Assignment 1

Project Inception

Due: Friday 4th September 2020, 11:55 your local time.

This is the first of four assignments that go together to comprise a major project. You will be making decisions that will govern how your team will go about the process of constructing software: you'll decide on the language and tools to use, you'll decide how to keep track of the time you're spending, and you'll decide how to communicate your progress to stakeholders outside your team.

If you make good decisions, you'll find it easier to implement your project later on. If you make poor decisions, you *will* be able to change your mind later – but this may come at the cost of having to re-do your work.

The problem: A student time/task tracker

Your team is required to implement a piece of software to help Software Engineering demonstrators mark work that has been done in groups – the technical term is *groupwork*.

It can be very challenging to mark groupwork. It seems easy enough at a superficial level: simply look at the product and give it a mark based on how good it is. Unfortunately, this model breaks down when we have to deal with students who do not contribute their fair share to the project. One of the most important aspects of ensuring fairness is keeping track of how much time each student spends on the group task, and also what tasks they have been working on. Helping your marker to handle these tricky situations will mean less stress for them, and probably fairer marking for their students.

Such a system would need to be easy for students to use, so that they are more likely to put their times in early rather than put it off or not do it at all. It would also need to be able to report time and task information accurately to the marker, and also to the team – they might be able to use the data to improve their work practices, and of course they've got a vested interest in making sure that it is accurate.

Your client will be one of your lab demonstrators – someone who would be expected to use a system like this in their day-to-day work. You will need to ask them for more information about the requirements. Please note that each team is likely to end up with a different set of requirements – this is expected, and more realistic than having all teams simultaneously implement the same system.

Again, make sure you are careful with your initial requirements: *it is not okay to assume that they don't matter, just because they haven't been written in this document.* One of our objectives in this unit is to give you some practice with eliciting requirements from clients and negotiating with them about the scope of your project, and you don't get to do that if you're handed the requirements up front.

Your team

You must do this assignment in a team of four or five students. Team members must all be in the same class, in order to prevent confusion over who is supposed to be marking which team. You should have been placed in a group already, but if not, please arrange for one as soon as possible – you may post to Moodle seeking a team if you like.

Teams should be formed and finalized by the end of Week 3, and repositories will be created for you at this time. You should use this repository to store your source code, any data you need, and your process documentation. You will be expected to use git in a reasonably sophisticated

way, and will be marked on this. We posted some reference material on git and GitLab to Moodle in Weeks 1 and 2, and there were some lab exercises on these topics in those weeks as well.

Your client

Your workshop facilitator is your client representative, and should be the source of your requirements. If you are in a workshop that has two facilitators, they will divide the teams up between them so that you don't have to deal with the hassle of having two client representatives with potentially conflicting wishes[1]. **Do not assume that the lack of detailed requirements in this assignment spec means that you can safely make them up without consulting your client!**

It is possible, indeed likely, that different facilitators will have different preferences and priorities. It is also possible (although not required) for different teams to end up with different product backlogs. Your team should negotiate with your demonstrator to come up with a project scope that you all agree is both feasible to implement and valuable to users.

The constraints

Your client and their organization do not care what language you use to implement your program. However, they are not in a position to provide you with any extra hardware or software – if you want to write a web application, you'll need to arrange for a server yourself[2]; if you want to write a phone app, you'll need to either target a phone you already own, or test your software on an emulator.

Another limitation is the software available to you. Since you'll primarily be developing on your own laptops or home computers, with the Monash student lab off-limits, you might not be able to get hold of your favourite software package. Even if it's commercial software that you have bought for yourself, that doesn't mean your teammates will necessarily have a copy.

Because this project is part of your assessment for FIT2101, the teaching staff have an additional requirement for you: we need you to keep track of who has done what on the project, and how long they have spent doing it. This includes both programming tasks and non-programming tasks – don't assume that tasks like management, quality assurance, and design are trivial! (You'd be paid for the time you spend doing these things in industry, so they clearly contribute to your project's bottom line.) We'll be looking at this information later to see how much time this project takes, and also to help resolve any disputes that arise over workload or contribution within each team.

We expect you to use a variant on Scrum for this project. We will be evaluating your product in a product review to be held in your lab time in **odd-numbered weeks**.

Deliverable 1: the Project Plan

You must develop a Project Plan to document your team's policies. This document will list the "rules" that your team will be expected to follow during development. In it, you should explain:

- your project's vision
- who your team members are, including
  - how teammates should contact them
  - their roles and responsibilities

[1] Of course it is *definitely* possible for real-world projects to suffer from this problem – but we don't want to inflict that on you in second year!

[2] The git server that Software Engineering students use in ENG1003 is configured to allow students to deploy to a web server that Monash provides, but the shared server you'll be using is not.

- an explanation of your team's process model[3]
- your team's Definition of Done – a checklist of things that have to happen before a feature can be considered "done" and ready to ship (including any review or testing that your team requires)
- how and when you will run the Scrum Ceremonies, and what you will do in them:
  - Sprint Planning
  - Daily Scrum
  - Sprint Review
  - Retrospective
- how your team will allocate tasks to team members
- how your team will keep track of progress on your project
- how your team will store and manage your backlogs
- how your team will keep track of time spent on project tasks

Your Project Plan is a living document: we will expect you to update the information it contains if your team decides to change the way it operates. Make sure you keep it in a format that allows it to be easily changed.

As you are developing your rules for tracking progress and tasks, bear in mind that one of the reasons we do this is so that team members will be able to prove that they have been contributing as agreed to the project.

You may use online tools such as the GitLab issue tracker, Trello, or Google Drive to help you with these tasks if you like. If so, please make sure that you give your workshop facilitators access to them too, and that you include links to them in your repository's `README.md` file.

Note: Project Plans are covered in lectures and readings in Week 3.

Deliverable 2: Analysis of Alternatives

First, your team needs to make some important **architectural decisions**. You'll need to decide on a programming language, and you'll need to decide on a platform – will you write a web application, a desktop application, or a mobile application? Which operating system/s or browsers will you target? These are important decisions that will have a severe impact on your project's success if you need to change them later on, so we want you to write an analysis of alternatives so that you can show you have considered them thoroughly.

To do a good job of this analysis, you will probably need to do some online research, and you may also want to build a *spike* – some throwaway proof-of-concept code to see whether a particular approach is feasible. For example, if you are considering JavaScript as a language but some of your team haven't used it before, you could have them write a "Hello, World" program in it.

When considering your alternatives, please make sure that you consider the cost of any service you consider. For example, if you decide to implement a web application that must be hosted online, you'll need to find a hosting service.

Secondly, you must decide how your team will **communicate and collaborate**. This will be especially challenging this year, since it is unlikely that your team will be able to meet face to face for a long time. Many systems are available, including voice, text, and video chats. It is likely that

[3] Although we ask that you base your process on Scrum, there are many features of Scrum that will not be practical for your team. You'll need to explain which features you will cut, and what you will do instead.

most of your development work will take place outside class, so you will need to select tools and devise strategies that will minimize barriers to communication within the team.

You may examine as many options as you like, but you must include *at least* the following:

- Zoom – `https://zoom.us/`
- Microsoft Teams – `https://www.microsoft.com/en-au/microsoft-365/microsoft-teams/group-chat-software` (note: Microsoft Teams is free for Monash students)
- Slack – `https://slack.com/`

You may prepare more Analyses of Alternatives if you feel you need to, but you *must* at least write one for your system architecture (i.e. toolchain and platform) and one for your communication tools.

Note: Analyses of Alternatives are covered in lectures in Week 3.

Deliverable 3: Risk register

You must write a risk register, listing the risks that may affect your project's successful and timely completion. For each risk, you must list:

- its estimated impact and likelihood
- a monitoring strategy
- a mitigation plan

You can integrate your risk register with your Project Plan if you like.

Bear in mind that good risk management will not only improve your chance of successfully completing your project, it could also be worth marks. If you have good policies for monitoring and managing risks, and you follow these policies, your team can be compensated for marks that would have been lost as a result of your identified risks coming to pass.

Note: risk management is covered in lectures and readings in Week 3.

Submission

You are expected to store your project documentation either in Google Drive or in the project repository we will create for you on `http://git.infotech.monash.edu`, alongside your project source code. This will make it easy for all project members to access the current version, and to change it if needed. There is no separate submission required; simply make sure your project documentation is up-to-date. Your marker will check out a snapshot of your master branch at the due date and time.

We recommend that you use text files in markdown (`.md`) format. GitLab uses kramdown for this, which is documented at `https://kramdown.gettalong.org/quickref.html`. Markdown is easy to learn, and will let you create attractive and useful READMEs for your GitHub or GitLab work portfolio, if you choose to create one.

If you use third party systems to help organize your project, including Google Drive as well as systems such as Trello or Asana, you must grant access to these to your marker, and must include a link to your material in your `README.md`.

If you choose to use Google Drive, you must make sure that your `README.md` contains links to each of your files, and that your files will be both readable and editable[4] by all staff members.

[4] Edit access means we can see the edit history and verify that all team members have contributed.

This is what we expect you to deliver:

- a Project Plan
- two or more Analyses of Alternatives to justify your choices of communications tool and system architecture
- a risk register, which may be integrated into your Project Plan

You are *not* required to write a Software Quality Assurance Plan (SQAP) as described in FIT2107, but you can if you like.

A marking scheme for this assignment has been posted on Moodle. Please refer to it in order to understand what your marker will be looking for.

Appendix: Iterations 1-3

It may help you plan your project if you know how it will be assessed later.

You will be assessed on the quality of the software you produce, but most of the marks will be assigned to an evaluation of your process. This means that you should have a well-defined process specified in your Project Plan and carry it out in a disciplined manner. You are of course allowed (and encouraged!) to modify your process if you decide to do so during a sprint retrospective, but if you do this you'll need to make sure your Project Plan is kept up to date.

Marks for your process will depend on how good your plans are for process, risk, task, and time management, and how well you have followed them. This means that you should try to design a process that is not too complex or burdensome to follow.

For full marks on your product, you will need to implement all of your client's "must-have" functionality to an acceptable level of quality. If you identify and implement some additional functionality, or *stretch goals*, you may be eligible for bonus marks – *if and only if* these stretch goals were implemented using a disciplined and sustainable process.

The functionality of your product will be evaluated by a *product review*, including demonstration, at the end of each sprint. This should take place during your lab, and you'll be expected to show your demonstrator the new features you have completed and merged into master since the previous iteration.

Note: This information is here to help you develop your Project Plan. You will *not* be required to do a software demonstration as part of your project inception – you're welcome to do a coding spike if you wish, but you don't have to.

Marking schemes for the three iterations will be posted online alongside their respective assignment sheets.