Work Placement documentation

Release 2022.5.26

Yi Ming Tan

Table of Contents

Introduction

1.1 Introduction

My name is Tan, Yi Ming. I am third year student from Malaysia, studying software development in MTU Cork. I am currently working as a software engineer intern at Qualcomm as part of the work placement module of the course.

To me, doing internships in general is extremely valuable as it helps to gain real-world software engineering knowledge and experience. And I am very fortunate enough to be doing my internship here at Qualcomm, one of the best and largest technology companies in the world. I honestly believe that gaining work experience during college provides a huge head start advantage. It allows me to correlate all the subjects I have learnt in college during the past two and a half years with actual real life scenarios in the tech world. It also prepares me for my future career by getting real meaningful work experience under my belt straight out of university.

This is actually my second internship with Qualcomm. I did a 10 week summer internship with Qualcomm during my second year of college from July to September 2021, fully remote. I absolutely loved it, I enjoyed my time working in the company, I learnt so much within the 10 week period and I am glad that I am doing my third year work placement here at Qualcomm again in 2022. I am super grateful to be given the chance to gain experience at Qualcomm during summer last year.

My current work placement duration started from the 4th of January 2022, and the end date is recently extended from June to September 16th. Mode of work is blended, I can either work remotely from home or in the Penrose Dock, Cork office. My line manager is Mohamed Wahbi, he is the one I go to if I have any questions about the internship, the work, career or life advice and many more. I also report to my project manager Kalpesh Shah from Bangalore, India, whom I work with in an infrastructure project team.

Chapter 2

Organizational Profile

2.1 About Qualcomm

Qualcomm is the world's leading wireless technology innovator. The company is based in San Diego, California, USA. It has offices all around the world, including Cork, Ireland. Qualcomm Incorporated operates as a multinational semiconductor and telecommunications equipment company. Qualcomm produces and sells semiconductor products in a fabless manufacturing model. It develops semiconductor components or software for automotive, smart-watches, laptops, tablets, smartphones, and other types of electronic devices. Nowadays, most people know Qualcomm for its Snapdragon chips powering mid-range to top of the line flagship Android phones and tablets available in the market today.

2.2 Qualcomm's Brief History

The company began in July 1985, created by 7 former Linkabit employees, led by Irwin Jacobs. The company was named Qualcomm which stands for "QUALity COMMunications". It started out as a contract research and development center. The company went public, filing an IPO in September 1991. When Paul E. Jacobs, son of Irwin Jacobs took over as Qualcomm's new CEO, Qualcomm focused more on research and development on projects related to Internet of Things. Then when Qualcomm announced that Steven Mollenkopf would succeed Paul Jacobs as Qualcomm's new CEO, MollenKopf expanded the company's focus on wireless technology for cars, wearable devices, and other new markets. Now, Qualcomm's CEO is Christiano Amon, he got appointed as president of the company on January 6th, 2021. The latest acquisition is the Advanced Driver Assistance Systems and Autonomous Driving software brand, Arriver.

2.3 Organization Structure

Qualcomm has a matrix structure in which the company operates based on vertical and horizontal relationships. Most employees in Qualcomm reports to two superiors: the functional manager and the project manager. Qualcomm Ireland moved its office from Mahon Point to Penrose Dock late 2020. Qualcomm owns three floors in one of the two blocks. Qualcomm Ireland currently has more than 600 employees, with a target to go beyond 1,000 employees by the end of the 2022. The workforce here in the Cork office is very diverse, consists of people from around the world. Even the new batch of interns this year alone is super diverse, consisting of people from all over the world. As far as I know, there are only 4 local intern hires from Cork out of around 30 new interns in this new batch. Qualcomm hires the best talents, regardless of their races, religions, gender, and nationalities.

2.4 Company Benefits

In Qualcomm Cork, there are a few benefits that also apply to interns as well. First, Qualcomm provides pension fund options for all interns. However, since there is no employer matching for interns unfortunately, I didn't apply for the pension. Interns are also eligible for paid time-offs, which is great.

Next, during covid-times, all employees are eligible to claim up to 25 euros back every month to subsidize for the internet broadband bill. I only applied for the claim in January. Also, all employees including interns can request for an office chair free of charge.

Furthermore, this could vary between different teams, but our team has frequent team events. This includes lunches, dinners, outdoor/indoor team building activities all paid for of course. This is great for us interns especially as it allows us to get to know more about our colleagues better.

Moreover, the company has a games room, filled with board games, foosball table, a huge TV, bean bags and more. I often hang out there with some other interns and colleagues during lunch break to play some video games.

Lastly, there is a huge and luxurious shower area in the office that is open to all Qualcomm employees. It has free lockers, hairdryers, huge mirrors, even a drying room to dry your wet clothes and towel. It has everything someone could ever need. I take advantage of this as I cycle to work. By the time I reach the office, I either get soaked in my own sweat or the rain, so being able to take a shower before I start my day is a huge plus.

Chapter 3

Description of Duties

3.1 Main Functions

I work as a software engineering intern, I work with an infrastructure team based mainly in Bangalore, India. Even so, this is still a multinational team, consisting of team members across India, Ireland, and the USA. All of us are working on developing an internal tool for Qualcomm engineers. This tool is a full stack app that comes with a GUI interface and a CLI option. I don't work on the front-end part, I work on the back-end of the project, mainly developing and testing APIs and CLI.

My scope of work in this team project is quite broad. I am responsible for:

1. New feature implementation with unittesting, good coverage and pylint scores

   I get tasked to implement new software features, new APIs, or just enhancing current APIs by adding more functionalities requested by our clients. Normally, before I write code, I will first come up with the unittests that are relevant to the task specification. This process is called Test Driven Development (TDD). I learnt this through a Python training provided by an engineer in Qualcomm. This will result in higher code coverage score because I am only writing code that is needed by the unittests to pass. Coverage is a measure, noting what parts of the code have been executed, mostly used to test the effectiveness of tests that I have written. Pylint is a code analysis tool to identify errors in Python code and helps developers enforce good coding style and habits. I always strive for the highest possible coverage and pylint scores when coding.

2. Debugging and fixing bugs

   I also do spend some time debugging and fixing bugs from the tickets I got assigned to. Some of the tickets are raised by us developers internally, while some others are raised by our clients from other teams. I also help out my team debugging together at times via pair programming, I believe two pairs of eyes normally discover the bugs twice as fast.

3. Setting up and the maintenance of our project documentation via Sphinx

   Before I joined, the project documentation is only hosted on confluence. It wasn't very detailed or tidy. So I took the initiative to suggest the team that we should use Sphinx to document our project. The team liked the suggestion and so we started documenting our code. Since then, Sphinx documentation is integrated into the release of our software, and will continue to be included for the future releases.

4. Refactoring and rearchitecting parts of the codebase to meet requirements change with good standards

   Sometimes, I get tasked with some code refactoring tasks. The point of these tasks are basically to improve code quality, readability, performance and maintainability. Some other times, refactoring is required because of change in requirements. In software engineering, requirements change all the time, and so must the software. When doing code refactoring, I always take it as a chance to improve the standard and the quality of the

code architecture. The team mainly codes in Python in this project, and in Python, there is this thing called the `Zen of Python`, which is basically a collection of 19 "guiding principles" for writing computer programs that influence the design of the Python programming language. The team is highly encouraged to follow the Zen of Python and any good software engineering principles in general as closely as possible.

3.2 Software/Technologies used

3.2.1 Development Environment

For my environment, I mainly connect and use the remote desktop located outside of Ireland. The remote desktop is a Linux system desktop machine, where all engineers share resources in a shared file system. For coding, I choose Visual Studio Code on my local machine as my preferred IDE. There is an extension in VSCode that allows me to connect to my remote desktop and code as if I am using my remote desktop. I have a ton of other extensions in VSCode that improves my productivity such as vim key binding plugin, intellisense plugins, python plugins and many others. For quick note taking or drafting code snippets, I use sublime text for it.

3.2.2 Project Tech Stack and Workflow

As for the team project I am working on, we use Python for the most part. Main components include: Python Flask for the GUI part, argparse library for the CLI, SQLAlchemy for the database management system. For version control, we use Git. And we use GitHub for our source code management. Our workflow is as follows: when we want to create a new feature or fix a bug, we create a feature branch based on the current development branch. When we have completed the feature or fixed the bug, we create a pull request to merge into the development branch. Here, we have some unittest checks automatically triggered to check if this new pull request will break anything. If all goes well in code review and the tests, it will be merged into the development branch. When a release is ready, we will merge the development branch into the release branch. This way, the release branch will always be stable, and not prone to bugs. We then will also tag the HEAD of the release branch and make a release on GitHub, along with all the release notes.

3.2.3 Issue Tracking System

Qualcomm uses Jira as the issue and project tracking tool. This is the main platform where we manage all our project related tickets. Whenever someone reports a bug, or want to start building a new feature, or have a query on something, Jira will be used to keep track of all issue tickets. We can give description, comment, assign an engineer, CC people that should be aware of the ticket, and many more.

3.2.4 Documentation

For documentation, our team used to use Confluence for all sorts of topics. However, we have transitioned to using Sphinx recently as it is the smarter and more efficient choice to document a project. Sphinx is a tool that makes it super easy to create intelligent and beautiful documentation. One amazing feature Sphinx has is the ability to automatically read the doc strings of the source code and generate documentation for it. Sphinx takes in plain-text files in `reStructuredText` format and transforms it into HTML, PDF, and any other format desired. The main selling point is that everything is automatically generated, there is little to none effort required to style the documentation.

3.2.5 Tests

For writing unittests, we use the python built-in unittest library to write all our tests. We also use pytest to run our tests, as they are both compatible to be used together. We also have a unittest base class set up, so whenever a new test needs to be written, the developer just needs to inherit that base class, and all the setUpClass and tearDownClass logic will be handled and taken care of.

3.2.6 CI/CD

For CI/CD, we use Jenkins to handle all our builds. We have set up nightly builds, which will run all test suites once every day to make sure everything in production and development is stable. We also set up a GitHub hook so that whenever a developer makes a pull request, a jenkins build will be triggered to run all tests. The team will receive an email on the result of the build, if the build passes, that means the Pull Request is safe to be merged, otherwise, there is still some work to do there.

Chapter 4

Account of Actual Work Experience

4.1 Initial Expectations vs How it Actually Went

Before starting this role in January 2022, I already know what team I will be joining and what project I will be working on. I did an interview with the team lead right before my first internship ended summer last year. The team lead explained to me what the project is, purpose of it and the progress so far. He also told me that this project is very important and impactful as it will be used by most engineers in Qualcomm when it is mature and ready. I was super excited to be joining the team of course. So I had a rough idea about what I will be working on, I was told that I will be part of the back-end team helping to build APIs with Python.

My initial expectations of this project is that I will be working with a group of very bright engineers, building something impactful for the company. I had also hoped that I can be a good contributor to the team, be a good team player, bring new ideas to the team, despite being an undergraduate intern.

In comparison to reality right now, my initial expectations were actually not that far off. I am indeed working with a group of helpful and kind geniuses. I am learning so much from them in such a small amount of time. Furthermore, I also do contribute to the team consistently and frequently, I am able to complete all tasks thrown at me. Whenever I get stuck, I know I can always reach out for my colleagues. Lastly, I did bring some new ideas to the team that they loved and actually implemented into the project. So, I am feeling good so far, and I am going to keep it up.

4.2 Critical Assessment of Achievement of PLAC7009 Learning Outcomes

4.2.1 LO1 Critically analyze the enterprise, its culture and organization

Qualcomm Ireland has a very good working culture as it stresses a lot on work-life balance. There isn't any sort of micromanagement in the workplace. It is flexi-hours, so as long as you don't miss any meetings and you can get work done, it doesn't really matter what time you clock in or clock out. People in Qualcomm are super nice, helpful, kind and always want the best for me. I especially love the culture in my team, where we have weekly technical tips sharing session which I find very helpful. Moreover, we also have a lot of social gatherings and team events such as team lunches, dinners, team building activities such as karting, hiking.

4.2.2 LO2 Communicate in a professional manner within the workplace

Good and professional communication is crucial in any team or any organization. I am a lot more confident about my communication skills now that I have conducted code reviews with other engineers, raised and proposed new different ideas to the team, gave training about specific topics to engineers, and many other occasions. Another important thing is that I have learned how to ask good

questions and when to ask them. Before reaching out to another engineer for help, I should have already spent a good amount of time and effort before first. If the problem still couldn't be solved, I would reach out to my colleagues and clearly explain the problem statement, the expected outcome, and the things that I have tried.

4.2.3 LO3 Demonstrate initiative and leadership skills whilst working alone and in teams

I am glad to say that I have taken initiative in doing something and actually paid off. I once proposed the usage of Sphinx to document our project since there wasn't any sort of concise code documentation prior. I took the initiative and spent some time studying and going through some trainings about Sphinx. After that, I prepared a quick demo and proposed it to the team. They loved it and integrated Sphinx into the project, and I have been taking charge the code documentation part ever since, reviewing other developer's documentation to make sure it follows the best practices and guidelines. I also constantly take the initiative to explore other technologies that might be beneficial to the team.

4.2.4 LO4 Apply knowledge, skills and competencies acquired during the program of study to the analysis and solution of workplace problems

In college, we are taught about a lot of core software engineering topics such as Object oriented programming, database management, design patterns, code documentation, unit testing, agile methodologies and many more. I am glad that we learnt these topics because a lot of what I learnt is applicable to current project that I am working on.

4.2.5 LO5 Reflect on and analyze the learning experience resulting from the work placement

Overall, I have learnt a lot so far in Qualcomm about software engineering through this internship. I have learnt about the importance of project code documentation, unit testing, good code coverage, code reviews, Continuous Integration/Continuous Deployment with tools like Jenkins. As well as how to write good and concise code comments, how to improve code readability, how to communicate with other team members in a professional manner, how to ask for help, how to debug code fast, how to propose new ideas/solutions, and so many more. Overall, these skills and knowledge are super important to obtain, and I am very happy to be learning from Qualcomm. These skills and knowledge are transferable and will be brought along with me to my future career.

4.3 Difficulties Experienced

It's totally normal and expected that there will be problems or difficulties that us interns will experience. One of the main difficulties I experienced was that most engineers here in Qualcomm speak in hardware/electrical related jargons and terms, especially during meetings with people from other department. Coming from a software background, I had trouble understanding what exactly they are referring to. I wasn't sure if I should spend some time asking and try to understand all the jargons they use or if I can get away without knowing exactly what they mean.

Another issue I faced is team project related. When I first joined the team project in January, I was spending a good amount of time reading the documentation of the project. However, I had difficulty understanding because of the sub-par organization and layout of the documentation. Some docs are outdated, so it's misleading, some docs are missing blank. That caused me more time than necessary to pick up and understand the project. I eventually understood more about the project by setting up meetings with colleagues and ask for their help explaining certain topics to me. And to make sure the next new-comer won't face the same issue as I did, I took the initiative and proposed a collaborative team effort to use Sphinx to document our project as detailed as possible.

4.4 Lessons Learned

4.4.1 Technical

1. Simple code is better than complicated

   When writing code, always choose the easiest and simplest option that will get the job done. Simple and clear to understand code will give other developers a better time reading and debugging it. Always Keep It Stupid Simple (KISS). I used to try to implement all the functionalities at one go, which in the end, some if not most of the functionalities become useless. What I learnt is that I should only add more functionalities when new demands arise. With this, I can achieve a lean development software.

2. If the implementation is hard to explain, it's a bad idea. If the implementation is easy to explain, it may be a good idea

   Code needs to be written in a way so that it is understable not just by the developer who wrote it, but also by other developers that maintain the code. If high-performance code is so complicated that is almost impossible for other developers to figure out, then it's probably bad code. On the other hand, just because a piece of code is easy to explain to someone else doesn't mean it isn't bad code either.

3. Special cases aren't special enough to break the rules

   When writing code, we are most likely following the system design the developers laid out before the project started. This means that there are project specific guidelines when writing code and we should follow them and not just code in any way we like although the end output might be the same. If you find your implementation can't fit the current system design architecture, you might be tempted to come up with some quick hack methods. However, this will only lead to inconsistent and unreadable code. At the same time, forcing yourself to adhere to rules and guidelines can also result to abstract and unredable code too. So, we need to learn how to walk the line in between.

4. Test Driven Development (TDD)

   TDD is a concept which tests are written first and then code is written in order to pass the tests. When all the tests pass, the code is then complete. TDD is an efficient process that ensures clean and reusable code. At first, I struggled to wrap my head around this concept as I am so used to writing tests after coding. After writing more tests in TDD manner, I found that I began to write more direct and efficient code more effortlessly.

4.4.2 Non-technical

1. Know when and how to ask questions

   When you are stuck at a problem, before reaching out to someone else for help, make sure that you have at least attempted to solve the problem yourself. When reaching out for someone, explain clearly the problem statement, the desired goal and show them your attempts to solve the problem.

2. Learn something enough to be able to teach someone else

   I feel like this concept applies to everything. If you want to be truly good at something, you need to learn it enough so that you are capable to explain and teach someone else in a simple language. In my opinion, being able to understand complex concepts thoroughly and explain something in an eloquent way are the two important skills required to become a good developer.

3. Don't be afraid to propose something new

   If you have an idea that you genuinely think that it can be beneficial to the team project,

you mustn't stay quiet. Instead, you should be confident and voice out your opinions and ideas to the team.

4. Have Fun!

   Lastly, I learnt that internships are meant to be enjoyable and fun. So, take advantage of all the resources and try to learn as much as possible, make new connections with people, and enjoy the time while it lasts!

Chapter 5

Student Profile and Relevance to Organization

5.1 Relevance between the role and degree program

The fact that I am a software development student means that I at least have the foundation level and understanding of software engineering. With that, I am able to dive into codebase and pick up quickly, identify what project architecture and design patterns used, be better at knowing what to expect in general. In team collaboration aspect, I can work with any team in an agile manner, meaning working in sprints, using ticket filing system such as Jira, attending stand-up meetings. In technical/programming aspect, I can contribute to the project with good quality code within short lead time, along with good documentation and readability. This is all due to fact that I learnt the core topics of software engineering in college such as OOP, the Python language, basic unit testing, code documenting concepts, and agile methodologies. In a nutshell, this role is completely relevant to my degree in college.

Chapter 6

Conclusion

6.1 Future Career Planning

Working as a software engineer intern at Qualcomm for these past few months has led me to realize that how lucky and fortunate I am to have chosen the right major and career path three years ago. I love what I do, I enjoy working as a software engineer, I love all the aspects of it, I really do. This current internship at Qualcomm just further deepened and strengthened my passion for software engineering. I am very thankful for Qualcomm for giving me this fantastic opportunity and experience. I now have no doubt about what I want to do in the future, my future goal is very I want to be the best software engineer I possibly can.

6.2 Last Words

All in all, I absolutely enjoy my time here currently in Qualcomm. This has been the best time and experience for me as a software development student here in Cork, Ireland. Even though it has only been 5 months, I know for a fact that I have learnt and grew so much. I am being taught the best practices of coding in Python and software development in general by the best engineers. Not only that, but I can also tell that my soft skills have improved significantly as well. I learned how to communicate with a professional manner, how to interact with other engineers and build meaningful connections and relationships not just here in Cork, but also in San Diego and Bangalore. I can confidently say that Qualcomm is a great place to work in, and I am going to keep it up, do my best and enjoy the rest of the internship!

Chapter 7

Monthly Report

7.1 Janurary

7.1.1 Progress of Placement Work Plan

Week 1

1. Received work devices such as laptop, peripherals, bagpack, etc
2. Setup work laptop, getting access to approprite lists and groups
3. Setup remote desktop access
4. Had initial meeting with line manager and the team here in Cork

Week 2

1. Had initial meeting with project manager and the project team
2. Had initial meeting with the rest of the other interns
3. Took some advanced Python courses on LinkedIn Learning
4. Did a few trainings about Company Confidential Information, Export Compliance, Covid Guidelines, and more

Week 3

1. Gained access to project repo on GitHub
2. Setup coding environment, installing IDE, required extensions, etc
3. Read project documentation, trying to understand more about the project
4. Explored project codebase, trying to understand the architecture

Week 4

1. Started 2 API enhancement related tickets, just to get my feet wet
2. Created a pull request, did code review with another engineer
3. Started another CLI format enhancement related ticket

7.1.2 Critical Learning Experience

1. The importance of protecting Company Confidential Information

2. The seriousness of company export compliance
3. New hires don't get task assignments normally at least after two weeks later, the first few weeks are for settling in

7.1.3 Reflection

The first month was pretty laid back. It was basically just setting up my work environment. Spent some time getting to know my new team members better, as well as reading project documentations, and diving into the codebase, trying to understand the project better. Overall the orientation experience has been great even though everything was done remotely.

7.2 February

7.2.1 Progress of Placement Work Plan

Week 1

1. Creating unit tests for the new feature enhancement I did
2. Understood how automatic Jenkins build for unit testing works for our project
3. API enhancement task pull request still open, needs some further edits
4. CLI format enhancement task on-going

Week 2

1. API enhancement along with the unit tests are completed, the pull request is merged into development branch
2. Did code review for the CLI format enhancement task with another engineer
3. Studied and did training for Sphinx documentation

Week 3

1. CLI format enhancement task is completed, created a pull request and is merged into development branch
2. Did refactoring on our database session handling with another engineer via pair-programming
3. Prepared a demo about Sphinx and showed it to the team, proposing to document our code with this tool

Week 4

1. Reported two critical bugs in the development branch, created tickets for them and fixed them
2. Done with the two bug fixes, created pull requests, and were merged into development branch
3. Started a new API enhancement task
4. Created automatically generated documentation pages for all our APIs

7.2.2 Critical Learning Experience

1. Writing unittests with the industry standards
2. CI/CD is important and necessary for all big software projects
3. Learnt the git workflow my team is using

4. Project documentation is important and learnt about Sphinx documentation
5. Importance of conducting code reviews with other engineers
6. Importance of good pylint and coverage score

7.2.3 Reflection

I would say that February is the month which I really start to pick up some new topics. I feel more integrated into the team since I started working on some tasks, and doing code reviews with other engineers. I feel good as I now understand more about the project and also starting to contribute to the team.

7.3 March

7.3.1 Progress of Placement Work Plan

Week 1

1. Created another ticket for a failed unittest, due to a recent update in a submodule dependency
2. Fixed the failed unittest, created a pull request, approved, and merged into development branch
3. Last API enhancement task completed, did code review with an engineer. Created pull request, approved and merged into development branch
4. Proposed to team that we use Rest format for our doc strings, team agreed
5. Created CLI manual page via Sphinx, the manual is also automatically generated
6. Explored sphinx-multiversion to enable multiple versions of documentations to be in one place

Week 2

1. Added more unittests for the recent API enhancement task
2. Proposed a more standard and easier way to do setUpClass and tearDownClass for our unittests
3. Added some new functionalites to an API to handle new scenarios
4. **Gave a training about Sphinx documentation to engineers from another team, topics include:**
    - sphinx-quickstart
    - document customization and configuration
    - autodoc
    - doc strings
    - doc versioning

Week 3

1. Started updating the doc strings for our API module, as the previous doc strings format used, is not recognized by Sphinx
2. Completed update for doc string of the API module, created a pull request, approved and merged into development branch
3. Generated PDF version of Sphinx doc
4. Completed a new ticket for a new API feature, code review meeting is set as well

7.3.2 Critical Learning Experience

1. Learnt how git submodule works
2. Gained confidence into proposing something new to the team
3. Learnt how argparse library works and how to fully customize the CLI

7.3.3 Reflection

March is the month where I first introduced a new major idea to the team which the team adopted. I am very happy that the team liked the idea and decided to integrate it into the project. This might be the first moment where I am truly impactful and giving good contributions. March is also the month where all engineers are allowed to come work in the office. I love working in the office as I can interact with other interns and engineers in person.

7.4 April

7.4.1 Progress of Placement Work Plan

Week 1

1. Did an early release (User Acceptance Testing phase)
2. Hosted our project documentation in our internal site, available to all engineers to view
3. Started a new code refactoring task
4. Did some study and came up with a solution to log our API easily using a logging wrapper

Week 2

1. Conducted code reviews with two engineers about their new features
2. Started working on writing unittests for a new feature coming up
3. Did some documentation review with other engineers

Week 3

1. Reported two major bugs in the development branch, created tickets for them, and fixed it
2. Created pull requests for the two bug fixes, and were merged into development branch
3. Created pull request for the new unittests, was merged into development branch
4. Did the official release

7.4.2 Critical Learning Experience

1. Learnt the concept of software pre-release and an official relase
2. Learnt that clean code doesn't really exist, engineers are always cleaning dirty code
3. Learnt more about the concept of logging and decorators

7.4.3 Reflection

Our team released a new version of the software this month. I now realize even more how much spotlight this project has. When we sent out the official release notification email, we received so many

kudos and congratulations emails. I feel really good working on a project that has so much attention.

7.5 May

7.5.1 Progress of Placement Work Plan

Week 1

1. Got assigned to a major and critical code refactoring task
2. Implemented unittests for the code refactoring
3. Updated an API that was deprecated
4. Fixed an object cloning logic and some API filtering issue, implemented more robust test cases for it
5. Configured the log level of a submodule dependency to silent it some log messages
6. Created a script to auto import modules upon creating a new module in the same directory
7. A bug was reported regarding the test suite I created last week, started investigating into it

Week 2

1. The major code refactoring task is done, as well as the unittests for it. Pull request created, waiting for review
2. Found the problem that was causing the failed unittests from last week. Started work on the fix
3. Fixed the failed unittests, created a pull request, approved and merged into development branch
4. Doing some research into Python Click, could potentially be more lightweight and efficient compared to argparse

Week 3

1. Setting up Sphinx documentation for the upcoming release. Implemented multi-version mechanism
2. Improved the unittest coverage score up to 97% for the major code refactoring task
3. Continued doing more research into Python Click

Week 4

1. The major code refactoring task is completed, pull request is also reviewed and merged into development branch
2. Did another major code refactoring task, it's basically to convert the large module into a package of multiple modules
3. Both major refactoring tasks are completed and merged into development branch

7.5.2 Critical Learning Experience

1. Understood more about the logging mechanism in python
2. Learnt how to propertly write good code with high coverage and pylint score
3. Learnt more about the new python click libary

7.5.3 Reflection

May is the month where I truly feel more comfortable with the codebase, and more confident working on tasks by myself. I am farmiliar with the codebase architecture way more compared to when I first joined. I am not afraid to speak out my mind, my ideas, opinions on topics. I find myself more in important discussion meetings. I almost feel like I am one of them now, and not just an intern anymore.