**IBL Student Report: Saral Gautam**

**Contents**

**4A Organisation Description**

**4B Role and Tasks**

**4C Major Project**

**Introduction**

Hello to the FIT IBL and any readers of this report. My name is Saral, and I had my IBL placement at Origin Energy working in the Reporting and Data Operations team. Over the course of my placement, I completed both operational and development-based tasks with my overarching legacy being a metric input page for Origin’s business metrics. This report will recount my experience with Origin, beginning first by outlining Origin as an organisation. Following this description, I will then elaborate on my individual role within Origin and the key experiences I learnt from. The report will then conclude, with an analysis of my legacy project along with units from my university studies I found to be useful through my time at Origin.

**Origin – The Organisation**

Origin Energy was founded in February 2000, when Boral (another major Australian company) shareholders approved their energy business to become its own company. Now Origin Energy is one of the largest energy retailers in Australia, along with being an energy producer managing power stations in Queensland, New South Wales and Victoria. As an organisation the main purpose of Origin is to build a sustainable energy future, as such Origin has placed significant investments within solar and battery technology and digital tools to support its vision. Additionally, customer engagement and satisfaction are also a major part of Origin’s vision - something which I personally saw through the work the Reporting and Data team do. Having a large scale in operations, means Origin has a wide data footprint covering everything from customers, the energy market, financial reporting- almost anything that can be measured. All this data is managed within Origin’s data lake, aptly named after an actual lake in the Snowy Mountains on NSW- Jindabyne. The team I worked in, broadly came under the Future Energy & Technology wing of Origin working specifically in Reporting and Data Operations, as shown in the organisational chart below (source peopleCentral). Just as shown in the chart, Origin divides each major area of the business. Each area performs a different function to support the business as a whole. The area of Origin I was working in was purely data driven, even within the Reporting and Data Operations Team we had people who would work on creating and maintaining business reports with Tableau (a data visualisation tool) or worked on the batch jobs which were the data sources for the reports. Thus, I was part of a huge organisation with many moving parts which I was able to support through my daily tasks and the major bodies of work I completed.

**The Individual Role**

At Origin my main role was to support my team in their reporting-based tasks, though I broadened my scope of work over the course of the placement. The technical name of my role was a ‘Reporting Analyst’ and my main reporting-based duties involved two major reports at Origin- The Net Position Report and the Origin Active Number (OAN) Report.

Graphical user interface

Description automatically generated

Figure 1: The Net Position report. One of the main reports I worked with during the placement.

These two reports were perhaps the most direct measures of Origins business performance, the Net Position report showing the number of customers won and lost over the financial year and the OAN showing the total number of accounts Origin is providing a service for. Within retail one could see their specific domain such as electricity, gas, broadband and its total sales in the Net position report, so it gave a good indication as to the success of any new business initiative at Origin. Additionally, in OAN one could see the overall status of any Origin account. This included their billing status, the consumption status (some premises were consuming electricity from Origin but did not have an Origin account), scale of operations and the overall movement of customers in any of Origin’s sectors.

Since these reports are used to assess Origin’s performance in retail and in audits from other companies their correctness is critical. Furthermore, OAN and Net Position act to verify the success of different business plans- if your team was doing well then it should show in one of the two reports.

Graphical user interface, application

Description automatically generatedAs such my main task was to refresh these reports daily and investigate the numbers within them. If there were any sharp increases or decreases, I would report this to my team or investigate further with SQL queries to Jindabyne data tables or look at the report itself with Tableau. Some days were almost entirely spent running SQL queries to analyse OAN movement or explain any drastic changes in Net position. The classification of customers within these two reports depended on ‘business rules’: essentially case statements in SQL. When the energy market changed in any form like with the 5MS implementation on 1st October 2021, my team would have to change the SQL code for the tables accordingly or the tableau reports themselves to ensure the reports correctness. With the 5MS implementation for example, this affected when losses were records in the Daily Sales and Losses report which had a subsequent effect on the projections in Net Position, so we had to change the way we made projections to ensure accuracy. Moreover, Origin could also change its internal workings which would also lead to code changes in OAN and Net Position. This was highlighted in the change from the SAP billing system to Kraken, a company based in England, so all the underlying Jindabyne tables were remade so they could again be used in reporting. In the process of migrating customers to Kraken from SAP, there could be some ‘missing’ customers which we could frequently have to investigate for OAN, as it’s business rules did not factor for any leakages of customers during migration. All these changes would need to be explained on a weekly basis in key meetings that my team lead, Eva, had so that senior managers could use this information for business decisions (like maybe launching a new campaign in an under-performing area).

Figure 2: The OAN (Origin Active Number) report. One could see the details of any registered Origin account in this report.

Thus, there was an aspect of constant maintenance to these reports. I also worked on contributing to some of the reports, creating views for stakeholders or adjusting their frequently used reports so they could find specific information.

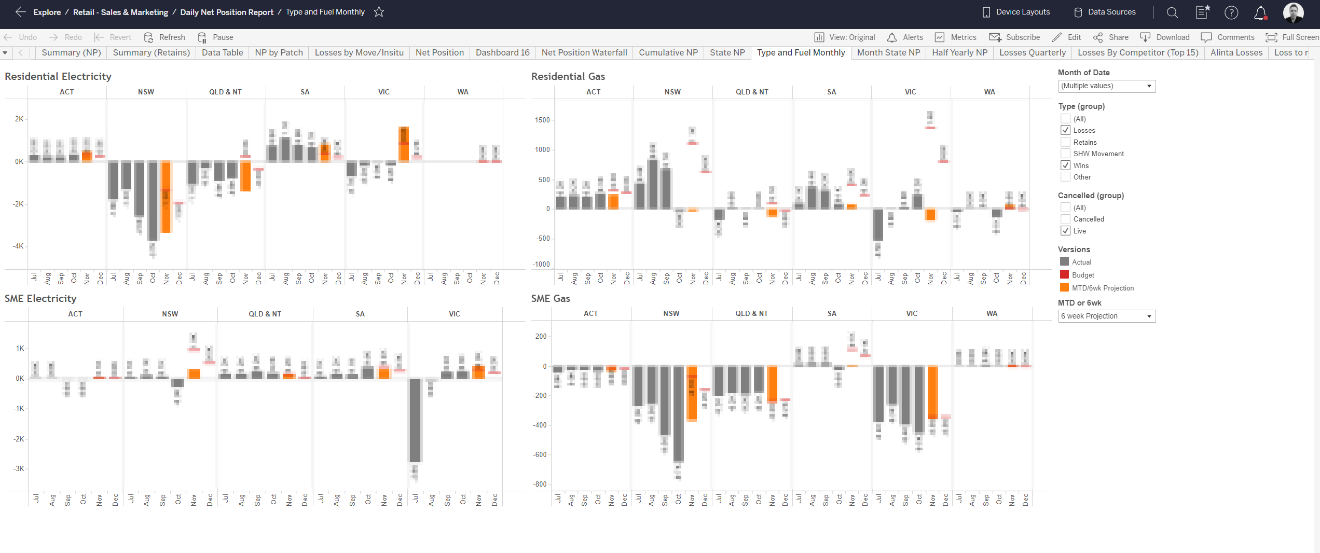


Figure 3: A view I made, showing how each area residential or SME (small to medium sized enterprise) was performing against budget,

Though my role involved reporting based tasks, most of my work was actually in software development. This came about because one of my colleagues, Kai, was looking to create an application that could directly input data into a Jindabyne table for Origin Scorecard. The Origin Scorecard was an important report because it was used in one of Origin’s board reports, but it was currently managed with a spreadsheet. The whole workflow for gathering data for the metrics in the report was highly inefficient. Kai would generally have to email executives or managers for the results of metrics that Origin Scorecard contained, and their gathering really depended on whether they would respond to his email. Furthermore, these people would have deadlines for inputting this data and could sometimes need to input data after work hours, where Kai would be unavailable. Thus, he envisaged an application where the executive and managers could directly come to input a result and it would then update into the Scorecard instantaneously. When he posited this idea, he noted that another Origin employee, Issac Ekbote, had already created a similar application, but it was seldom used because of its grid-like user interface. Taking Issac’s work as a proof of concept, I began working on developing the application and within a month’s time, Kai and I put a production version of the application into his Origin Scorecard Report.

A picture containing shape

Description automatically generated

Figure 4: The landing page of the Origin Scorecard build. A user could select a metric and then input a result.

With Scorecard we were able to create a direct link between any user and Jindabyne, which once their data was in Jindabyne, meant that it could be used for all the reporting capabilities that Origin had to offer. Previously such a link only came through an intermediate person, an analyst perhaps like Kai or myself, who may take data from a spreadsheet to load into Jindabyne or an overnight batch job. Establishing this link to Jindabyne allowed us to broaden the spectrum of data capture with Origin-wide reports, which was pivotal in my legacy project.

**Legacy Project Outline**

Building on from Origin Scorecard came the Sustainability Data Capture (SDC) which was my legacy project at Origin. I will describe this project in further detail in later sections, but for the purposes of reflecting upon the tasks I performed, most of the key lessons I learnt came from or were emphasised within the SDC build. In brief, the SDC was a web application that allowed users to input and review metric data into Jindabyne so the data could be used for reporting purposes. In creating the SDC, my team were given only a ‘requirements’ document which contained all the functionality the SDC was expected to have- other than that we were on our own.

**The Three Key Lessons**

Below is covered the three main lessons that cover most important skills I learnt through my placement at Origin. Beginning with reading the documentation.

*Read the documentation*

In the SDC build, one expected requirement was that it could validate a user without them having to fill in any kind of login prompt. Thus, I had to be able to retrieve the username of the user entering the application without any kind of form. Though I had already done this in Scorecard, SDC was different because it was expected work outside of Tableau (Tableau allowed us to send the username without any kind of code implementations). When confronted with a specific coding problem such as this, my workflow at this point was just to look for commands or keywords on google or stack overflow posts and then go through a process of mindless trial and failure. This was inefficient for two main reasons:

1. It was massively time-consuming.
2. Each failure does not inform the next, so there is no progression to a solution.

This trial and failure process were exactly what I did when confronted with the username retrieval problem. I had set a day to implement this functionality; four hours had passed, and I was not any closer to a solution then when I began. In a despondent state, I asked Issac for help and what he said will perhaps resonate with me for the rest of my ‘career’:

I asked, “Issac do you know any way I can retrieve the username?”.

He replied, “It is already there.”

He elaborated saying that the solution was in the documentation- the one place I hadn’t looked. I then implemented the command he showed, and I was able to retrieve the username. Later on, in the SDC build I reached a similar point, where I needed a command that could send data to and from my front-end. This used ajax, a method in javascript, but this time I went straight to the documentation. I read each line and found the data method, which worked instantaneously. This time was different because I took the time to read the documentation line by line to progressively arrive at the method I was looking for. Furthermore, since documentation for modules are written in this style where methods are first general and then become more specific, each method you read bring you closer to the solution- the complete opposite of the mindless trial and failure I was following previously.

*Targeted Questions*

While working at Origin, particularly in reporting-based tasks it was very common that a stakeholder would ask for a view or some kind of functionality but did not know exactly what they wanted. This may sound ridiculous but to elaborate- they knew what information they needed, but not how it should be presented. Many times, I would hear questions like:

“Kai could you make me a view that gives the number of wins?”

“Can we have a pipeline view….?”

“Could we add a flag for that?”

“Can we filter by date?”

In none of these questions do they explain how it should be presented. This is important because tableau is a data visualisation tool, so the visual part a huge component of their request. Thus, how should one direct their work efforts? You could make a view, and have it be different to what the user expected though it may still meet their requirements. If we call the user’s expectation ‘Point B’ and our starting position, ‘Point A’ this workflow can be represented by the figure below.

Chart, line chart

Description automatically generated

Figure 4: Following an approach that may not meet the users expectation. Then one may have to make significant adjustments to move back to Point B.

This style of work has a huge risk associated with it; you may make something that is completely against user expectation, as I believe Issac did with his initial sustainability build. Instead, what I found my team did was using targeted questions. By targeted question I mean very specific questions on functionality, which allowed one to gauge the user’s expectation without them saying it directly. I noticed this in their reporting- when they completed a feature, they would go back to user, ask for feedback perhaps have a meeting and discuss any further changes. Then they would continue, an again ask and re-implement. This created a workflow more like the one shown below:

Diagram

Description automatically generated

Figure 5: The reporting teams’ workflow when developing an body of work.

This workflow is of high importance to me because it allowed my team to make a successful sustainability build. Any time I developed a key feature I would inform Kai; he would suggest changes if needed. We had weekly meetings with Mary, the project manager where she would also contribute with her own feedback. Additionally, Mary also staged meetings with key stakeholders- another opportunity to steer ourselves towards user expectation and even beyond. Thus, what I found was that using targeted questions allows one to navigate to Point B (user expectation) from Point A (the starting position) without being given a path from any external source.

*Structured code*

Once feedback is received on a developed feature, one must then implement the changes but how can this be done quickly? Not just quickly but comprehensively, exactly as the user pictured? Prior to my placement, I had heard statements similar to:

“Structure your code.”

“Create a function so you can reuse the code.”

“Structuring helps you debug.”

All statements I ignored with my scorecard build. In scorecard my html code was not well structured. By this I mean I had many repeated components, inconsistent styling and just unnecessarily lengthy code (scrolling up and down on an editor can be time-consuming and even confusing). This was important because the front-end will generally receive the most feedback from a user, so it’s important to be able to adjust it easily. This I found from my own experience; when Kai would ask for changes in scorecard, I would have to go through and change every file relevant to his suggested change.

However, when I worked on sustainability, I saw how Issac structured his code. He put code repeated frequently into classes- a practice I adopted since I wanted to start working as quickly as possible and this meant keeping as much of what was already made. Any time I made a key feature I would do a ‘code refactor’ which was just removing all the unnecessary pieces of code and putting functions in separate files. This may seem like a small change, but it had a drastic impact on my work output for sustainability. Now when a user suggested a change, I would only have to change that function that was relevant. I reached a point where sometimes I could adjust while on the call and I myself, despite being the developer, was amazed at how dynamic the application became with good code structuring.

*Soft Skills Inherent*

Though the above experiences were mostly from a technical background there was a soft skill embedded within all of them that I came to develop over the course of placement. Reading the documentation was about being able to ask for help and reaching out to a colleague. This was important because I was able to build a network of colleagues that could support me when troubleshooting and similarly, I could also help them. Targeted questions involved including the end-user through the development process using empathy to target the questions and gauge the end-user’s product expectation. This was particularly important – putting yourself in the user’s shoes made it much easier to direct feature development in sustainability. Simple front-end features like making the buttons on the ‘Add/Update’ page larger or the file upload functionality were developed by imagining we were the end-users. Furthermore, code structuring was about being able to receive and act upon feedback given upon features though the feedback could be given in a stern and forthright manner. Being able to acknowledge its importance in the context of the project and then implementing the feedback was a key soft skill I learnt through the medium of structuring my code. Thus, what I found was that soft skills allowed me to direct my technical focus. Targeted questions, acting upon feedback and being able to reach out to colleagues were all initiatives that gave insight on what I should be working on at any given point in time. This is going beyond how beneficial soft skills are for teamwork, developing good relationships with colleagues and settling into the work environment which are their most direct outcomes.

Key Acknowledgment: Never stay still

Through the SDC build, I faced many technical challenges. The most difficult were when I had to link my front and back end with sound coding logic or familiarise myself with new coding language modules (like ajax as I had previously mentioned). Despite all this, going from one technical challenge to the next, I felt a sense of momentum as I neared completion. Then I encountered a challenge, previously unknown to me.

At one point in the SDC build, my origin account did not have the necessary permissions to query a Jindabyne table, so I had to wait for access to a general service account. I was at a point where I literally could not continue working on the application which was highly frustrating because:

1. My momentum had come to a complete stop.
2. I had no control over when I would get the service account.

This showed me the project-based side of the SDC build where to increase its scale, I had to wait for the general account so that any developer could use to make adjustments to the application and if I continued without it, I would essentially be wasting time.

Since I could not work on the application, I did other related work. I knew that using git and bitbucket were skills I lacked so I worked on them over the week. It took a couple days for me to first get access to bitbucket and then familiarise myself with the commands. A week passed before I got the service account, and then I began working normally this time using bitbucket to push code changes to a repository I made. Now all this might seem unrelated, but later on in the SDC build, Mary was pushing for completion much sooner than I had expected. One day in late November she asked for three changes in a single meeting all regarding front end styling. One of the changes in particular was adding the tableau logo to the home page banner something which I had previously done and removed from my current version of the SDC.

However, because I was using git and bitbucket I had a commit on there that contained this feature, so I was simply able to go to that commit and copy paste the code relevant to the tableau logo. Implementing this may have taken another day or so but because I had spent the week where I was waiting for the service account learning git, I was able to do it instantaneously. In the end the otherwise idle week, became one of the primary reasons I was able to complete the sustainability build. Thus, I had learned that in a project setting, there was always relevant work to do which to me is surmised in the title of this paragraph- *never stay still*.

In Depth: The Sustainability Data Capture (SDC)

As mentioned previously my legacy project at Origin was the Sustainability Data Capture which was a web application that allowed users to directly input and review results for business metrics. At Origin the word ‘Sustainability’ is quite broad it can mean anything from actual carbon emissions-based metrics or even customer satisfaction-based ones. So, the application had numerous metrics to account for which would then be drafted and used in the annual Origin Sustainability Report (OSR).

The Origin Sustainability Report is a document published annually by Origin that gives an overview of all the initiatives taking place at Origin. It also details Origin’s performance in different business areas, which in the documents are categorised as ‘Customers’, ‘Communities’, ‘Planet’, ‘People and Culture’ and ‘Our reporting’. Though the OSR is not a legally required document, its publication serves two main purposes:

1. Anyone can use it to find out more about Origin as a business. This can include any potential investors or any government investigation, and
2. It acts as a certificate of operation, which shows that Origin is aware of it social responsibilities by reporting on its carbon footprint and contribution to local communities for example.

With the OSR being of such high importance the correctness of its reported values is crucial. Prior to the SDC the way these results were managed were a single spreadsheet which was sent back and forth to different data contributors. This had numerous issues some of which included:

1. Anyone could adjust the spreadsheet, so there were virtually no security measures.
2. There was no traceability- you could not see who inputted the results for metrics.
3. Each metric required approval from executives, the only way was to go through each result in the spreadsheet
4. Each result would manually have to be put into the spreadsheet.
5. Executives or metric owners may leave Origin leaving empty results (this was actually a big issue).

Thus, the main objective of the SDC was to fix all of these issues yet have a dynamic page that could be easily adjusted should new features need adding.

*Technical Summary: How it Works*

Like Scorecard the SDC uses a python module called Flask. Flask is a web development module, and we chose this module mainly because IBL students have experience with python, and it was already developed by Issac using Flask as well. Though Flask may sound quite technical, once its key concepts are understood it becomes much easier to work with.

When an application is made it can be hosted on a url. We did this with one of Origin’s in-house urls, so that when someone comes to the url an instance of the application is launched. Within Flask you can specify specific url paths or ‘routes’ for example:



So that when a user comes to this ‘route’ then the code below it is launched where normally a html is displayed at the end. That’s it. How does the user come to this route? On the html pages there will normally be a form tag such as:



The most important part is the ‘action’. This is saying “if the user submits a form, then go to the url ‘/index’”. Once we go to ‘/index’ whatever code is below this route will be launched. This process is repeated for any route, thus creating a web application that acts over different url paths.

*My role*

In the Sustainability build I was the sole developer. I created the whole front-end of the application, with feedback mostly from Kai and Mary. With the back end, I built upon what Issac already made but it required significant adjustment since I needed to implement the search functionality, the forms and the routes. The only pieces of his build I kept were the user validation and the models for the databases. Issac also helped me significantly when it came to the JavaScript code for the search functionality and some of the other front-end features that required jQuery (a module of JavaScript) to make the application more interactive.

Application Overview

Home Page

Graphical user interface, text, application, email

Description automatically generated

The home page had some key administrative information displayed. There were links to requesting access to Tableau, sending emails to the Operational reporting team and a guide to the csv file upload format.

Add/Update Result

Graphical user interface, application

Description automatically generated

This page was the main screen of the application. Here a user could search for a metric or upload a csv file for a bulk upload of metric results which removed the need for manually inputting multiple results. The search here was an important feature. Depending on what the user inputted in any field the rest of the fields would filter. So, if the user inputted ‘Person A’ for the data steward, then the metrics that had ‘Person A’ as a data steward would be the options in ‘Metric ID’. Once the user searched for a metric the page would show the metric’s metadata (on the left), history for that metric and the user could also input a new result. If the user did input a new result, it’s ‘Status’ would display as ‘Pending’ as it had not yet been approved and this record would be inserted into a Jindabyne table. Thus we were able to keep an audit trail of any inputted results along with which user put the result, which was previously unknown in the spreadsheet method.

Graphical user interface

Description automatically generated

Approve Metric

Graphical user interface, application

Description automatically generated

Like the Add/Update result page the ‘Approve Metric’ page also had the same search functionality. Depending on what the user inputted into the search the relevant metric results would be displayed along with their status. The user could also filter the search on the ‘Status’ which could be either ‘Pending’, ‘Approved’ or ‘Review’. Once searched different metric results would display and the user could either click the tick for ‘approve’ or the question mark for ‘review’. This workflow acted to simplify to approval of metric results. There was also the option to ‘Approve All’, should the user want to bulk approve results.

Graphical user interface, table

Description automatically generated

*Stakeholder Analysis*

From creating the search functionality in the SDC, I could see some of the names that could be used to search for metrics. The names I saw were truly Origin-wide of executive managers, group managers, team managers and analysts. The SDC makes it easier for these stakeholders to input results and verify their correctness. The metric results are initially input by data contributors, then approved by group managers before their final sign of by executives and general managers, following a progressive path through the company. The application also makes it easier for the Sustainability team who actually create the Sustainability report, since they can begin drafting their report once the results have been entered. Furthermore, the results from the report can be put into a spreadsheet, into any required format so that the Sustainability team can send it to Tangelo (an annual report producing software company) who further support the drafting of the Sustainability report. The results themselves can also be audited by consultancies, usually one of the big four and the connectivity of the input tables in the SDC the Jindabyne further support the auditability as the inputted date and time along with the approved date and time are also recorded by Jindabyne.

*Current Status*

At the time of writing the SDC is in production, and it is planned that it will be used in early January 2022. The main point of improvement for the build is to add functionality for calculated metrics. Many of the metrics used in the sustainability report are calculated from other metric results – the main aim is to be able to input the most basic metric results and the calculated ones could automatically be adjusted for their status and result values. A table has already been made for the calculated metrics all we require is the metadata on the calculated metrics from key data stewards so that we can then transfer this information to the relevant Jindabyne tables. Since there are many metrics and only a handful of people know this metadata, this manual process of transferring the metadata to Jindabyne may take a month or so.

Along with the data adjustments, the application may also have to undergo a security review by the Origin security team to ensure its complying with their regulatory standards. If the application comes to use single-sign on (like Microsoft authenticator for example) or it can be accessed from outside the Origin VPN then it will require security based improvements.

*University Learning*

In the computer science degree Monash offers, all of the core problem-solving based units are taught entirely in python. These units for me were FIT1045, FIT1008, FIT2004 and FIT3155. Since each unit goes for 3 months, I had virtually a year’s worth of python coding experience before my placement. Additionally, classes are taught in depth in FIT1008 which are extremely useful I think, in industry-based settings. Thus, in terms of just coding experience alone, I think university learning is quite comprehensive. This is important because I think it’s much easier to become a good programmer learning one language in depth before the next one. That way it’s more a matter of learning syntax and commands rather than having to learn key programming concepts, like classes and functions.

Aside from the core python units, I also did a unit called FIT2102 which taught JavaScript and functional programming. This unit was incredibly useful for me because it teaches you how to write clean code, which was especially important in the SDC build. When I applied the programming principles, I was taught here in the SDC, I was able to drastically reduce the amount of code I wrote which overall simplified the application. Furthermore, the programming languages taught here are completely different to python, so you again have more exposure to different types of programming.

From a database perspective, FIT3171 was crucial to my placement. FIT3171 covers database concepts and SQL which were the foundation of my team’s work. Looking at just SQL alone, there was not a single line of SQL code I read during my placement that I did not understand cause the language is covered so thoroughly through this unit. Furthermore, the unit also covers database concepts like entity relationship (ER) diagrams, relationship types and sequences which were all topics Origin actually used in its Reporting and Data Operations teams. It would not be uncommon to see an ER diagram in a meeting for example. If hypothetically, I had not covered this unit, I think I would not have been much use to my team even over the 5-month IBL period- it was that important.

In terms of what Monash could improve on, I think in all of the introductory first year units there should be at least two to three weeks covering git alone and in later units maybe virtual environments. Git and the usage of virtual environments are industry standard tools, especially git which in itself is like a programming language. It’s commands could be easily interwoven into the course content to make it’s teaching seamless (like for example, there should be at least one git revert command in your repository before submission).

**Final Words**

All through my IBL placement, there were many different experiences that I had through the reporting analyst role and the more software development based one I took on in the later stages of my placement. The key experiences that I had taught me practices that I could use to become a better IT professional from both a technical and soft skills based side. My legacy project will hopefully be used far beyond my placement and the link that Scorecard and the SDC have created between the user and Jindabyne will hopefully support the development of many more applications at Origin. Additionally, the units I had enrolled in at university greatly supported my transition to work at Origin and the skills I have developed, I will be sure to carry far into my career.