of modern development practices in terms of strategies to understand diverse user needs, accessibility and how to drive adoption

## **K11:** Monitoring and alerting technologies and an awareness of the insights that can be derived from the infrastructure and applications – collecting logs and metrics, configuring alerting thresholds, firing alerts and visualizing data

## **K12**: The persistence/data layer, including which database/storage technologies are appropriate to each platform type and application when considering non-functional and functional needs; e.g. monolith, microservice, read heavy, write heavy, recovery plans

## **K13:** Automation techniques, such as scripting and use of APIs

## **K14:** Test Driven Development and the Test Pyramid. How the practice is underpinned by unit testing, the importance of automation, appropriate use of test doubles and mocking strategies, reducing a reliance on end-to-end testing

## **K15**: The principles and application of Continuous Integration, Continuous Delivery and Continuous Deployment, including the differences between them

## **K16:** How best to secure data; e.g. encryption in transit, encryption at rest and access control list (ACL)

## **K17:** What an API is, how to find them and interpret the accompanying documentation

## **K18:** Roles within a multidisciplinary team and interfaces with other areas of an organisation

## **K19:** Different methods of communication and choosing the appropriate one – e.g. face-to-face (synchronous, high bandwidth), instant messaging, email (asynchronous, low bandwidth), visualisations vs. words

Every week I have a 9.30am meeting for 1 hour with my manager on my weekly catchup. Discussion covers the project, current tasks and potential future tasks, any blockage I have, and other activities I have been working on. Later during the week on Friday, the whole DSG London Team have a stand up at 10am to discuss support that we have accomplished on Linux Dev Desktop throughout the week – any new unknown errors, or what complications occurred on support for the week. I also use Symphony (like slack) a lot to do one-on-one messaging, which then escalates to me inviting the other user to Zoom. When it comes to learning new skills or knowledge at work, I set up Zoom meetings as I prefer learning visually, as unknown jargon can confuse me, or the internal document may be outdated.

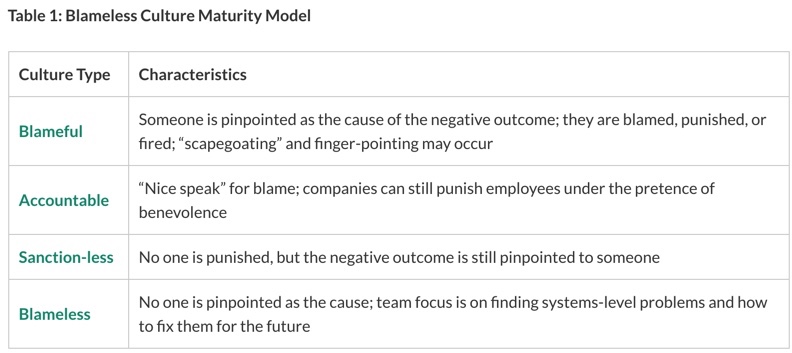
## **K20:** Pair/mob programming techniques and when to use each technique

## **K21:** Architecture principles, common patterns and common strategies for translating user needs into both cloud infrastructure and application code

## **K22:** How their occupation fits into the wilder digital landscape and any current or future regulatory requirements

## **K23:** The importance of continual improvement within a blameless culture

Within an organisation, I find that there are different tiers of ‘blame culture’. Below is the Blameless Culture Maturity Model:



The idea of a blameless culture helps use find solutions, rather than putting blame on an individual. Individual contributors feel safer, eagerly report failures and rather seek solutions at an earlier step, than leaving it too late or creating scape goats. By continually improving, we get less errors occurring, stronger team bonding, and people feel more motivated to do work.

## **K24:** The difference between Software-as-a-Service (SaaS) v bespoke v enterprise tooling and how to make an informed choice that suits each use case

## **K25:** Maintain an awareness of cloud certification requirements

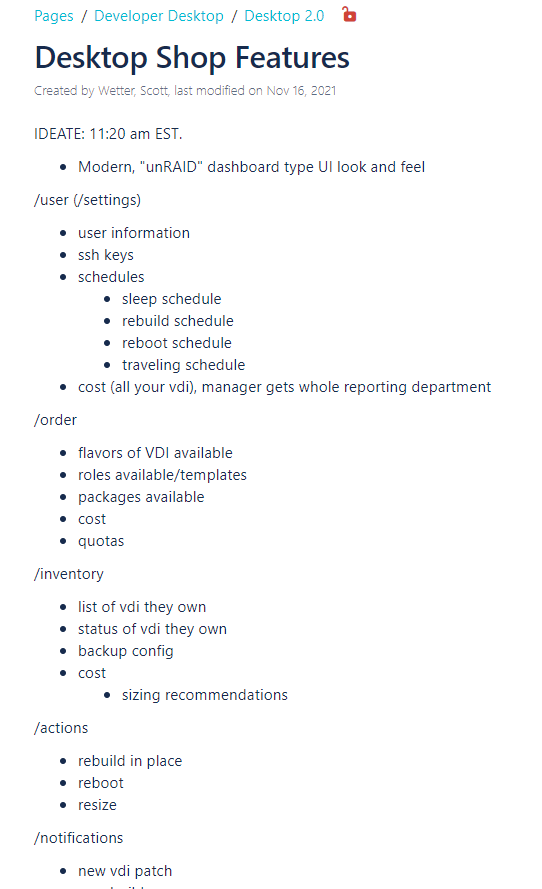
I look at different cloud providers certificates, and currently slowly studying the AWS SAA (recommended) – see S16

# **Skills**

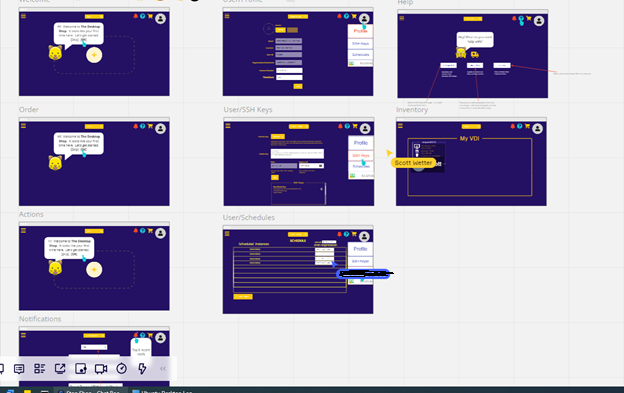
## S1 – Communicate credibly with technical and non-technical people at all levels, using a range of methods; e.g “Show and Tell” and “Demonstrations”

Every two months or so, the team have a Show and Tell in our calendar which we highlight to our team what we have achieved in our projects, and what we want to do. As our team is all technical, I would be able to use the jargon they would understand. However, I am also part of a Cohort of new members in the firm which consists of non-technical employees. I was able to demonstrate my project using non-technical descriptions of what my project is, how it works and the benefits of the project to the clients. The idea of demonstrating our projects to each other in the cohort was my idea to the cohort leader, as it makes us build a stronger network, but also take interest in each other’s work and provide feedback.

## S2 – Work with different organisational cultures with both internal and external parties

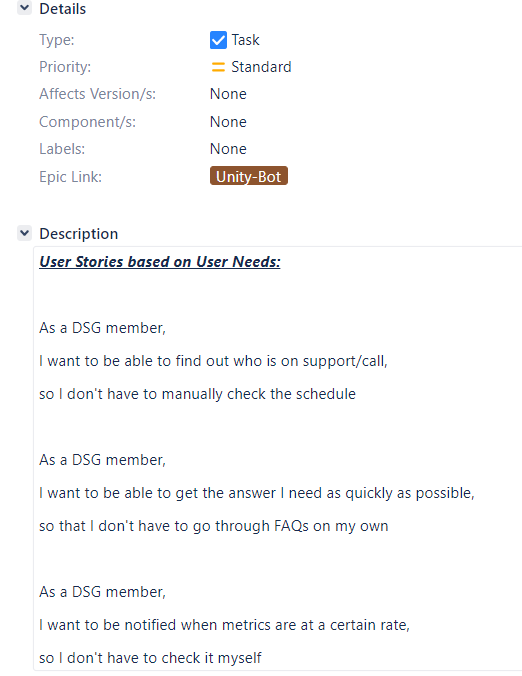


For this project, I worked with a different team outside my usual London team. They were based in America, and it meant adapting to the time zone differences and working culture. The project came to live as an idea from a hackathon to which we were all able to decide which project catches out attention. I thought this was a good project to be involved in as it meant I was able to do some front-end coding – which I rarely do, and it would be cool to learn. We done a 2-day sprint which meant on the first day we mapped out what our MVP would look like, and what we want it to cover – which is seen above. I took in charge of just directly designing the webpage layouts with my manager to mock it up. Below is the outcome.



We used the Miro board to mock these up, which is a new tool for me. The team and I didn’t use any tickets

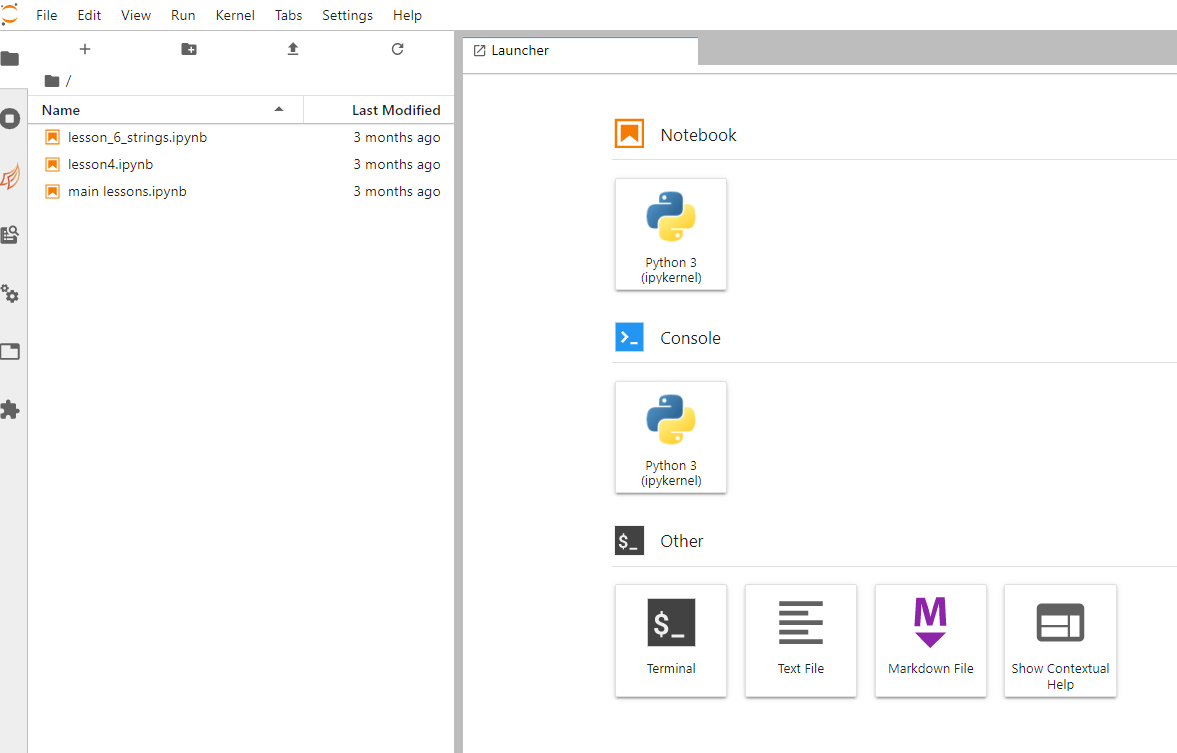
## S3 – Translate user needs into deliverable tasks, writing clear, concise and unambiguous user stories that the whole team can understand



Above is a picture of the User Stories that I have written from the user needs in a Jira ticket. All 3 user stories will do some sort of api call, and update users with certain notifications. However, these are the basic user needs and there are future updates that will be made. I got this ticket by discussing with the clients what the user needs are around the Symphony Bot, and then created a jira ticket to implement them. I didn’t really have to do much research on what a user story is due to my bootcamp training, so I sort of understood how to write them follow the 3 step guide of who the client is, what they want to achieve, and why they want to do it. I did this work by myself and then spoke to my tech lead to ensure that I was designing the user stories correctly.

## S4 – Initiate and facilitate knowledge sharing and technical collaboration

Throughout my time at TRP, I was part of the Code Club Beginners EMEA cohort. I was able to learn the very basics of Python. After finishing the in-house bootcamp, I felt like I wanted to teach non-technical and technical colleagues of TRP on how to code too. The reason I felt comfortable with this is due to increase of confidence within the firm, and it helped me understand certain algorithms better. I was able to have a collaboration with both non-tech people, and the tech team. I tried to get as much ‘students’ involved with the cohort and get as many coaches as possible to teach. The coaches and I taught students individually 1 on 1 over zoom and created many fun data visualization (graphs etc). Alongside this, I create documents for myself and other members to follow when we have to do a certain task. For example, how one can ssh into someone’s VDI and rejoin them back to DNS due to dropout. This helped us finish some support tasks that we had undertaken.



Above is the screenshot of the lessons I taught using Jupyter notebook



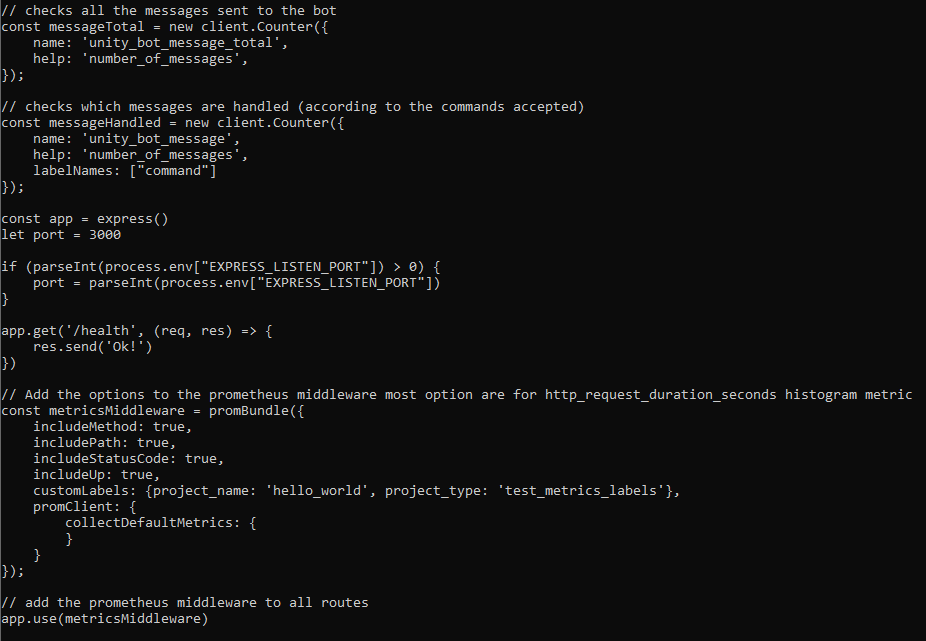
Here is a taster of the coding exercises. On the lessons I got stuck with, I let the senior engineers take over only so I don’t mislead the members learning such as using Pandas. I actually learnt a lot during this Cohort, not just Python, but that I can be confident enough to learn something new, and try out new experiences even if they seem hard at first. I can collaborate with people from different departments even if they are not technical.

**Notice**: For confidentially reasons, I cannot show the document pages I made for the “sshing into vdis”.

S5 – Deploy immutable infrastructure

## S6 – Install, manage and troubleshoot monitoring tools

On my project, I’m currently using Prometheus-client for NodeJS to set up metrics regarding my project. Alongside this, I’m using Express to create a simple minimalist web frame to which I can access the metrics. Below is a snippet of my code to which Prometheus is setup and Express.



The advantage of having metrics is so I can see if my service (bot) is working the way it should be. For starters, I’m looking at what commands the clients are using, what incorrect commands are being placed, and further metrics we will implement is how many errors are being thrown to clients due to the downfall of other services, for example, xmatters being down.

S7 – Navigate and troubleshoot stateful distributed systems, in order to locate across the end-to-end service

S8 – Work in agile, multi-disciplinary delivery teams, taking a flexible, collaborative and pragmatic approach to delivering tasks

S9 – Application of a range of cloud security tools and techniques – e.g. threat modelling, vulnerability scanning, dependency checking, reducing attack surface area – incorporating these tools and techniques into the automated pipeline whenever possible

S10 – Assess identified and potential security threats and take appropriate action based on likelihood v impact

## S11 – Employ a systematic approach to solving problems, using logic and hypotheses / experimentation to identify the source of issues

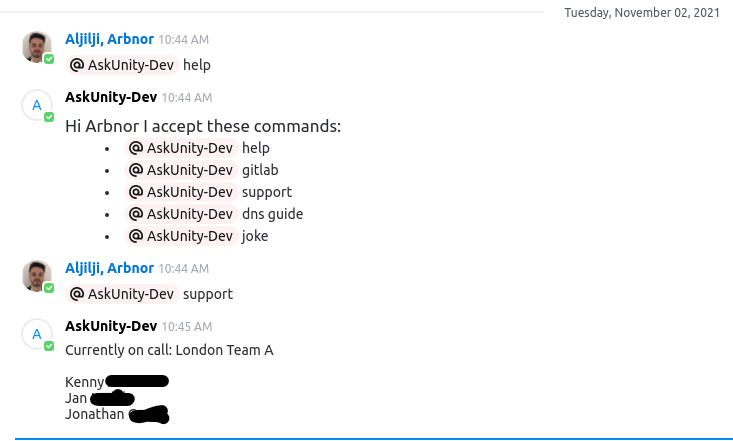
When it comes to employing a systematic approach to solving problems, I tend to follow a 6-step rule which I have managed to research and break down and implement in my daily tasks at TRP. This 6-step process is seen below:

* Identify the problem
* Contain the problem
* Investigate
* Cause analysis
* Improvement
* Closure

This process at the beginning was hard to implement due to having not a lot of knowledge in the Technology industry. However, once I started to learn more, I was able to make evaluations based on previous experience. This allowed me to act quicker, and more efficient to problem-solving. For example, when I first done an API call from my bot to Symphony, I couldn’t get a respond back to Symphony from xmatters due to Symphony sitting with some proxy settings. I didn’t realise why it wasn’t working, as there were no errors appearing in any logs. I decided to contain the problem and discuss it further with my tech lead. I learnt that due to security issues, we cannot just allow any traffic to come into our network without ensuring its safe, which is why we set up the proxy settings after. This sort of challenge and problem-solving has helped me think differently and act differently on when we have issues.

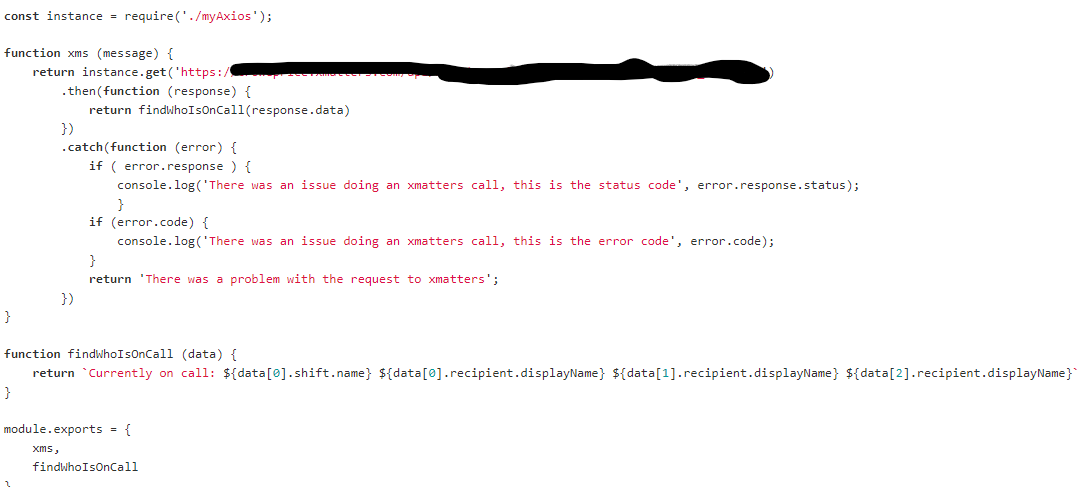
## S12 – Automate tasks where it introduces improvements to the efficiency of business processes and reduces waste, considering the effort and cost of automation

Currently, the bot does the xmatters API call, but there will be new features implementing from Feb-April as it will be pushed to Production this month. I have managed to solve a manual task, removing the toil of logging into an internal website, and look at the rota which can be quite an annoying, repeated process. I have screenshotted below the bot being able to do an API call to xmatters and doing a GET request of the current team on shift. I came across this task as it was one of my user stories in my project to which I used Jira to create the ticket. This user story was created by listening and understanding the client needs.

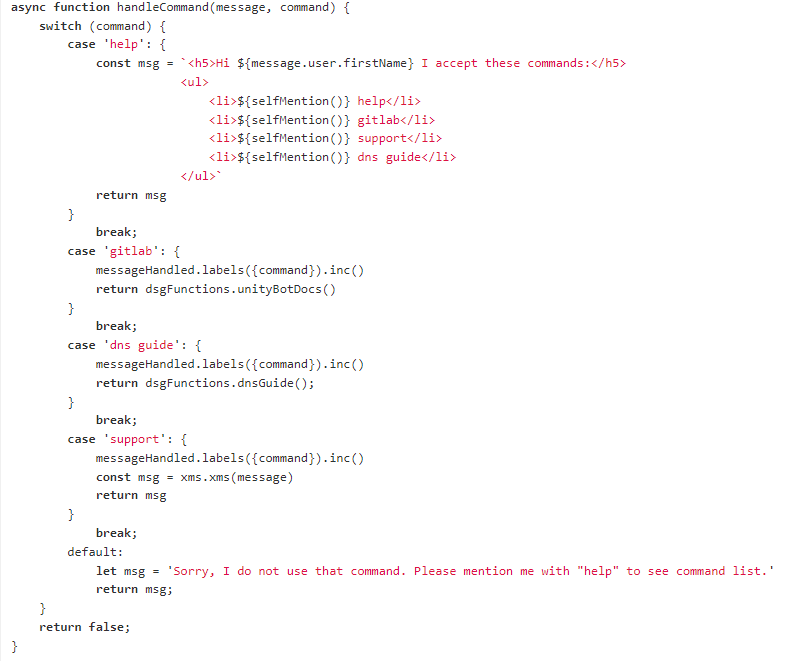


## S13 – Engage in productive pair/mob programming

There is no ticket needed to this skill. Pair programming usually occurs whenever me and my tech lead have a meeting set up to refactor the code, or when I am lost at understanding a concept and need it introduced. I set up the meetings on Zoom early mornings as we find that’s best practice for both of us. The concepts I found difficult to understand at the time was the asynchronous code in NodeJS and callbacks/promises. I understood the concept of it but wasn’t sure how to write the code to do the api call to xmatters. I had researched ES6 and many other things which helped me understand, but I still had some difficulty working around it which is why we had done pair programming.



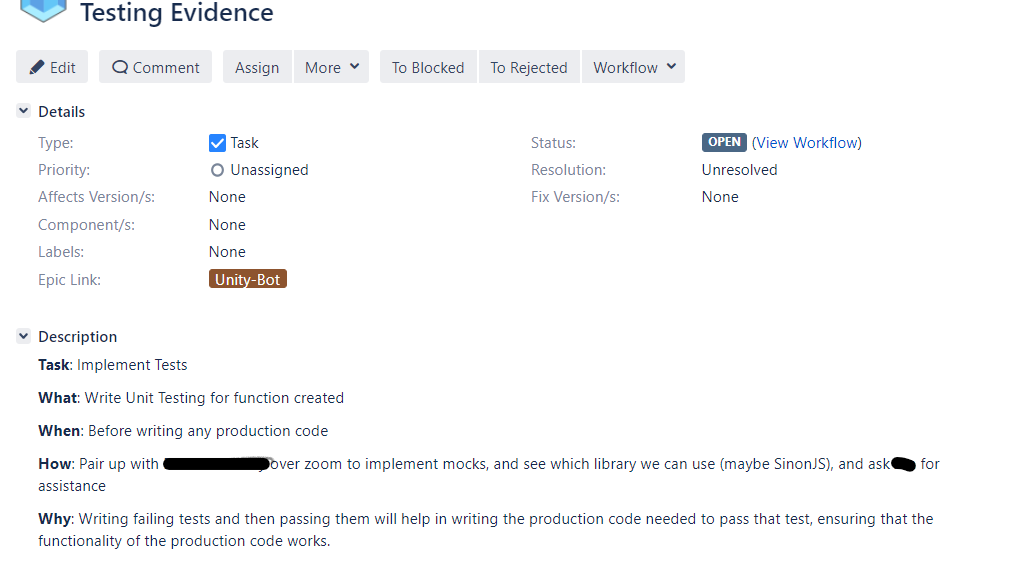
Above is the xms function that we both coded to do the get request api call to xmatters.



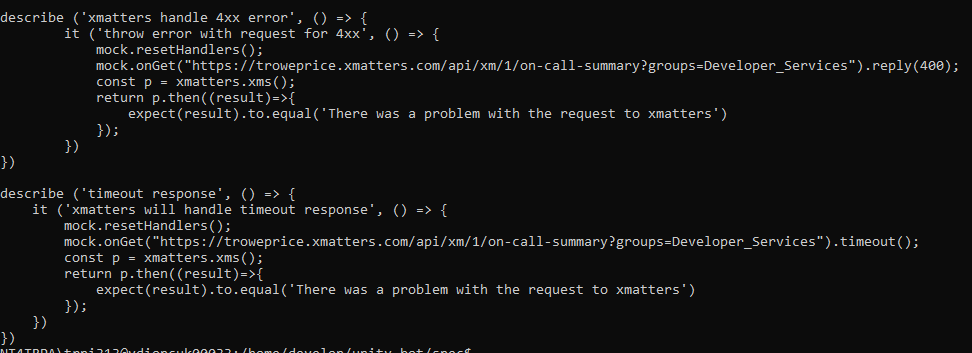
Above is the async function which I couldn’t understand on why it’s async, but I do now. From this pair programming, I learnt what asynchronous coding is vs synchronous, what switch statement is compared to using if else statements. The impact this had on my code is that it’s neater for other users to understand, and it’s more adaptable for others to contribute and add their own ‘case’, so by making it easier, it’s more usable.

## S14 – Write tests and follow Test Driven Development discipline in various contexts

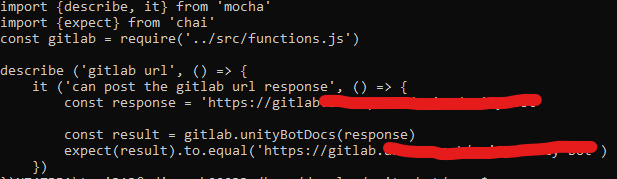
I have included this skill as a ticket due to the nature of TDD being important in a project.



I created this ticket as one of the main priorities in my project as there are many benefits to testing before writing code. Many of our tickets are either assigned to us, but as I own this project, I have assigned this ticket to myself. I have had to research TDD quite a lot, only due to the nature of forgetting how to do TDD with more advanced coding concepts. For example, if I was to test a calculator and return a value, it might be easier than trying to mock test a 3rd party client/library etc. As I was using axios library to do HTTP client requests, I used axios-mock-adapter to create mocks of the xmatters data. **Due to the nature of protection, I cannot post all the tests**. However, I have tested many different test types, for example, status codes, whether a function returns the value intended etc. Two examples are below:

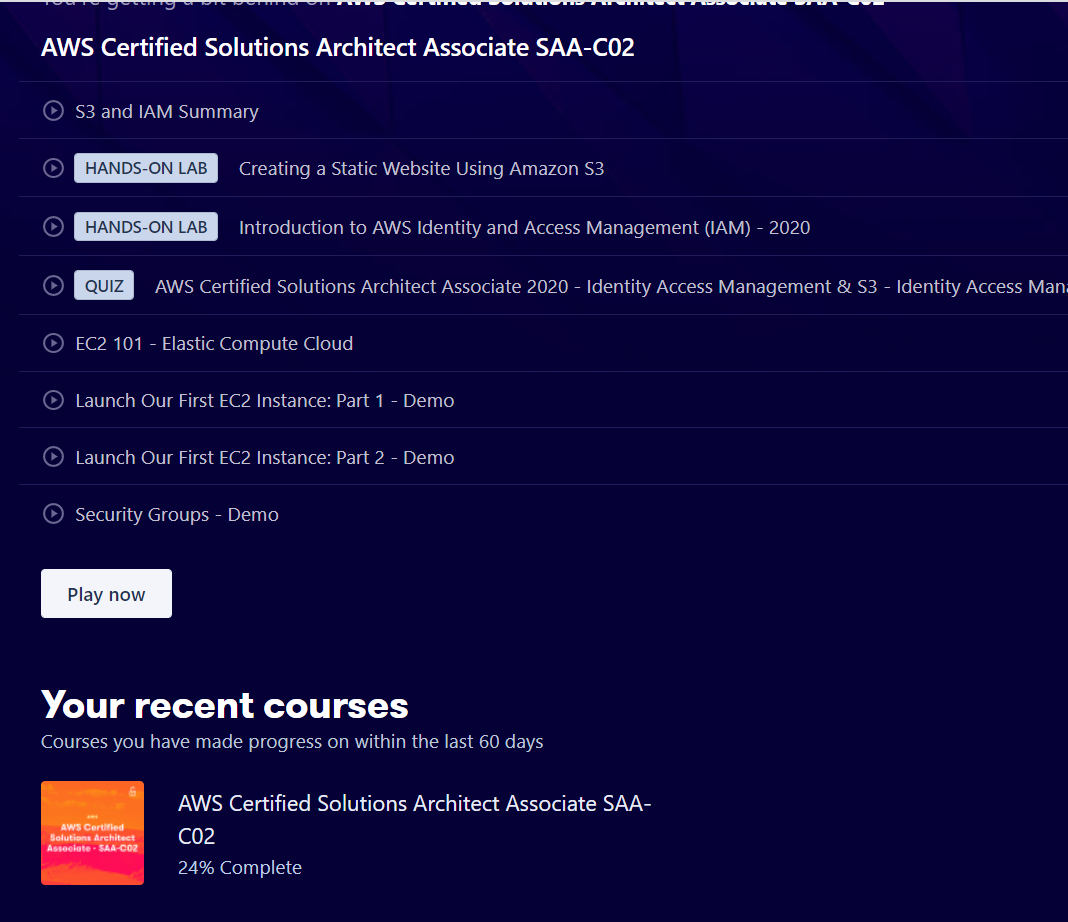


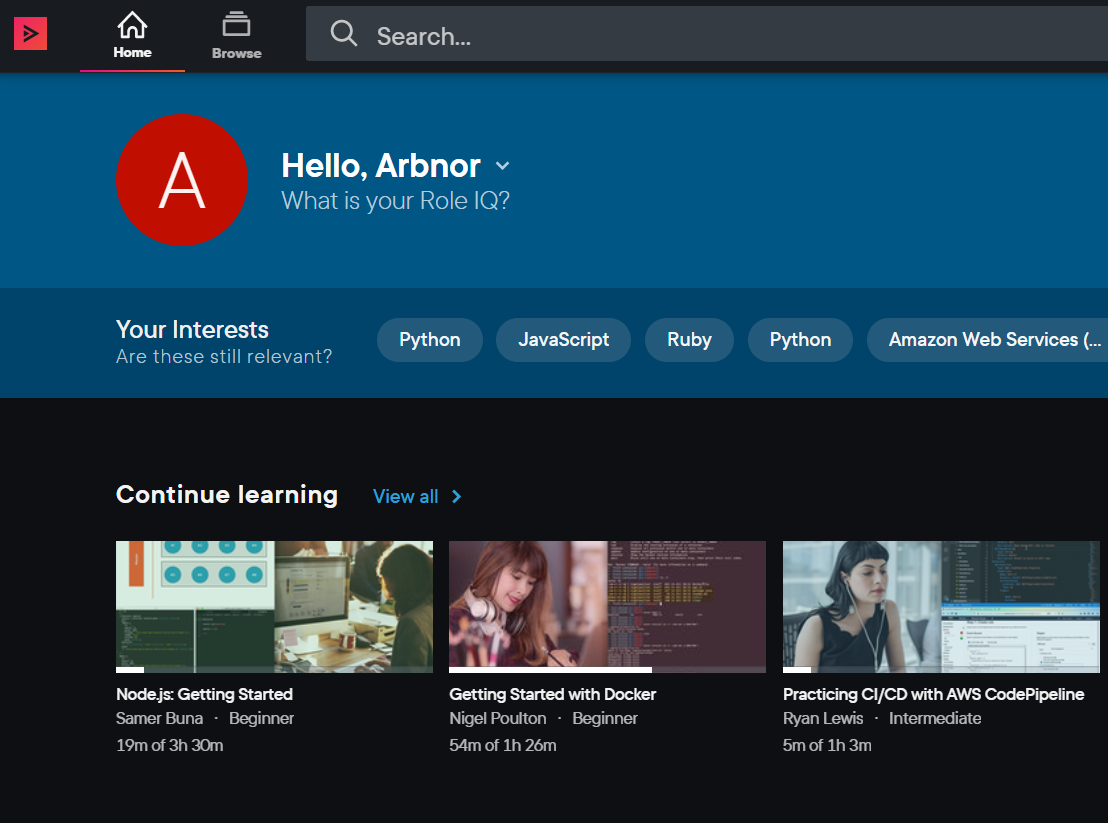
Usually I try to write the test codes such as the function testing for my applications first, however, to do the nature of the complicated test I would sometimes write the code first to see if the API request works and then test and work around it. However, below is a screenshot of a test function before writing the function. There were a lot of problems with testing, as I have never done mock testing continuously to understand it. However, working with my SDETs has been really helpful, and it has helped me understand testing more. Nonetheless, testing before writing code has helped me understand all the potential issues I may get such as 4xx status, other server side being down (such as xmatters) etc. By writing efficient tests, I have reached a 80%+ code coverage which is significant for the business, while I am still yet to cover the remaining 20%.



S15 – Release automation and orchestration as part of a Continuous Integration workflow and Continuous Delivery pipeline, automating the delivery of code from source control to the end users

## S16 – Invest in continuous learning, both your own development and others, ensuring learning activities dovetail with changing job requirements. Keeping up with cutting edge



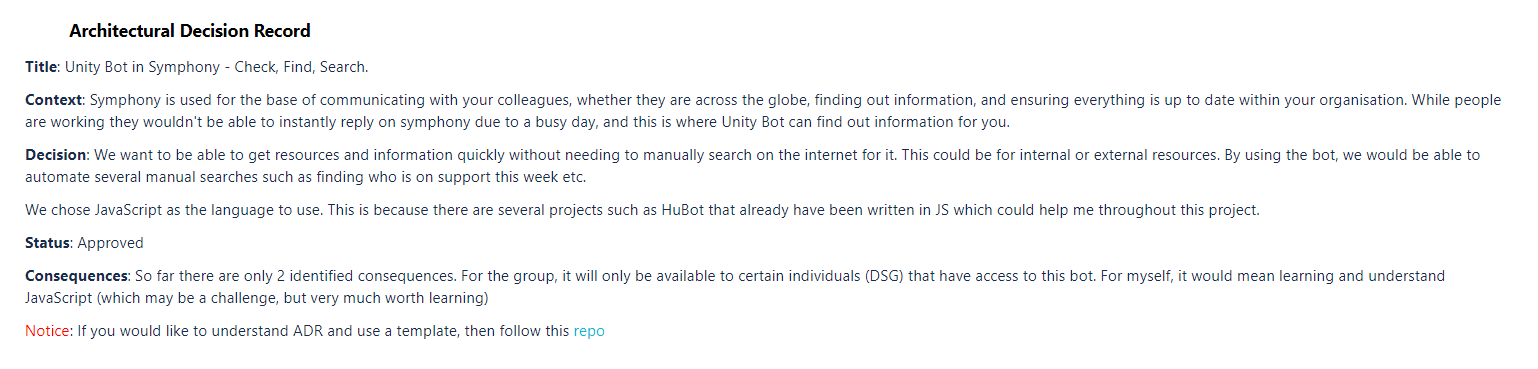


As the role develops and learning becomes more natural to the job, I use ACloudGuru, PluralSIght, and Udemy to stay up to date with technology. I have learnt more on Udemy with NodeJS courses and for TDD. At TRP, we also have our own internal training courses for learning new technology. For example, this week the team and I have learnt about Vault as it has been added to our support rota, and how we fix the tickets raised from other developers.

## S17 – Code in a general-purpose programming language



Above is a snippet of code written in NodeJS of my Symphony Bot.



I used the Architectural Design Record to decide how the project will be structured and coded in. I worked with my tech lead on completing the ADR as it was my first time doing such documentation. For me to learn on what ADR is, or what to mention, I had to simple google what an ADR is, and look at previous documentation from fellow peers.

S18 – Specify cloud infrastructure in an infrastructure-as-code domain-specific language

S19 – Interpret logs and metrics data within the appropriate context to identify issues and make informed decisions

S20 – Writing code in such a way that makes merging easier and facilitates branching by abstraction – i.e. feature toggling

S21 – Application of lightweight modelling techniques, such as whiteboarding, in order to gain consensus as a team on evolving architecture

S22 – Incremental refactoring by applying small behavior-preserving code changes to evolve the architecture

# **Behaviours**

## B1 – Exhibits enthusiasm, openness and an aptitude for working as part of a collaborative community; e.g. sharing best practice, pairing with team members, learning from others and engaging in peer view practices

I intend to have meetings with my peers and ask them to show me systems within T. Rowe Price. For example, how to use Vault, to be able to store users and passwords and read back from it. I also have learnt a lot of security concerns within the SDLC such as not hard coding API access keys etc. I also learn a lot from my team when they discuss different issues in our Friday 10am stand downs. My team also what the best practice when it comes to creating a project, and how to identify its need. I usually have regular meetings with my tech lead who teaches me a lot about Computing in general, and another colleague who has gone through in depth more about the SRE side in my team. I also have discussions with another colleague in the SDET team who have helped me understand TEA more and what additional metrics they are implementing. Apart from my team, I also participate in Code Club Beginners in Python, once I learnt the basics, I was able to teach other colleagues at work who are have no idea in coding. I try to also learn soft skills to improve overall by participating in a Cohort which has 50 + members. I’m able to discuss ideas, present ideas, socialize and learn from them. As they are more senior than me and know more about Computing in general, I find this very valuable as I like to see the thought process from many users on how they tackle different situations.

B2 – Invest time and effort in their own development, recognizing that technology evolves at a rapid rate

## B3 – Displays a commitment to the mantra “You build it, you run it”, taking ownership of deployed code and being accountable for its continual improvement, learning from experience and taking collective responsibility when things fall

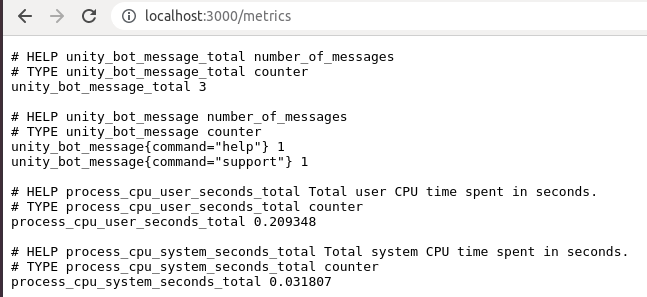
When working within my project, I accept ownership of change as I am the developer of the app. After hearing the client user needs change, and adapting to it, I can modify my project around their needs and implement a new structure to it (with minor changes on it being made). For example, a client wanted the bot to be able to not be @ mentioned when in a private room, and only in a group room. This made sense as always @ the bot can be annoying, rather than you can just type the message and it will listen. Since that client need was suggested and required, I have been researching how to implement that new code change and taking full ownership of their need, but that will be a future release. For now, I am currently working on the 3 main features, and adding the new client need as future release. As I’m new to the learning experience of software engineering, I try to take as much accountably for the failure of the service, so when the bot is not listening after it’s been deployed on AWS, I try give it a shot to debug what the issue is. However, when I get stuck, I ask for help from my tech lead.

B4 – Is inclusive, professional and maintains a blameless culture

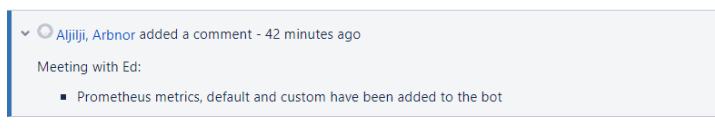
----------------- IGNORE EVERYTHING BELOW, REFACTORING AND REMOVING THIS DUTY -----------------

Just need for my own evidence

Duty 10 – Implement a good coverage of monitoring (metrics, logs), ensuring that alerts are visible, tune-able and actionable



I have progressed on implementing Prometheus metrics using Express for my bot. I currently use the default metrics but have also implemented my own custom metrics. For example, how many messages the bot listens to, and how many commands it should be listening to. E.g. “help” is a command it will respond to, but “foobar” is a command it will not listen to, however, I add that counter to the message\_total.



After completing this, I add it as a comment to my Unity project in Jira, so I know what stage in my project I am at. More metrics to be added via AWS