ENGR 301 - 2021 - Individual Performance Assessment

- **Due Date:** 2359 Friday 28 May 2021
- **Assessment Weight:** Worth 40% of ENGR 301 final grade

This *non-submission* assessment is of how well you have performed at following good project management practices over the second half of the trimester. You will be assessed on how well you, as an individual in the context of a group project, are following the processes and practices described in lectures in both the management and practical aspects of your project. This is an evidence-based assessment which will draw on several sources including, but not limited to, the setup and structure of your project work in GitLab (issues, boards, milestones, epics, roadmaps etc.), profile and event information such as the tile view, the output from `gitinspector` and the observations of the tutor / senior manager of your team's processes and practices in laboratories.

The assessment will be on a 5-trait basis for ENGR301:

Number	Trait	Indicative Weighting
1.	Planning	35%
2.	Construction	20%
3.	Integration, Testing, and Automation	5%
4.	Working Systems	5%
5.	Communication, Collaboration, and Review	35%

These traits are an Engineer's core competencies when working on projects, as described in the GitLab Workflow overview, Introduction to GitLab Flow and the DevOps Lifecycle. Note that it is understood that students can't exhibit all of the traits all of the time, and students shouldn't feel any pressure (or *temptation*) to "tick boxes". What is looked for is the regular expression of these traits over the period of the assessment, in a way that makes positive contribution to your team in the context of your project.

Note: For some projects the individual's mark weighting may not be fair. We reserve the right to adjust weightings to ensure the marks are fair.

1. Planning

Project development is directed toward well-defined outcomes and deliverables both during and at the end of the project; all project management areas have a well-organised and maintained representation in GitLab. Almost all systems, be they hardware or software, are for good reason developed iteratively and everyone is expected to plan and organise their work individually in the context of the group project. For all teams, this will mean not only establishing and executing a plan but also holding iteration-start meetings for the specific purpose of monitoring, control and and revision of the plan. One good way to help everyone to engage in planing is the "Iteration Planning Game", which was introduced in lectures/tutorials. It is expected that individuals who exhibit this trait will be able to explain how their project work contributes to overall

project goals in a way which easily comprehensible by anyone: team, client or senior management.

Evidence we will look for:

- Engagement with, and contribution to planning and organisation of project **scope** and **time** using issues, labels, boards, milestones, epics, roadmaps, etc., including:
    - project work is broken down and organised into epics, stories, and tasks,
    - an appropriate level of written detail in descriptions and comments,
    - start and end/due dates are set for issues, milestones and epics, which fit into a coherent whole,
    - maintaining the project issues, labels, boards, milestones,
    - revising, prioritising and estimating the work plan for coming iterations,
    - referencing what's related by using GitLab's cross-linking and referencing features,
    - issue closing patterns in merge requests to close issues,
- Interaction with the client at client-team meetings to obtain their input and involvement in planning, especially the prioritisation for future iterations and the organisation of the project in time.
- Defining the **quality** metrics for the project and defining the processes for actively monitoring these metrics,
- Active management of:
    - **stakeholders**, especially the project client;
    - **risk**, including taking ownership of project risks in an area and ensuring risk decreases with time;
    - **communication** by maintaining written documentation in the project repositories and notes in the project wiki which are linked to descriptions and comments in issues;
    - **cost** and **procurement**, to ensure the project has what it needs when it needs it.
- Tutor observations of individual performance at planning and client-team meetings.

2. Construction

Good development practice is established and everyone follows it throughout execution. Teams are expected to determine from *best practice* what *good practice* means for their project context, to establish software and/or hardware standards for the project, and to put in place the tools to check them. For individuals this means *at least* aligning their work to follow GitLab Flow so that their work is tracked by the Cycle Analytics. All code/schematics should be merged to `master` under the "four eyes" principle: either pair programmed/bread-boarded, or with explicit reviews on merge requests, or with reviews of hardware quality.

Evidence we will look for:

- Adherence to GitLab Flow as shown by the Cycle Analytics.
- Design and coding standards, via the use of linting (at the most trivial) and code quality tools, can be shown or seen by inspection of the GitLab Project.
- Code refactoring and electronics design are a purposeful activity.
- When pair programming/bread-boarding, committers alternate within pairs when the driver and navigator roles rotate.
- Coding/bread-boarding/soldering/reviewing partners switch often, such that contact is had with all team members and most parts of the technical work.
- Contribution of issue and merge request templates *and* evidence that templates are *both* useful and *maintained*.
- Reviews the work of others and their own work is reviewed by someone other than the author(s).

- Incorporates external performance and/or quality standards, e.g. ISO/IEEE Standards, when appropriate.

3. Integration, Testing and Automation

Technical work is automatically tested and continuously integrated into the project. All engineers working on projects test their systems for common/known/potential errors and for conformance with **scope** (i.e. to catch feature regressions) and against **quality** metrics. This testing should be set up to run automatically and the work, when passed, should be built automatically, using the GitLab Continuous Integration feature as part of GitLab Flow. Reviews of merge requests should take place *after* automated tests have passed (rather than being a substitute for automated — or even manual — testing).

Evidence we will look for (only expecting some evidence of this for ENGR301, frst three activities):

- Work is committed regularly within an iteration (rather than near the end)
- Branches are short-lived on the time scale of an iteration; there are no stale branches nor branches which are behind/ahead of `master` by several 10's of commits or more.
- Testing is performed on commits for all work.
- Contributions are made to test suites and test suites are maintained over the duration of the project.
- Continuous Integration pipelines are set up to perform meaningful tests and are maintained.
- Code Coverage pipelines are set up to perform a meaningful tests.

4. Working Systems

Project work contributes business value to the client on an ongoing basis during the project and prior to final delivery. Maintaining working systems and delivering business value to the client are two very effective ways of ensuring you meet project goals on delivery. To strongly exhibit this trait, SWEN and NWEN students should be able to rebuild the software "from scratch" and demonstrate a working system. ECEN students should be able to demonstrate, without significant effort, the most recent working system. All students should be able to clearly show the relationship between current tasks and business value.

Evidence we will look for (only expecting some evidence of this for ENGR301, first two activities):

- Able to demonstrate a working system at any time;
- Able to explain orally *and* show written evidence of how the task in hand is contributing business value to the client.
- Software or systems are deployed to the client's infrastructure.
- Continuous Deployment pipelines are set up and maintained.

5. Communication, Collaboration, and Review

Project work, experience and expertise is shared within a self-organising, cross-functional team with explicit collective code/system ownership, creating a positive feedback loop for iterations. In a high-functioning team, at the beginning of an iteration, we expect to see individuals self-select tasks, then get on with it. Tasks would be chosen to increase exposure to other parts of system, and to help upskill other team members. At the end of an iteration we expect to see a *sprint* or *iteration review* prior to planning for the next iteration (see Trait 1 *Planning*). These reviews are for the purpose of sharing sharing the tools, discoveries, and lessons learned during the iteration and to provide context for the planning of future iterations (plural!) *and* the project roadmap.

Retrospectives, introduced in lectures, are a valuable larger-scale review of project progress and process at key points during the execution of a project. Teams are expected to conduct retrospectives at least twice during project execution. Some teams, who have key events in their project roadmap such as field trips or other events, will find it natural to hold retrospectives immediately after. Minutes of the retrospective should always be taken and the minutes should be easily comprehensible by anyone, including those outside of the team. Any recommended changes should be **SMART** (**S**pecific, **M**easureable, **A**chieveable, **R**elevant and **T**imebound). Retrospectives may be facilitated by a team member but are preferably facilitated by someone external to the team. Teams are encouraged to make facilitation exchange agreements, where one member of each team facilitates the retrospective of the other. Please note that a Tutor must be present at the retrospective *as an observer* for this aspect of the trait to be counted.

Evidence we will look for:

- Everyone can explain all parts of the project and how they contribute to project goals and deliverables.
- Each team member has committed/contributed to multiple parts of the project, as seen in the various logs (git, Issue, Contribution, etc.).
- Individuals are observed to be team players and work as a whole team.
- Attendance at laboratories consistent and
- Attendance at review/planning meetings is consistent and punctual.
- Engagement with, and contribution to, retrospective(s).
- Sustained positive changes as a result of retrospective(s).
- Sharing tools, discoveries, and lessons outside of the team; helping members of other teams with their technical and non-technical work (e.g. in person or via Mattermost).