Project Sprint 1

I hope everyone has fun and gains a lot of valuable experience. The Product Owners, TAs and I are here to help you achieve both of these goals. In class we talked about how an Agile team works on a software development project: the process, the roles, etc. The project is where you put all this knowledge in practice. This document outlines how your work will be assessed and what is expected from Sprint 1 of your project.

Product Owner

If you are working on an iCube@UTM project, your project owner is the company. If you decide to have your own project, the product owner will be your Scrum Demo TA (You will get announced later).

Project Planning

Include all aspects of your project plan. Among other artifacts, we will definitely want to see:

- The release planning meeting document `RPM.md`, document the Release Planning Meeting during your tutorial. This document has to indicate clearely the release goal, the scope of the project (at least in terms of epics/key features) and the participants.
- Once you complete the Release Planning Meeting, you should also do the Sprint 1 Planning Meeting (during tutorial) documented in `sprint1.md`. This document must clearly indicate the sprint goal, all stories for this sprint clearly identified, team capacity recorded, participants are recorded, decisions about user stories to be completed this sprint are clear, tasks breakdown is done.

Project Tracking using Jira

See marking scheme.

Standups

See marking scheme.

System Design

The System Design document must be prepared in a format that we can read (PDF, MS-Word, md, html) and it must be stored in the same folder `doc/sprint1`.
You are probably not seasoned developers (yet!), and you will likely learn and use a new technology in the course of the project. It is therefore likely that the system design you provide here will undergo major changes as you work on the project. Don't panic! Get together and spend some time brainstorming. You will also receive feedback on this from me and the TA.

- Include a high-level description of your classes using CRC Cards: what they are, what their responsibilities are, and what is the interaction between them.
- You can use the following template for the CRC Cards:

```
Class name: Classname
Parent class (if any):
Classname Subclasses (if any): List all the subclasses separated by a comma.
Responsibilities:
*g
*h
```

```
*i
Collaborators:
*j
*k
*l
```

- The description of system interaction with environment should indicate any dependencies or assumptions made about the operating environment of the system. E.g. OS, programming language compilers and virtual machine, DB's, network configuration, etc.
- Describe the architecture of the system, that is the most abstract view of how your system is divided into components and how those components are interconnected. The architecture should be described with a diagram showing components and how they are related (or equivalent in words). Beware of designs based on large number of components, they may signal a design that is overly complex.
- The system decomposition should relate the system architecture to the detailed design, to identify the role of each component in the higher-level architectural view. Description of strategy for dealing with errors and exceptional cases (e.g. invalid user input, network or external system failure) that might arise in the use of the software. For anticipated errors and exceptions, a summary of how the software will respond in these situations.

Following tools may be useful in generating your software architecture diagram:

1. Draw.io/mxGraph - http://www.draw.io
2. LovelyCharts - http://my.lovelycharts.com/
3. Cacoo - http://cacoo.com
4. Draw Anywhere - http://www.drawanywhere.com/
5. Creately - http://creately.com
6. Diagrammr - http://www.diagrammr.com/
7. Grapholite - http://grapholite.com/
8. Gliffy - http://www.gliffy.com
9. LucidChart - http://www.lucidchart.com

Interview with the TA

Please make sure to stay within the scope identified during planning meeting. You should be able to demo your software end of sprint 1 during your tutorial.
During the demo, all the team must be present. The TA will mark the attendance for everyone on the team and ask you to show (no more than 3-5 min!) your working software. The TA should be able to use the software to the extent of the feature(s) you have already installed.
Once the other teams present to the TA, you should conduct your Sprint 1 retrospective meeting. Appoint a scribe and document your observations about sprint 1 in a document named `SR1.md` in `doc/sprint1`. You should record:

- the participants in the meeting
- unfinished tasks and group them into stories; add them to `SR1.md` in the form of new user stories. Update your `PB.md` and save the updated copy in `doc/sprint2` (yes, sprint2 - prepare for next sprint!)
- what are practices that you should continue during next sprint
- what are some new practices that you might want to use during next sprint
- what are (if any) harmful practices you should stop using during next sprint
- what was your best/worst experience during sprint 1

Mentor TA

The mentor TA for your project may different from your scrum demo TA. The mentor TA will give you advice on user story, task breaking, project framework, and etc.

The time for talking to mentor TA is totally flexible, you can directly book a time with the mentor TA by the booking system (Coming up later). The voice/text conversation with the mentor TA should happen in the **official CSC301H5 discord server**.

We encourage you to talk to your project mentor TA for a better stable release of your final project.

You will get the Mentor TA assigned later.

Marking (8% of your final mark)

See marking scheme, also check out carefully the Peer Evaluation posted on the project page.