Assessment Task E2.8.3 – IT Business Case

Work Integrated Learning 2 – Spring 2018

Hayden Crain – 98105873

Table of Contents

Background

iP Edge is a software development company situated within the Northern Beaches of Sydney focusing on providing consultancy services for a range of businesses. Currently, iP Edge are in the process of rebuilding and upgrading their in-house administration tool, which is most notably used to create tasks and log time. Due the amount of developers working on this project, they have moved into an agile development format, integrating fortnightly sprints into their workflow, dramatically increasing the amount of times they are able to release new features to the production environment. Tasks that a developer is working on are placed into the "in Dev" column. Once a task is done, it is moved into the "Quality Assurance" column, to let the other developers know that it ready for testing and code review. Appendix A shows a screenshot of our current sprint planning board, showing tasks are tracked through each stage.

iP Edge have also implemented a new practice into their current source control workflow, known as branching (Bitbucket 2018). This technique has seen some success and has overall been a quite effective tool. Branching allow developers to bundle code changes into separate 'feature branches', which is good because a new feature won't be applied to the main codebase until it has passed all code review and quality assurance checks.

As a result of implementing feature branching, the development team have dramatically improved the time an effort the put into quality assurance and testing. There have been positive outcomes from this, including a decrease in the number of bugs that appear on the production environment. However, the increase in testing time has also occurred due to the amount of regression testing that is involved within this process. When testing the introduction of a new feature, a developer will have to first make sure that it does not break any other sections of the application (Soffer, P. 2018). As the number of features increase within the application, so do the number of potential bugs may arise from certain features conflicting with each other, and thus the time taken to complete regression testing significantly increases.

Additionally, a large portion of the regression testing process is left up to the developer, leaving a large margin for human error. Without maintaining and keeping a list of all potential use cases, it is up to developer to remember and test all potential edge cases. As the number of features increase, it is becoming near impossible to remember and have time to test each case, leaving a large risk for bugs to make it to the production environment before they are spotted.

Business Problem Definition

iP Edge requires their current Quality Assurance process to be improved in order to stay competitively viable, as well as to increase efficiency within its employees. Within their current admin upgrade project, iP Edge is spending roughly 2-3 hours on code review and testing per feature, with this number expected to rise as the number of features within the application increase. Each feature is tested manually by a developer within the team. There is no formal outline to follow when testing, which means it is entirely up to the developer to decide when a feature is deemed satisfactory, ultimately leaving the potential to miss out on testing high-risk interactions with other features. From this, we can separate their main problem into three separate components:

1. To decrease the time taken for developers to complete the quality assurance and testing process.
2. To ensure that all identifiable use cases are thoroughly tested each time a new feature is introduced.
3. To remove the need for developers to conduct manual testing every time a new change is made.

Research Objective

Over the past few months working at iP Edge, I have become quite familiarised and involved with their current agile development process. Through this, I have identified that the use of IT tooling could be applied to ultimately improve their testing process. I propose to assess the efficiency of introducing testing automation into our current workflow to determine whether it will provide any overwhelming value to our process respective to the time and effort it will take for introduction and implementation. In my assessment, I will aim to achieve the following goals:

- Identify and outline the main types of automated testing currently available.
- Outline assessment criteria needed for the assessment of each type of testing automation.
- Compare and assess each type of testing automation, weighing the benefits of each within specific situations.
- Determine if the introduction of testing automation will reduce the testing time taken during the Quality Assurance stage.
- Determine if the introduction of testing automation will remove the need for developers to re-test the entire project

Test automation is a special IT tool which aims to execute tests that compare actual outcomes with predicted results (Wikipedia 2018). Within software testing, there are many different types of software testing, all designed to undertake specific duties. For my research, I will aim to investigate automation primarily within functional testing, which includes software testing types such as unit testing, integration testing, and system testing (Software Testing Help 2018).

Currently, our quality assurance and testing process is as follows:

1. Once a feature is completed, a pull-request is made, and the task is moved into the Quality Assurance column of our sprint board. Another developer will see this new pull request and begin the testing process.
2. The testing developer will ‘pull’ (download the code) the changes made to their own computer. They will then begin a manual test, testing that the code:
    a. passes all the acceptance criteria for the feature
    b. does not interfere with any other parts of the application
    c. does not cause any bugs or issues
    d. does not cause any old bugs or issues to re-appear (also known as regressions)
3. Once the manual test is completed, the written code is reviewed to ensure that the developer is following best coding practices and will be notified if any enhancements or improvements can be made.
4. Once stages 2 and 3 are completed, the new feature is approved. The code changes are added to the ‘master’ branch, which in turn will trigger a deployment to the staging server (a server which mimics the ‘production’ server and is used for more rigorous testing).
5. Once the feature has been deployed the staging server, the feature undergoes one final check to ensure that the feature runs smoothly. Business Approval is also conducted at this stage.
6. Once passed stage 5, the feature can now be marked as ‘done’.

A visual representation of this process can be seen in Appendix C. Note that the largest pain point that has been identified occurs within the ‘run application and test for regressions’ process.

Given that testing automation can be confirmed to provide business value and can be successfully implemented into iP Edge’s agile workflow, testing automation ultimately will affect the workflow as seen in

Appendix C. It is possible to run such automated tasks as soon as the pull request is made, which means that the testing developer is not required until later in the process.

Business Analysis

Within the problem definition defined earlier, we identified three main components that were regarded as the strongest pain points within the current quality assurance process. For this proposal, there are two possible choices that we can make that could potentially improve the efficiency of the quality assurance process. Those two choices are to:

- Integrate testing automation into the current QA process; or
- Keep the current process as is (manual testing).

Choice A – Integrate Testing Automation

Costs

There are some noticeable costs that should be considered if iP Edge were to integrate testing automation into the QA process. Firstly, introducing a new tool to team of developers introduces a certain amount of training required before they can become efficient with the tool. Depending on how the training is conducted and how long the training process lasts for, costs will have to be considered regarding developer salaries during the training process. Additionally, the trainee's wage will also need to be considered.

On the other hand, it is possible to ask the developers learn the tool in their own time (out of work hours) as more of a hobbyist project. This means that there will be no costs for the training process. However, this also means that the timeline for the training process will need to be extended, as developers will have less time (depending on the developer, maybe half an hour per day) to teach themselves.

Once implemented, we expect that the costs of initial training will be outweighed by the time savings due to its introduction. Time usually spent testing the entire project every time a new feature is introduced (roughly two hours and will increase as the number of features increase) will be cut down due to the automated process. However, there will be a slight increase in time when beginning to develop a new feature, as new tests will have to be written by the developer specifically for the feature.

Benefits

- Less Reliance on human interaction – allowing a computer to do mundane tasks for a human is already a huge benefit as it removes the potential risk of human error.
- Test cases are well documented – Before the computer can automate tests, you first must give it something that it can understand (e.g. a test case written in code). This means that test cases will be neatly stored and documented into code files, making it easier in the long run to figure out what is and what isn't being tested. Additionally, this means that code and test cases can be well documented for future developers.
- More accurate testing – More accurate testing will be done as test cases are formalised and can are standardised over time.

Risks

- The time needed to write tests become inefficient – The downside to automated testing is the added initial stage of writing test cases for the feature you are to implement. If this ends up taking longer than what it would to manually test all the features, then the testing automation does not have any perceived business value to it.

- High reliance on the document test cases – There is a chance for developers to forget to write tests before the commencement of a new feature. This will result in the tests to become undocumented. Without manual testing, it will be very hard to notice that a bug exists as a result.

Choice B – Keep QA Process as is

Costs

Although we are keeping the process the same with this choice, there are still some costs that need to be considered. As discussed previously within this proposal, as the number of features within the system increases, the number of things need to be tested also increases. Therefore, we will need more time when testing the entire project during the QA process. This ultimately leads to an increase in cost, in order to pay for the increased time that the developer spends on the task.

Benefits

- Familiarity – the developers working at iP Edge are familiar with this current process and introducing an extra step / altering an old process may cause some disruption in the workplace.

Risks

- Decreased productivity – as the number of feature increase, the more time that will be required to test the entire application.

Cost Estimation

As preparation for this research task, I created a high-level overview of the potential benefits of automated testing, as seen in Appendix D. The provided are given that an average developer rate at iP Edge (including all costs associated with the developer) is $80 per hour, and an average development time per feature of 8 hours (when undertaking manual testing). These costs only consider the implementation costs, and do not consider time spent during the research proposed within this report.

Additionally, I have created a graph of these results, show in Appendix E. This shows the number of features that will be required to be developed before the automated testing option will breakeven to the manual testing option. An average iP Edge sprint will usually completed around 18-20 tasks. A sprint lasts two weeks, which means that the testing automation choice will breakeven during the third sprint (roughly six weeks, and after around 40 features).

Recommendation

My recommendation is to go ahead integrating testing automation into the QA process, as it will ultimately enable us to see if iP Edge will be able to: (1) decrease the time taken for developers to complete the quality assurance and testing process; (2) ensure that all identifiable use cases are thoroughly tested each time a new feature is introduced; and (3) remove the need for developers to conduct manual testing every time a new change is made.

Implementation Strategy

The implementation strategy for this proposal would be broken down into two stages. The first stage would be the initial research stage. Within iP Edge, we regularly conduct research tasks, which we call ‘spikes’.

The first step is to introduce a spike for myself to investigate which sections of testing automation would be most useful to investigate and integrate into our project. There are many different types of functional testing, including unit testing, integration testing, system testing and regression testing (Software Testing Help 2018). All of these will have to be investigated and assessed in order to determine whether it would be useful to implement.

The second step within the initial research stage would be to investigate potential test automation tooling that exist for our project's technology stack. As we are using the .NET framework for our web development projects, I initially explored a few options that were designed specifically for .NET applications. The most notable options available were XUnit for unit testing, NSpec and SpecFlow for acceptance testing, and TestServer for integration testing.

The third step will involve compare and assess each of the potential test automation tools identified in step two. They will then be assessed according to a set criterion, which will aim to assess each tool's usability for the developer as well as the time it takes to run tests.

As of the creation of this proposal, I have made progress into much of the initial research stage. Appendix F shows the task that we had created for me to work on, requiring me to investigate which automated testing options would be useful for the back-end side of our project. This task was timeboxed to 15 hours, which means I was to stop work at the 15-hour mark and report my findings. Through my research, I was able to discover that integration testing would be starting point for integrating automated testing. I was able to write some basic tests in order to test to see that our API was working as intended, as seen in Appendix G. Appendix H shows one of the test cases I had written. The test is written using XUnit and was to test that unauthenticated users should not be able to access authorised data.

The second stage of my implementation strategy would consist of the actual implementation process. Once the automated testing options have been scoped out, it will then be required to train the current developers working on the application. This should take no less than a day. As I have been conducting research in the previous steps, I will be the trainee within this process. Once trained up, it is expected that the developers will create test cases before the commencing work on their feature. This process is also known as Test Driven Development (TDD) (Farcic, V. 2013).

Measuring Success

Once implemented, it is required for there to be a way to measure the amount of value testing automation has provided for the project. Looking at the three components defined in the Business Definition Problem of this stage, it is evident that time is a big concern for all three. The amount of time taken within development is directly correlated to the amount that each feature will cost. Ultimately, decreasing the amount of time taken per feature while retaining business value for the customer is the key goal for iP Edge.

Therefore, I have come up with two very basic criteria that will be enough in determining the success of testing automation. The integration can be deemed successful if there is:

- A notable decrease in development time per feature
- A notable decrease in the number of bugs present in production.

Additionally, I have proposed to assess testing automation using three main Software Architecture Quality Attributes; Reliability, Usability and Performance. These three attributes represent areas of concern that have the potential for application wide impact (Microsoft 2009). In relation to automated testing, I have proposed these attributes to be of the largest importance:

- Reliability – The automated tests must continue operating in the expected way over time without any changes.
- Usability – The automated tests must be easy for the developer to use and must be designed with future developers in mind.
- Performance – The automated tests must be fast enough to provide an overall benefit compared to manual testing.

- Scalability – As the number of features increase within the project, so will the number of potential test cases. Scalability is ability of a system to either handle increases in load without impact on the performance of the system, or the ability to be readily enlarged.

Budget

As stated within the business analysis section, there are some noticeable costs that should be considered. The costs regarding the latter stage of Implementation are difficult to estimate, as it is an ongoing process. However, Appendix D and Appendix E attempt to best estimate the potential cost of implementation over time.

There are also some costs regarding the initial research phase, as well as the beginning of implementation. Research will be primarily conduct by myself, and I have estimated it will take roughly 30 hours total. At a rate of $25 per hour, it is estimated that total research costs will total to $750. Additionally, there are costs relating to the initial developer training. There are roughly five developers working on this project currently. I have also estimated that training will take around one day. As seen in Appendix D the project total cost for training is estimated at $3,200.

As a result, the total project cost for research and training is estimated at $3,950. All other costs after this are ongoing and are dependent on the number of features that are worked on.

Project Timeline

To reach the goal of completing the research project within a reasonable time, I have produced a Project Roadmap that I will use as a guide, as seen in Appendix I. After a successful initial pilot implementation and approval of my findings, there will be an initial training session at the beginning of January 2019. After training, implementation will continue as an ongoing process through the lifetime of the project.

References

Bitbucket 2018, *Using branches*, viewed 23 November 2018, <https://www.atlassian.com/git/tutorials/using-branches>.

Soffer, P. 2018, *What is Regression Testing?*, test IO, viewed 23 November 2018, <https://test.io/software-testing-guide/what-is-regression-testing/>.

Wikipedia 2018, *Test automation*, viewed 25 November 2018, <https://en.wikipedia.org/wiki/Test_automation>.

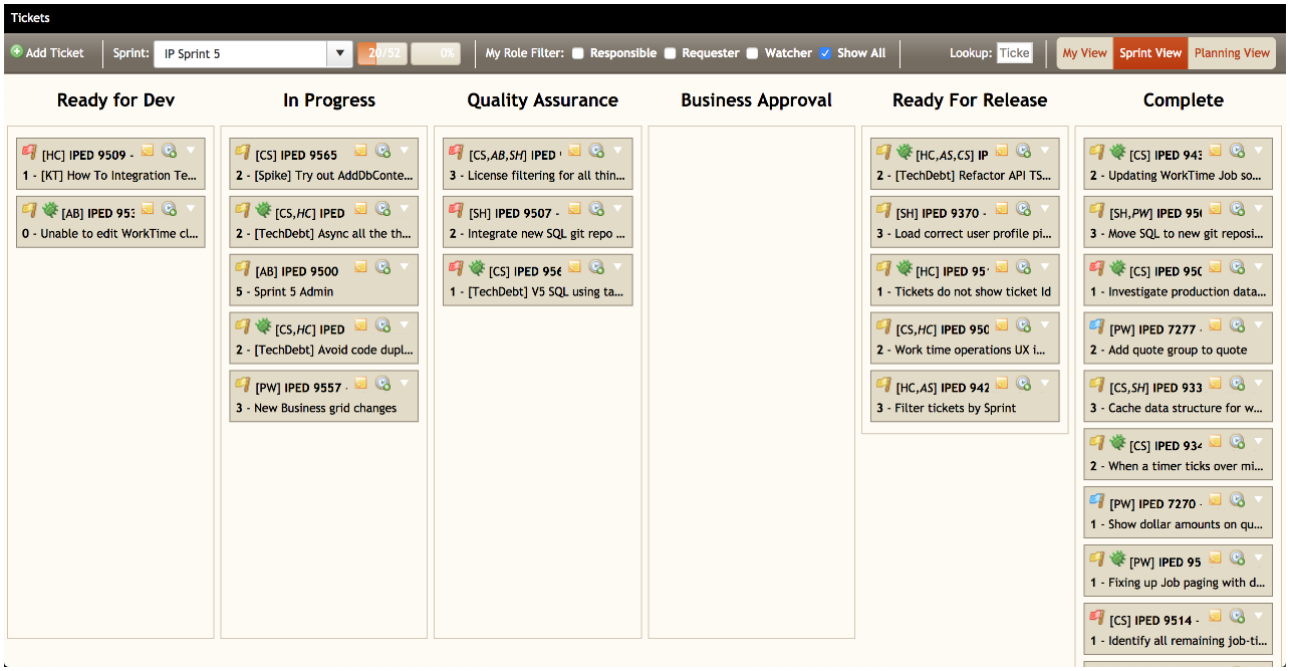
Software Testing Help 2018, *Types of Software Testing: Different Testing Types with Details*, viewed 26 November 2018, <https://www.softwaretestinghelp.com/types-of-software-testing/>.

Microsoft 2009, *Chapter 16: Quality Attributes,* Washington, viewed 10 December 2018, <https://msdn.microsoft.com/en-us/library/ee658094.aspx>.

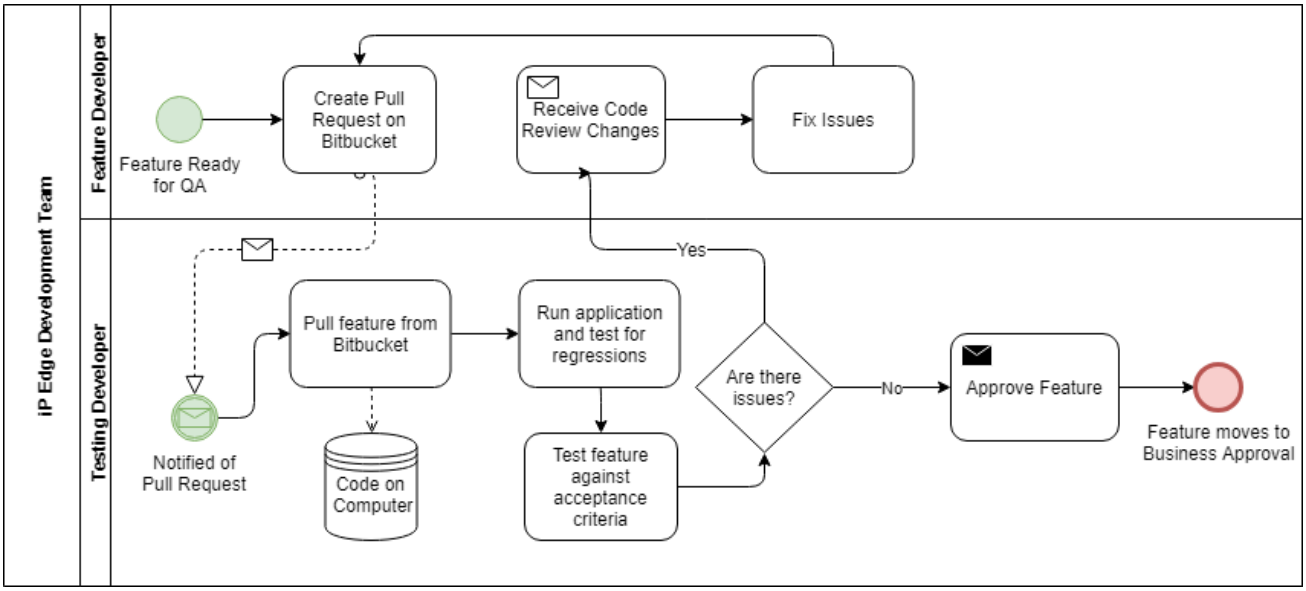
Farcic, V. 2013. *Test Driven Development (TDD): Example Walkthrough*, viewed 10 December 2018, <https://technologyconversations.com/2013/12/20/test-driven-development-tdd-example-walkthrough/>.

Appendices

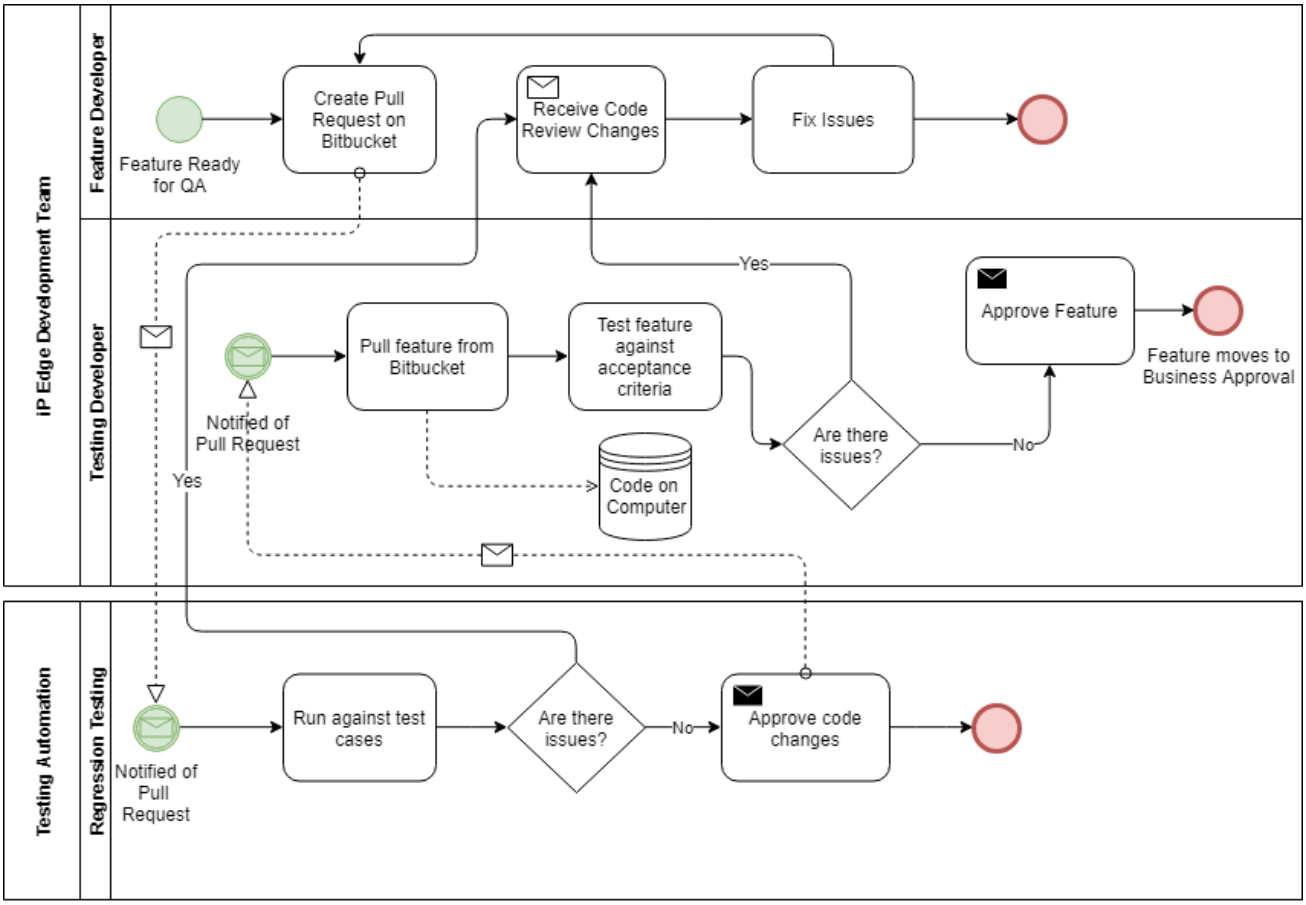
Appendix A – iP Edge Sprint Board



Appendix B – Current Quality Assurance Process



Appendix C – Quality Assurance process with regression testing integration

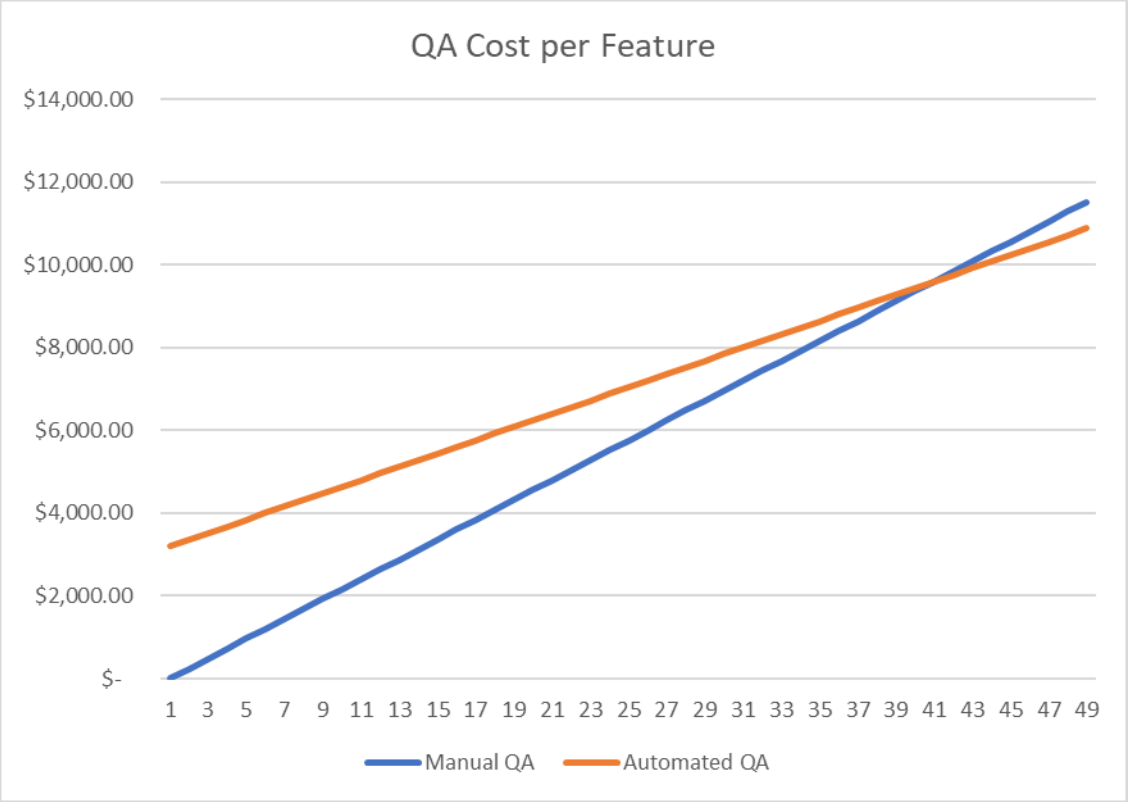


Appendix D – Cost Benefit Analysis

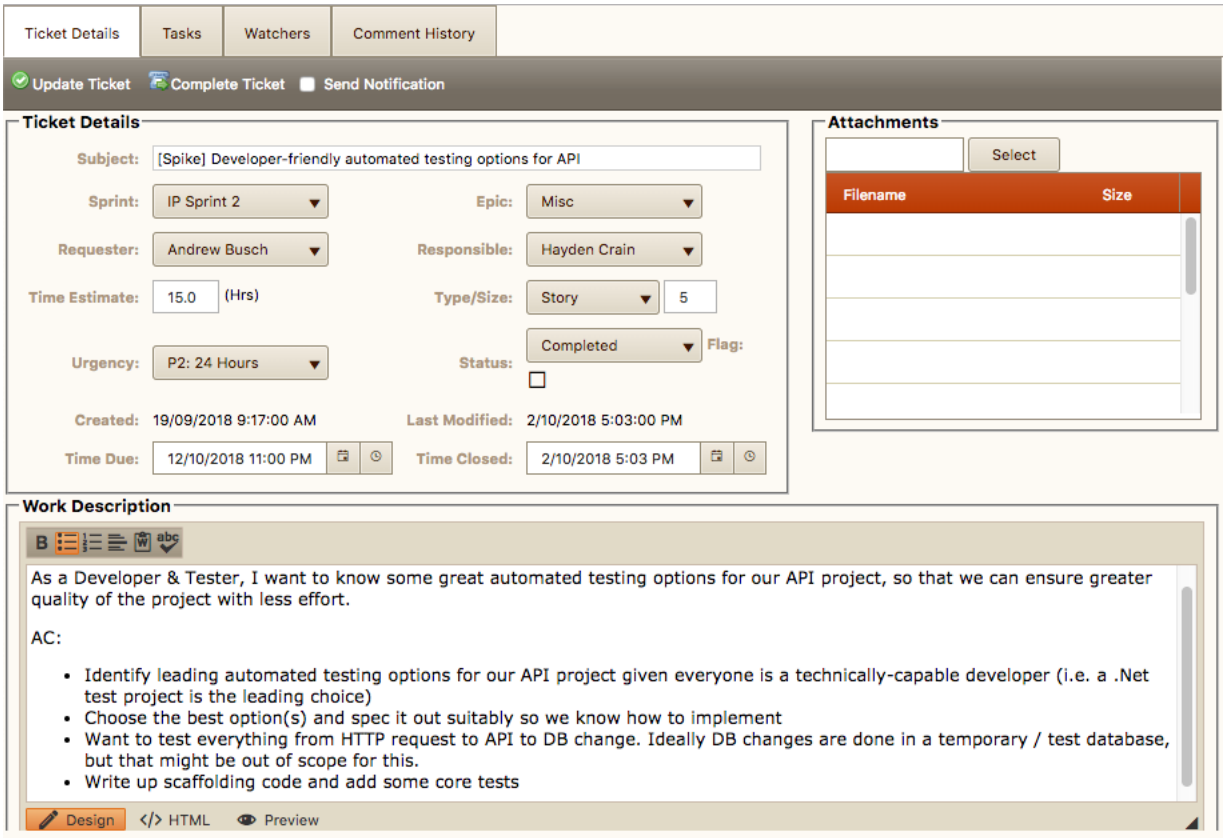
Testing Automation				
Fixed Costs				
Labour	**Quantity**	**Hours**	**Cost / Hour**	**Total**
Training	5	8	$ 80.00	$ 3,200.00
Reoccuring Costs (Per Feature)				
Development	1	1	$ 80.00	$ 80.00
Testing	1	1	$ 80.00	$ 80.00
			Total	$ 160.00

Manual Testing				
Fixed Costs				
Labour	**Quantity**	**Hours**	**Cost / Hour**	**Total**
Training	0	0	$ 80.00	$ -
Reoccuring Costs (Per Feature)				
Development	0	0	$ 80.00	$ -
Testing	1	3	$ 80.00	$ 240.00
			Total	$ 240.00

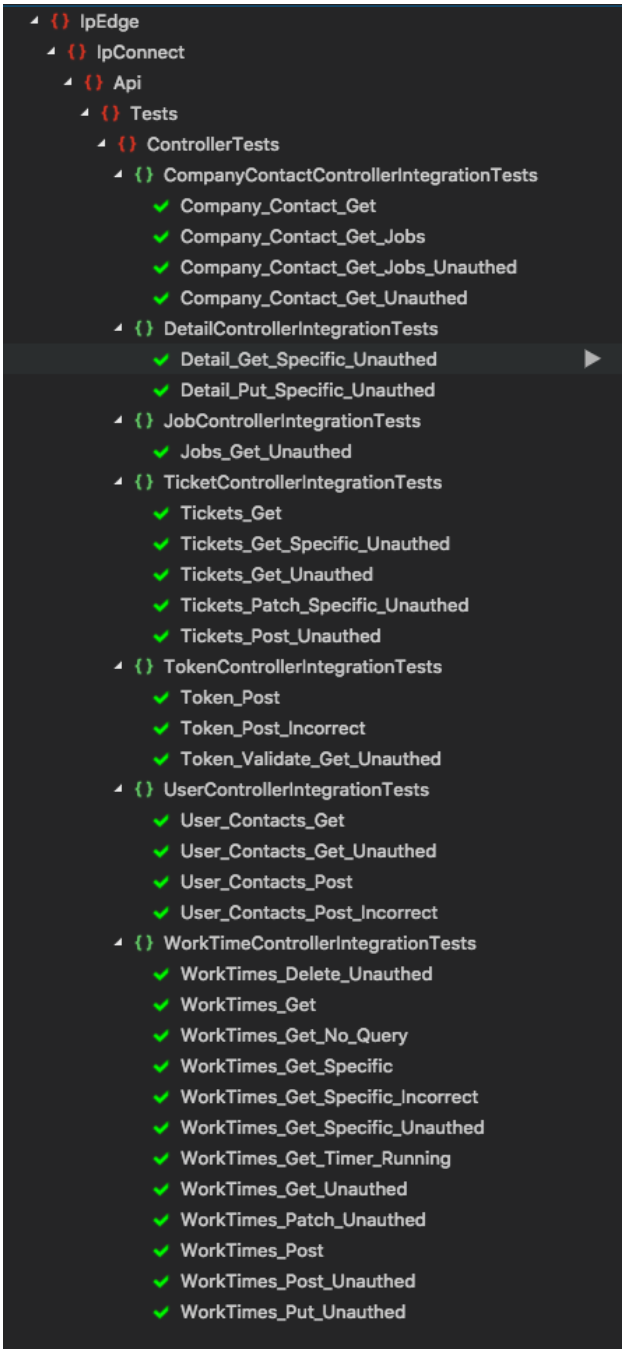
Appendix E – Graph of Cost Benefit Analysis



Appendix F – Testing Automation Research Spike



Appendix G – Current Integration Tests

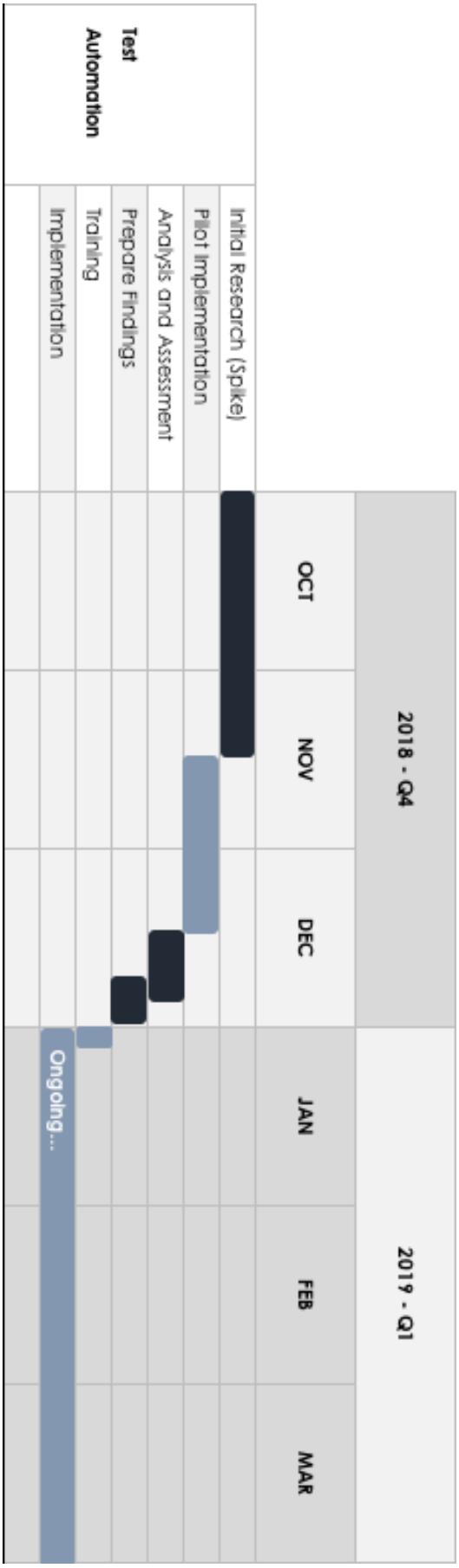


Appendix H – Unauthorised Access Test Case

```
[Fact]
0 references | Run Test | Debug Test
public async Task WorkTimes_Get_Unauthed()
{
    var endpoint = _path;

    // Given that a user is not authenticated
    _fixture.RemoveAuthenticationHeader();
    // The user should not be able to make a GET request on the /work-times endpoint
    await _fixture.Get_Unauthed_Receives_401(endpoint);
}
```

Appendix I – Project Roadmap



Appendix J – Original E.8.2 Proposal

1. Introduction

Over the past few months working at iP Edge, I have become acquainted and thoroughly involved with the workplace's business processes. During that time, I've seen many changes of aspects of our development processes, including an expansion of our technology stack, implementing agile development into our internal project, as well as improving the way we develop and integrate new features into our applications. However, there are many aspects that we are still actively wishing to improve on. This proposal report will aim to highlight a notable weakness within Quality Assurance phase of our current development practice and aim to conceive and evaluate the effectiveness of the introduction of IT solutions.

2. Statement of Problem

Within the iP Edge workplace, we use an in-house administration tool, which is most notably used to create tasks and log time. For this project, we have recently moved to agile development, integrating fortnightly sprints into our workflow. This has dramatically increased the amount of times we release new features to our beta admin client. Figure 1 below shows our current sprint, of which we are in the second week of development for. Tasks that a developer is working on are placed into the "in Dev" column. Once a task is done, it is moved into the "Quality Assurance" column, to let the other developers know that it ready for testing and code review.

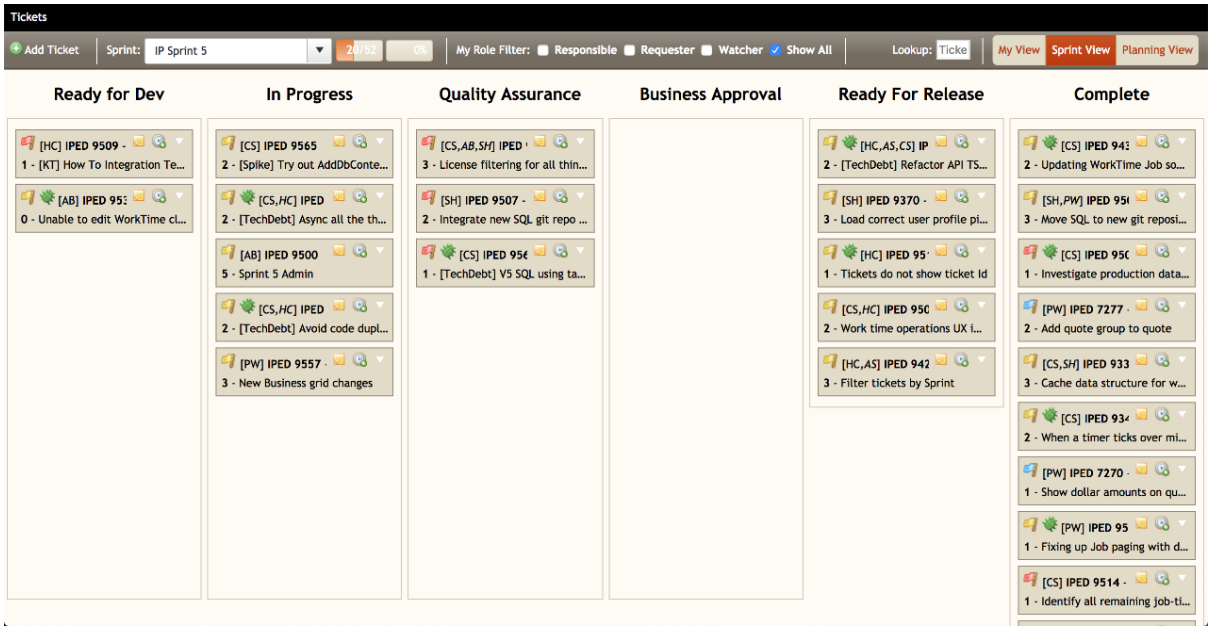


Figure 1 - iP Edge sprint board for our internal admin project

We have also successfully implemented a new practice into our current source control workflow, known as branching (Bitbucket 2018). Using this technique has been quite effective for us. Before this, all our code changes were being added directly to the master branch. With branching, it allows us to essentially bundle code changes into separate 'feature branches', which is good because a new feature won't be applied to the main codebase until it has passed all code review and Quality assurance checks.

However, we still have a couple of issues with this management process, specifically to do with Quality Assurance and testing. As our company do not have dedicated testers, the development team are required to partake in the QA process. Every time a new feature is ready for testing, there is a chance that the code added

for the feature may have affected another part of the application, also known as regression testing (Soffer, P. 2018). This means that when testing the new feature, a tester will have to first make sure that it does not break any other sections of the application, ultimately increasing time spent testing, as well as leaving a large margin for human error.

2. Objectives and IT Solutions

I propose to review the current Quality Assurance process that we partake in within iP Edge. In this review, I have identified two main goals:

1. Decrease the time taken for developers to complete quality assurance and testing.
2. Remove the need of developers testing the entire project every time a new change is made.

Currently, we are spending roughly 2-3 hours on code review and testing per feature, so a noticeable improvement would be made if we were to cut this time down. After investigating for some potential IT tooling that could benefit this process, I have come to realise that implementing automated testing into our projects may be useful in the long term and may be able to facilitate the two goals defined above. Test automation is a special IT tool which aims to execute tests that compare actual outcomes with predicted results (Wikipedia 2018). It is ultimately able to automate repetitive tasks in order to improve the testing process.

The first for my plan of action would be to investigate which sections of testing automation would be most useful to investigate and integrate into our project. There are many different types of functional testing, including unit testing, integration testing, system testing and regression testing (Software Testing Help 2018). All of these will have to be investigated and assessed in order to determine whether it would be useful to implement.

The second step for my plan of action would be to investigate potential test automation tooling that exist for our project's technology stack. As we are using the .NET framework for our web development projects, I initially explored a few options that were designed specifically for .NET applications. The two most notable options available were XUnit for unit testing, NSpec and SpecFlow for acceptance testing, and TestServer for integration testing. The third step will involve compare and assess each of the potential test automation tools identified in step two. They will then be assessed according to a set criterion, which will aim to assess each tool's usability for the developer as well as the time it takes to run tests

3. Analysis and Feedback

As preparation for this research task, I created a high-level overview of the potential benefits of automated testing. Although the time taken for testing will be decreased, there be an initial amount of time spent for developer training. Additionally, there will be an increase in development time, as developers will be required to code test cases before the completion of a feature. These values are given that an average developer rate at iP Edge (including all costs associated with the developer) is $80 per hour, and a average development time per feature of 8 hours (when undertaking manual testing).

Cost Benefit Analysis: Testing Automation				
Fixed Costs				
Category	**Quantity**	**Cost Per Hour**	**Hours**	**Total**
Training	5	$ 100.00	8	**$ 800.00**
Reoccuring Costs - Manual (Per feature)				
Category	**Hours In Dev**	**Hours in QA**	**Cost Per Hour**	**Total**
Manual Testing	5	3	$ 80.00	**$ 640.00**
Automated Testing	5.5	1	$ 80.00	**$ 520.00**

Figure 2 - Cost Benefit analysis

I reached out to my workplace mentor – Andrew – regarding this proposal, as seen in the figure below. After a few further discussions, we agreed that it would be useful to investigate and potentially introduce testing automation into our admin project.

Hey Andrew, for one of my university assignments, I've been required to explore some of the innovative IT solutions that have been brought into iP Edge over the past 10 or so years in order to evaluate its impact on the company as a whole. I've been asked to create a proposal for potentially introducting a new IT solution into the company.

I've been looking into the processes involved within our testing stage for our ip connect project, and I was thinking we might be able to improve our testing time by introducing automated testing? I've played around with the idea at another job, as well as during a couple of hobbyist projects. What would you thoughts be on this?

Just because we spend a lot of time currently testing the entire project every time we implement a new feature, it might be easier to create some base test cases so we only really have to focus on testing the new feature

andrew 6:28 PM

Hey Hayden, sounds like a good idea. Definitely keen to speed up our QA process, and automated testing should help with that. Think it'll mean we catch many regressions before releasing them, plus as we keep growing our test coverage, our confidence of the application reliability should increase over time.

plus all the cool kids are doing it

Figure 3 - Slack message showing the initial conversation with Andrew

3. Conclusion

The company of iP Edge has seen some notable changes within their development workflows over the past 10 or so years. However, there are still notable aspects of their current process that that could be improved on. This document has proposed research to evaluate the effectiveness of introduction automated testing into the Quality Assurance phase of our current development lifecycle. My recommendation is to go ahead with the research, as it will ultimately enable us to see if iP Edge will be able to: (1) decrease the time taken for developers to complete quality assurance and testing; and (2) remove the need of developers testing the entire project every time a new change is made.