Version control

Written by Jamie Knott (project VC engineer)   Date: 28/01/2020

The purpose of this document is to provide an overview of the main project toolset, GitLab, which will be utilized for the entirety of the Yuconz employee details system project. This document provides details for the ways in which the project team will utilize GitLab, and how this will be of benefit overall.

GitLab

One of the fundamental tools that will be utilized for the Yuconz project will be GitLab. The project group has been assigned a suitable GitLab repository for the purposes of the Yuconz project, of which will contain all project related code. All members of the project team have been briefed in the use of this space (during the second group meeting- see meeting minutes for details) and GitLab, in general. GitLab is a useful DevOps lifecycle tool, that contains a great deal of beneficial functionality for use during the project development.

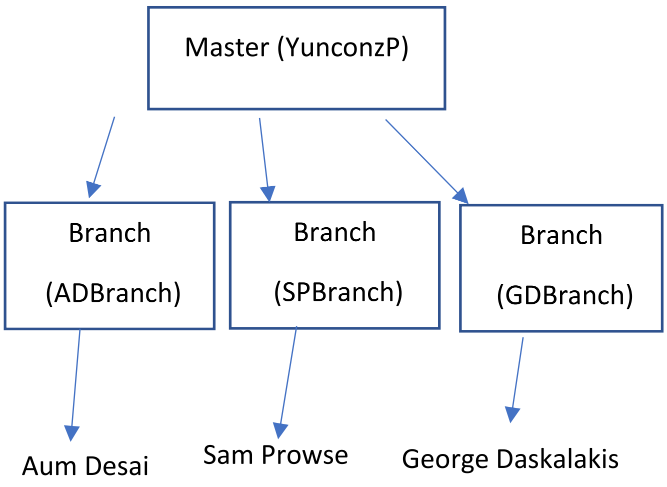
A key feature of GitLab, that the project team intends to utilize, is the ability to create, modify and set **Issues**. This will allow for efficient assignment and allocation of project tasks (likely only coding related tasks), without requiring the need for actual communication and thus no confusion will arise regarding one's assigned responsibilities. Each GitLab Issue will contain details about the task requirements, the project team member assigned to the task, the expected completion date, the actual completion date (once concluded) and also the draft code itself, for further review at a later date (before being committed). This will greatly organize each team member's coding tasks and responsibilities, and so will ensure that everyone can clearly see their own role within the project overall.

A final useful feature of GitLab, is its ability to accommodate **Pull Requests** for a project. This will greatly assist with reviewing, checking and eventually merging each member's code into the final product. This is a much more preferred and convenient means of code collation, as opposed to a team member manually trying to combine everybody's code.

GITLAB PROCEDURE

The following section provides an outline of the project team's policy regarding the use of GitLab.

- **Branches**



The rudimentary diagram to the right, displays an outline of the project team's GitLab branch structure. When producing any pieces of code (for the project), the team member must first push their code (using Git Commit) to their assigned GitLab Branch (which will be empty at the start of the project). Once this has been done, the project team member can then make a **pull request** of the master branch, in order to apply changes to the actual project code (stored within the master). Through using this branch as a subsidiary, code can be reviewed before it is merged (to the master). As shown by the diagram, the master branch has the name "YuconzP", with the sub-branches being of the format (Team Member's Initials + "Branch"). All project team member's will have their own branch (with the exception of team member, Jamie, who isn't assigned coding duties for this stage), in order to store their code, before making a pull request (for merging).

- **Conflict resolution**

In order to resolve potential merge conflicts, the project team has decided to use GitLab's **merge conflict interactive mode**. Through the use of this mode, any merge conflicts will be shown and highlighted to the project team member. The conflicts arisen must then be "marked" (to specify what should be merged), of which will result in an automatic merge of the target and source branches (resolving the conflict).