CPSC 3720 – Assignment 1

## Overview

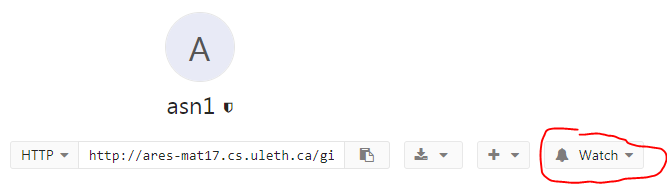
In this assignment, you will:

* Design and implement a text-based card game using Model-View-Controller.
* Keep track of your progress using version control.
* Write passing unit tests for Model classes.
* Determine how well your code is tested using code coverage.
* Use static analysis to detect bugs and avoid dangerous coding practices.
* Generate documentation for your code using doxygen.
* Use continuous integration to automate the running of software engineering tools.

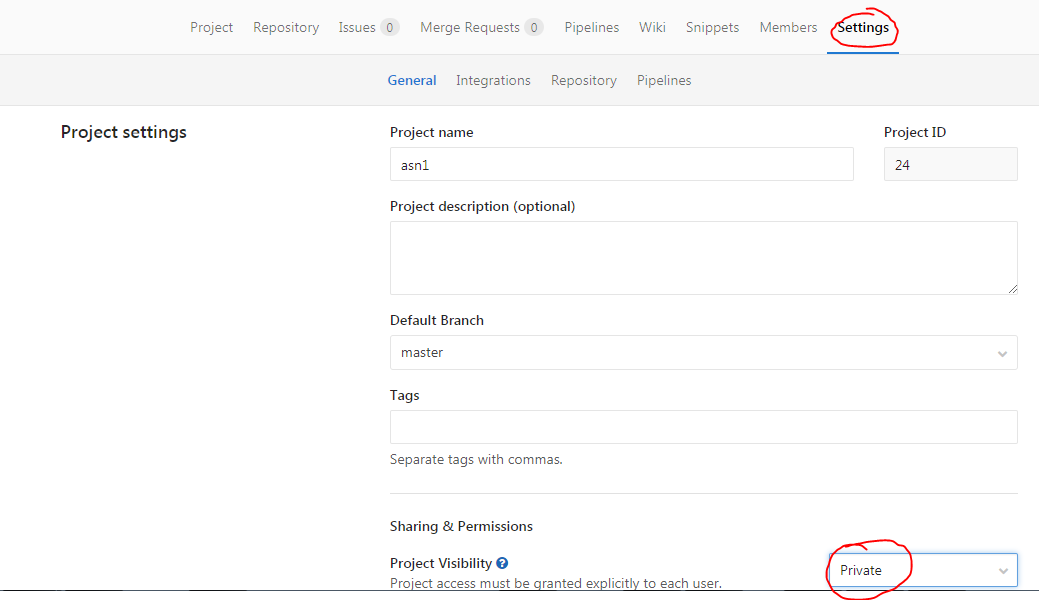
### Instructions

#### Setup

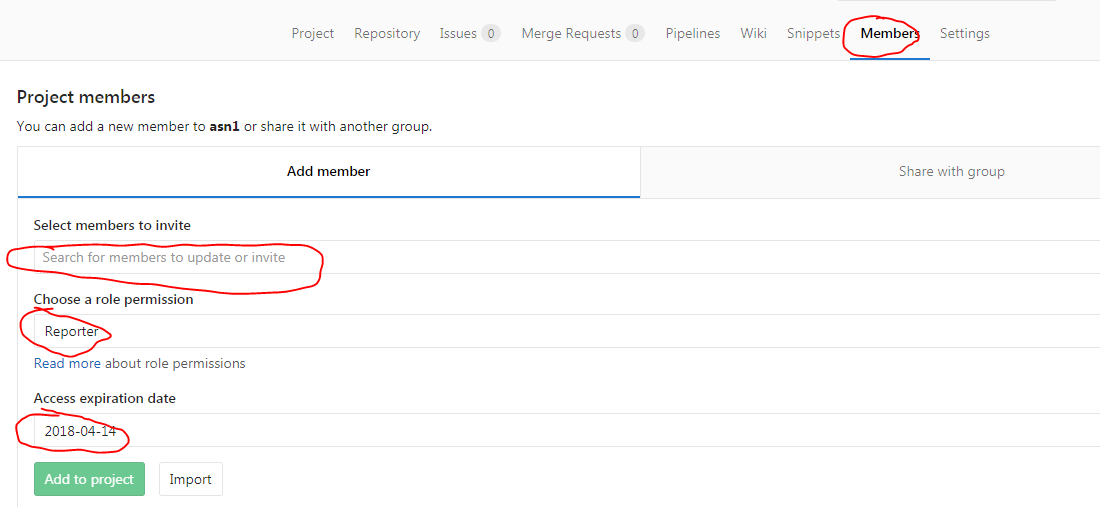
1. Go to the Git repository at <http://ares-mat17.cs.uleth.ca/gitlab/cpsc3720/asn1>. As it is a CS department server, you will only be able to do this on the campus network (or via VPN).
2. Set your notification settings for this repository to “Watch” so you will receive email notification if there are any changes to repository (e.g. clarifications are added to the instructions).



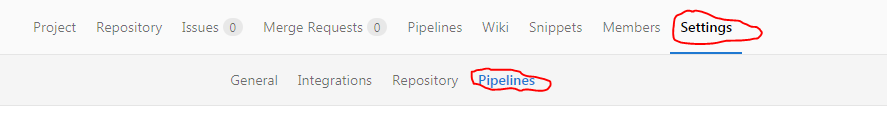
1. Fork the repository so you have your own copy.
2. Set the project visibility for your forked repository to “Private”.



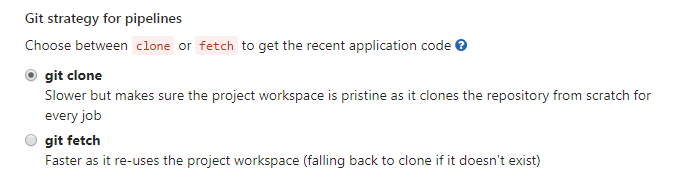
1. Add the marker and instructor (anvik) as a member of your project with the permission “Reporter”. You will be provided with the marker’s CS department user nam on Moodle. This is needed so the marker can grade your assignment.



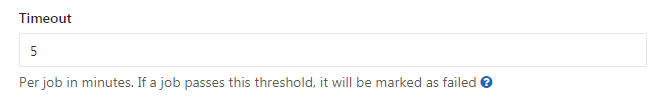
1. Setup your GitLab repository for running continuous integration for your project.



* 1. Set the *Git Strategy* to “git clone”



* 1. Set the Timeout to 5 (i.e. 5 minutes). Your CI job will be small, so this should be lots of time and will prevent any infinite loops from tying up the CI server.



#### Completing the Assignment

1. Create a local clone of your assignment repository.
   1. Create an upstream remote so you can pull in any updates to the assignment, if that happens.  
        
      git remote add upstream <http://ares-mat17.cs.uleth.ca/gitlab/cpsc3720/asn1>
   2. If you need to get updates from the assignment repository, you can pull them into yours.  
        
      git pull upstream master
2. Create a text-based game of GoFish.
   1. If you are not familiar with the game, see <http://www.bicyclecards.com/how-to-play/go-fish/>.
      1. To try and make things simpler for you:
         1. A player only gets one chance to ask for a rank. If the player asked has the card(s) then asking player collects the card(s) and the turn passes to the next player. A player does not get to keep asking for ranks if he “makes a catch”, either from another player or from the drawing from the deck.
         2. The game ends if a player has no more cards in their hand.
   2. The design of your system should clearly use the Model-View-Control pattern.
   3. Create unit tests using Google Test (gtest) that test your Model classes. Your model classes are to have as close to 100% as you can get.
   4. An example Makefile is provided to help you build and test your program, run static analysis, memory leak checking, and code coverage. Edit the Makefile so that it works for your assignment.
      1. The Makefile must have the following targets:
         1. game: Builds the game for system level testing.
         2. test: Builds and runs the unit tests.
         3. memcheck-game: Runs valgrind –memcheck to check for memory leaks when playing the game.
         4. memcheck-test: Runs valgrind –memcheck to check for memory leaks when running the unit tests.
         5. coverage: Runs lcov to generate HTML reports of the unit testing code coverage.
            1. The HTML reports are to be located in the coverage directory.
         6. style: Runs cppcheck to check for bugs and bad programming practices.
         7. docs: Generates HTML documentation using doxygen for your application.
            1. Use doxywizard to create your doxygen configuration file. Make sure any paths are relative, not absolute, otherwise the marker will not be able to generate the files after cloning your repository.
            2. Generate the HTML into the docs directory.

# Grading

Your assignment will be assessed by looking at:

* The status of most recent build in your repository’s GitLab pipeline nearest the deadline (but not after the deadline).
* Playing your game. The prompts for your game should be clear and concise.
* The number of passing unit tests and the code coverage for the unit tests.
* The number and size of the memory leaks. This value should be as close to 0 as possible for both the game and the unit tests.
* The number and type of errors identified by static analysis.
* Examining the generated documentation.

### Submission

Submit the URL of your forked repository to Moodle. The grader will clone your repository as of the deadline for grading.