Version Control

The aim of this document is to ensure that all members are familiar with Git's functionality for version control and how they will be used.

Gitlab:

Because Git specialises in version control, its features will be at the heart of managing our projects. The common way to represent different versions would be to rename files to reflect which version is which, however this unnecessarily consumes space, and Git provides functionality that renders this idea obsolete. The central repository stores the updated file(s), along with previous versions of the file and who modified them. It will also be required for members of our group to write a short comment about why the file changes have been made, which will remove the ambiguity of why different versions have been created in the first place.

Not only will previous versions be retained, they can also be restored. Updating code can often break what has already been implemented, so this functionality will be essential for recovering work that may have been corrupted in the process, or experimenting with alternative changes in another branch.

Branches are offshoots of Git's 'master branch' (where the main project is kept). They allow group members to work on one part of the project without affecting the main code, which is useful for experimentation and implementing fixes. It is required to name of our branches in a way that will describe their purpose so that we will be aware of which features are being worked on.

Git is a DVCS (Distributed Version Control System), which means that the contents of the project will be stored on every member's local repository. All of the work will be backed-up on multiple computers; because of this, we will have high speed access to the files and we will be able to work offline, which improves accessibility for all members. Having back-ups will also contribute to the security of the system.

Small changes will be committed to our local repositories as often as possible to ensure that there is a trail of versions to restore if necessary. Commits to the central repository will only happen if the code can fully compile and pass all of the necessary tests, thus is it expected that commits to the central repository will not be as frequent. In the event that two or more members are editing the same document at the same time, we will use the 'copy modify merge' approach rather than the 'lock modify unlock' solution to give members the freedom to work on documents simultaneously while staying up to date. We will avoid pushing our code/documents before pulling the most recent updates from the central repository, and to ensure that our local repositories are updated, we will commit to our local repositories before pushing to the central one.