IFB299 Release One Marking Criteria and Submission Guide

	Performance Standards				
	22 - 19	**18 - 15**	**14 - 11**	**10 - 7**	**< 7**
Development	Design, implement, verify and deliver a sophisticated computing system which meets evolving stakeholder needs.				
Weighting 75%	▪ Clear evidence that the team has very closely adhered to the principles of Scrum.	▪ Clear evidence that the team has closely adhered to the principles of Scrum.	▪ Evidence that the team has adhered to most of the principles of Scrum.	▪ Evidence that the team has generally adhered to most of the principles of Scrum.	▪ Little evidence that the team has adhered to the principles of Scrum.
	▪ Very good balanced use of team members' skills while maintaining shared knowledge of system.	▪ Good balanced use of team members' skills while maintaining shared knowledge of system.	▪ Reasonably balanced use of team members' skills while maintaining shared knowledge of system.	▪ Reasonably balanced use of team members' skills while maintaining some shared knowledge of system.	▪ Over reliance on team members' specialist skills thus not maintaining shared knowledge of system.
	▪ Team has a clear, consistent, shared understanding of the overall design.	▪ Team has a consistent, shared understanding of the overall design.	▪ Team has a fairly consistent, shared understanding of the overall design.	▪ Team has a reasonable shared understanding of the overall design.	▪ Team has little shared understanding of the overall design.
	▪ System architecture conforms to industry standards, is maintainable & extensible and can clearly be seen in code.	▪ System architecture conforms to industry standards, is maintainable & extensible and can fairly easily be seen in code.	▪ System architecture conforms to industry standards, is maintainable & extensible and can be determined from code.	▪ System architecture conforms to industry standards, is a reasonable choice and can be determined from code.	▪ System architecture does not conform to industry standards, is not a reasonable choice or cannot easily be seen in code.
	▪ High quality design exhibiting low coupling and high cohesion across the entire system.	▪ Good quality design exhibiting low coupling and high cohesion across most of the system.	▪ Fairly good quality design exhibiting low coupling and moderate cohesion across most of the system.	▪ Reasonable quality design exhibiting reasonably low coupling and moderate cohesion across most of the system.	▪ Poor quality design exhibiting too much coupling or too little cohesion across most of the system.
	▪ Probing, high quality acceptance testing.	▪ Good quality acceptance testing.	▪ Generally good quality acceptance testing.	▪ Generally good acceptance testing but a few tests are not very suitable.	▪ Some acceptance testing but a number of tests are not very suitable.
	▪ Acceptance tests for all stories and a large majority of their permutations.	▪ Acceptance tests for all stories and a good range of their permutations.	▪ Acceptance tests for all stories and the most important of their permutations.	▪ Acceptance tests for most stories and the most important of their permutations.	▪ Acceptance tests for some stories or very few of their permutations.
	▪ Test automation tools have been used to good advantage (e.g. JUnit, Selenium, Jenkins).	▪ Some test automation tools have been used to good advantage (e.g. JUnit, Selenium).	▪ Some simple test automation has been attempted (e.g. JUnit, Selenium).	▪ Both technical and acceptance testing appears to be well managed.	▪ Either technical or acceptance testing appears to be a bit chaotic.
	▪ Source control has been undertaken and logs show evidence of consistent good practice.	▪ Source control has been undertaken and logs show evidence of mostly good practice.	▪ Source control has been undertaken and logs show evidence of fairly good practice.	▪ Source control has been undertaken but logs show that use is a little sporadic.	▪ Source control has been undertaken but use is inconsistent.
	▪ All code is well structured, well commented and exhibits a consistent standard.	▪ Most code is well structured, well commented and exhibits a consistent standard.	▪ Most code is well structured and well commented.	▪ Most code is fairly well structured and fairly well commented.	▪ Haven't maintained consistent code quality.
	▪ Exemplary and consistent tracking at the task level, including recording time taken on tasks.	▪ Consistent tracking at the task level, including recording time taken on tasks.	▪ Tracking most tasks, including recording time taken on some tasks.	▪ Tracking most tasks, but little is recorded about time taken on tasks.	▪ Some tracking has been undertaken, but its use has been inconsistent.
	▪ Velocity and sprint & release burndown charts are always up-to-date and team has informative comments about them.	▪ Velocity and sprint & release burndown charts are always up-to-date and team can discuss them with little prompting.	▪ Velocity and sprint & release burndown charts are up-to-date by the end of the sprint and team can comment on them.	▪ Velocity and sprint & release burndown charts are up-to-date by the end of the release and team can comment on them.	▪ Velocity and sprint & release burndown charts may not all be up-to-date by the end of the release.

IFB299 Release One Marking Criteria and Submission Guide

	Performance Standards				
	8 - 7	**6 - 5**	**4 - 3**	**2 - 1**	**< 1**
Functionality	Delivered functionality consistent with a development team of this size over two sprints.				
Weighting 25%	▪ Successfully delivered the stories agreed during release planning, subject to appropriate revisions negotiated with the customer. ▪ Focus of the release and the coherence of the sprints have been maintained in any revisions made to the original plan. ▪ Release delivers an engaging, high quality and apparently bug free user experience.	▪ Successfully delivered almost all of the stories agreed during release planning, subject to appropriate revisions negotiated with the customer. ▪ Focus of the release and the coherence of the sprints have largely been maintained in any revisions made to the original plan. ▪ Release delivers a fairly engaging and good user experience with few minor bugs.	▪ Successfully delivered most of the stories agreed during release planning, subject to appropriate revisions negotiated with the customer. ▪ Focus of the release and the coherence of the sprints have largely been maintained; although some compromises have been made from the initial plan. ▪ Release delivers a good user experience with a few bugs.	▪ Successfully delivered most of the stories agreed during release planning, but some revisions were not managed well. ▪ Focus of the release and the coherence of the sprints are visible, but compromised. ▪ Release delivers a usable system with a few bugs.	▪ Successfully delivered some of the stories agreed during release planning, or revisions were not negotiated with customer. ▪ Focus of the release is compromised, with little coherence remaining. ▪ Release delivers a system that is awkward to use or with significant bugs.

Some guidance: This assessment will not revisit the assessment of the stories. We assume that the stories have been submitted and that they were of at least reasonable quality. The functionality criterion is thus not about the user stories, but about the functionality that was actually built.

For customer engagement write a one page of bullet points summarising your interactions with your client – revisions of user stories, agreement on acceptance criteria, customer decisions in prioritisation and release planning, and renegotiating release plans. Don't overplay it – just give your tutor a reasonable summary of your interactions to date. Remember that s/he should have some knowledge of these interactions anyway.

SUBMISSION

- You should provide your tutor with access to your code repository, including ***source code (both application and test), unit test and integration test results, and all documentation for the project. My recommendation here is that you place documentation in a directory called doc, but this can be agreed with your tutor.***
- Create a branch in your repository that contains the current version of all material, including up-to-date release plan and sprint plans, that reflects the state of your project at the end of your first release. (This will allow you to continue development on your project without worrying about any changes you make in sprint 3 impacting on your grade for the first release.)
- At your first release workshop meeting you will present a demo of your system to your tutor and client.
- Ensure that your tutor is able to access and explore your system after the first release workshop. You should arrange between you how best to facilitate this exploration. (e.g. Access to a cloud-based application, access to an external server, providing a copy of a virtual machine, …) Mostly, tutors will be happy not to have to install your system on their own machines.
- Finally, it is our expectation that most systems will be buggy. It is not your job to have the system fail during a demo, so there are no ethical problems in choosing to demonstrate the best implemented stories for this. But there *are* ethical problems in claiming bug free systems when they are not. Not to mention credibility problems.