**ANU TechLauncher Survival Guide**

**From a Previous Student**

Documentation by Phillip La

**Table of Contents**

**Introduction …………………………………………………………………………………**

**How TechLauncher Works ………………………………………………………………..**

**Responsibilities of the Team ……………………………………………………………...**

**Project Manager …………………………………………………………………….**

**Product Manager ……………………………………………………………………**

**Risk Manager ………………………………………………………………………..**

**Integration Manager ………………………………………………………………..**

**Test Manager ………………………………………………………………………...**

**How to Thrive in TechLauncher ………………………………………………………….**

**Importance of Process …………………………………………………………….**

**Emphasis on Documentation …………………………………………………….**

**Managing Destructive Behaviour ………………………………………………………..**

**Distribution of Contribution Marks ………………………………………………**

**Low Productivity ……………………………………………………………………**

**Faux Miscommunication …………………………………………………………..**

**Hostile Behaviour …………………………………………………………………..**

**Managing Relations with Tutors and Examiner ……………………………………….**

**Managing Relations with Clients ………………………………………………………...**

**How FireApp Should Run Each Sprint ………………………………………………….**

**Introduction**

The ANU TechLauncher has been one of the most rewarding courses that I’ve taken at ANU as a software engineering student. This course was challenging in ways that I had not anticipated, and I hope that you are able to extract as much of the important lessons from this handover documentation. Please do not skip reading this document, as it will save you much headache that new teams almost always suffer.

I worked as the project manager of FireApp for two semesters of 2022, and within this period, we had three different tutors. We also won the Best TechLauncher Team Award at the first semester showcase.

This documentation will give you advice on getting marks in the course as well as different techniques that I’ve gathered in managing a team of diverse students.

**How TechLauncher Works**

This section of the documentation will introduce you to how TechLauncher works as a whole. I will go into more detail after this section.

TechLauncher is a two-semester long course worth 12 units. Students may start the course either in semester 1 or semester 2. However, they must finish two semesters of the course in order to pass it.

Students may not change their teams in the middle of the course, including after one semester. However, students may drop out of the course after one semester, in which case, they would fail the course.

The course consists of a student software engineering team of 5-7 students who will be paired with an industry client. The student team will also have a dedicated tutor who will assess their work throughout the semesters. The tutor will report to the examiner who will manage assessment marks. The team will have another group of student software engineering team in the course, called shadow team, who will attend the same tutorial weekly. All the people described above are called auditors of the team.

This course is fast-paced and though the clients and tutors provide feedback, the project manager should still be broadly knowledgeable of Agile project management from the start of the course. The team should be ready from the start of the course to pass.

**Responsibilities of the Team**

The team’s responsibilities should be divided into clear and distinct roles. Having distinct roles from the start is necessary as it will be assessed by the auditors. It is important that students who take on these roles are motivated and proactive in their role.

**Project Manager**

The project manager is responsible for leading the team to the project’s successful completion. It is recommended that the project manager is also one of the spokespeople in the team. The following are some of the project manager’s responsibilities:

1. Project finishes within schedule.
2. Project has features and quality that please the clients.
3. Project has features and quality that please the tutor(s) and examiner.
4. Team is high performing, and contributions are fairly distributed.
5. Managing the expectations of the clients, tutors and shadow team.
6. Preparing meeting agenda and taking meeting minutes.
7. Being one of the main communication channels for auditors.
8. Team is on track to receiving desired audit marks.

Please view the *Overall Process* section of the documentation, which shows the process the team should follow. The project manager is responsible for managing most of the process in the diagram.

**Product Manager**

The product manager is responsible for deciding on the final features of the product. They are the single point of truth regarding what the end-users want from the product according to interviews and research that they hold. The following are some the product manager’s responsibilities:

1. Interview end-users and discover the project’s value proposition.
2. Organise and groom the user story map regularly with the latest user stories and detailed acceptance criteria.
3. Ensure IP Agreement is signed by involved parties.
4. Research competitor products and ensure the project’s product has an advantage.

**RIsk Manager**

The risk manager is another one of the most important roles in the team along with the project manager and product manager. The risk manager ensures the following:

1. Risk log is updated with the latest risks ***regularly***.
2. Items on the risk log have only one owner (person in charge of managing a particular risk item by due date) and due date.
3. Items on the risk log are being reminded to relevant team members during meetings and track their progress in managing it.

**Integration Manager**

The integration manager is responsible for the overall software architect of the repositories. The integration manager should be knowledgeable of the main technologies involved in this project, and be competent in AWS DevOps. The integration manager’s responsibilities are:

1. Ensure all the different repositories of the project are in sync.
2. Ensure all different components (frontend, backend, scheduler) of the codebase are successfully integrated with each other.
3. Ensure product is being successfully deployed and tested.
4. Ensure the project's security is not being compromised. ***(important)***

**Test Manager**

The test manager is responsible for the overall testing of the codebase and its product. The following are the test manager’s responsibilities:

1. Ensure the team has a clear template to follow for testing. See Kenneth Young for more advice.
2. Ensure all features of the codebase are being tested according to the acceptance criteria for each user story.
3. Ensure testing documentations are being created and managed on Confluence.

**How to Thrive in TechLauncher**

Many students who take this course for the first time have misconceptions about TechLauncher. Usually students who come from a computer science / software engineering background mistaken that this course is similar to their previous COMP courses. That is, they do the following:

* Try to organise a team with different roles.
* Try to hold meetings, some impromptu through messages. Maybe do some meeting minutes that summarise an hour long meeting with 9 people in only two sentences.
* Some team members go “missing”.
* Have a vague, ever-changing idea of the final product.
* Forget about the project and start rushing one week before it is due.
* Discover that the desired product cannot be delivered in time, so make last minute changes.
* The team leader (and their friends) panic and blame the whole team that they were the only ones working on the project.
* Did not document anything. Maybe start documenting at the end of the project after the codebase is implemented.
* Distribute contribution marks, giving a bit more to people who listened and a little less to people who did not impress you.
* Have disagreements among the team about contribution marks and get the tutor involved.

***If you do the above steps (which students have done for most of their past projects) in TechLauncher, you will fail.***

**Importance of Process**

The main goal of the team (especially the project manager) is to ensure that the team follows a process. That is, the team should aim to get to the *Norming* or *Performing* stage of the *Tuckman’s Model of Group Development* as soon as possible.

This means that eventually, the project manager’s duties become smaller and other team members take on regular scheduled tasks without constant reminders from the project manager.

This is especially important for the project manager. If this process is not set early in the course then the project manager will be massively overworking (especially if they have to implement the codebase as well).

**Emphasis on Documentation**

It is imperative that the team documents their work including their progress. The auditors will only mark on what is recorded in the team’s documentation. It is difficult to do enough documentation for most first time teams.

The following are the different forms of documentation:

* Confluence documentations
* Technical documentations
* Testing documentations
* Meeting agenda and minutes
* Calendar showing schedule
* User story map
* Jira kanban board
* Jira sprint burndown chart
* GitHub repositories showing successful branching, merging, commenting and code reviewing.
* Client value proposition map
* Project overview showing Jira roadmap, table of features and acceptance criteria.
* Emails cc’ing relevant auditors

The auditors (especially the tutor) are not like previous courses. Even if the team has followed a process, if there is no documentation, the tutor will still significantly mark the team down. The above listed documents are the team’s evidence for rightfully receiving marks.

**Managing Destructive Behaviour**

Managing destructive behaviour is one of the most important tasks in TechLauncher. People will not always have the same motives and level of motivation. Understanding that team members cannot be forced to have the same level of motivation as you is important in order to maximise your team members’ satisfaction.

**Distribution of Contribution Marks**

It is imperative to distribute contribution marks fairly. The examiners of the course, especially Charles Gretton, will find it suspicious if everyone in the team is given a high contribution. What is considered *fair* in TechLauncher will be very different from previous computer science courses. Compare the two contrasting ideas of fairness in the following scenarios.

**Fair in earlier COMP courses**

I think that student A worked hard, therefore, I will give them a 5 out of 5 for contribution. Though the assignment was difficult, I got through it in one piece, and I feel generous.

**Fair in TechLauncher**

I have evidence that student A has worked on the message feature, which is one of the most important features of the application. Student A has also worked a total of 15 story points. I however, worked 18 story points in the same sprint, therefore, I deserve 5 out of 5 and student A deserves a 4 out of 5 as they have done comparatively less work than me.

As you can see, contribution is comparative in TechLauncher; which makes more sense than giving everyone the same contribution mark. Afterall, the final group marks are not the result of equal contribution from students, regardless of how hardworking everyone was. Someone working harder than another person probably must have had a greater impact on the mark. It is not compulsory that contribution marks are distributed as in the second scenario. However, not doing so runs the risk of scrutiny from the examiners for collusion, but it also means that team members who play an important role and work hard will be less willing to do so; which is detrimental for the whole team. As a team member, and especially as the project manager, it is important that contribution is fairly distributed.

**Most Common Biases in TechLauncher**

Assuming that team members do not have ulterior motives, the two most common biases in TechLauncher are the Dunning-Kruger Effect and availability bias. The following define and outline some examples of the biases:

Dunning-Kruger Effect: Perceiving an event/concept to be simplistic due to one lacking knowledge about the event/concept. For example, team members may mistake the work of preparing a meeting agenda and taking meeting notes as simply summarising what’s spoken in the meeting.

Availability Bias: Making evaluations based on information that was the most available to the person. For example, a team member may give more contribution marks to a team member that directly helped them during the sprint, or a team member that they reviewed the code of.

**How to Distribute Contribution Marks Fairly**

There are different ways of fairly distributing contribution marks. It is ultimately up to the decision of the team, and what matters is that there is a clear system for distributing. One system that previous teams have been using is the story point system. The more story points a student has done, the greater contribution mark they deserve.

It is also highly recommended that team members have one mandatory meeting before the audit contribution statement link opens. In this meeting, each team member should go around in a circle and “promote” the work that they have done and the story points that they have completed. This is not to directly influence another team member’s assessment, but it prevents availability biases. One common problem in TechLauncher is when students falsely give more contribution marks to other team members who they had a direct connection to during the sprint. Another scenario is when a team member gives more contribution marks to a student that has helped them implement their user stories. Holding this final meeting allows everyone’s work to be at least acknowledged for a fairer assessment.

**Low Productivity**

Low productivity is a major concern during TechLauncher. Even if tasks are distributed fairly, there is a tendency for students to prioritise other courses over TechLauncher. This commonly results in team members seriously starting the implementation of their user stories a couple days before the end of the sprint. One method for possibly managing this is for the project manager to ask team members daily for their progress. It is even better if the team is willing to meet every day for 10-15 minutes for daily scrum, where they ask the following questions for each student:

1. What have you completed since the last scrum standup?
2. What will you get done until the next scrum standup?
3. What are the impediments stopping you from achieving this goal?

This could be micromanaging, but I highly recommend that the project manager considers all different methods for managing idleness. Again, if the team does not agree to such micromanaging, then a simple questioning on a communication channel like Slack should do the trick.

**Faux Miscommunication**

Miscommunication is a real issue in TechLauncher. However, sometimes team members may abuse miscommunication to blame their low productivity. For example, a team member could state that they have not done a task because of miscommunication. However, if there are detailed meeting notes, then it is also the responsibility of the team member to check back at the notes for clarification. I highly recommend the project manager does mutual decision-making and makes public records to keep everyone accountable and on the same page.

**Hostile Behaviour**

Some team members could display hostile behaviour, such as name-calling, shouting, leaving meetings without notice etc. Such behaviour is unacceptable, and the student on the receiving end must act wisely. Speaking with the perpetrator student and coming to a mutual understanding should be the first choice of action. For example, gently saying “I do not appreciate your language… We can hold a mature discussion when you are ready.”

**Managing Relations with Tutors and Examiners**

Tutors report back to the examiner during the semester. The tutor is an interesting role in TechLauncher where they are there to support you but at the same time, they look for red flags to mark you down. Your tutorial is a time to ask questions, feedback and more importantly, promote your progress. It is not a time to simply display all of your team’s dysfunctions.

**Managing Relations with Clients**

The clients also play an important role in making sure that you are supported as a team, for example, access to people in the industry for requirements gathering. It is important to manage client’s expectations, i.e. do not make false promises on the project, and if you do, then talk it out with the clients and come to an agreement to change your roadmap. The team should show their progress each client meeting to give the clients a sense of security and understanding of the team’s process.

**How FireApp Should Run Each Sprint**

I highly recommend that the project manager uses an online calendar (e.g. Google Calendar) to divide the entirety of the semester into 2-week long sprints. The project manager and product manager should also collaborate when creating a user story map with swimlanes dividing user stories into 2-week sprints corresponding with the calendar. The following are some of the activities that each sprint must at least have:

**A day before a sprint:** sprint planning meeting where team members are free to commit to user stories in the corresponding swimlane on the user story map. Story point estimates should also be finalised here using methods such as Planning Poker. The sprint planning meeting should be held at a set time and date every new sprint, and team members should be required to view all the user stories with acceptance criteria,, before the meeting, to decide which ones they may commit to during the sprint planning meeting.

**During a sprint:** Daily standups or the project manager asking simple questions to check up on progress for each team member. Also, holding a client meeting at a set time and date every week. The project manager should regularly use burndown charts and kanban boards on jira to track progress of each team member. Test engineers should regularly test as each user story is completed during the sprint.

**End of sprint:** The project manager should compile and document all the story points and tasks that were completed by each team member in a table on Confluence. Then, the team should reflect on their contributions in a sprint retrospective meeting. The sprint retrospective meeting should be held at a specific time and date.