## Branches:

master – The main branch. This branch should always remain in a fully functional “release” state. All unit tests must pass before changes are merged into master. This allows us to have a client-facing branch that should only be updated to reflect the completion of project milestones.

dev – The development branch. This branch is used for changes and updates between releases. Outside of exceptional situations, the dev branch should also remain in a functional state. This branch allows for changes to be made to a branch that is similar to the release branch, while creating a staging area for changes to be tested before merging into master.

fix/feature/change – Appropriately named fix/feature/change branches are used for the implementation of new features. These branches allow for the development of new features and/or hotfixes that will not affect the stability of the master branch, and should only be merged into the dev branch after testing.

hotfix – A branch based on master for the purpose of fixing a bug found after release. Hotfixes should only be created if a bug is found in the master branch after release and the dev branch has been changed since that release. This allows the developers to fix bugs quickly in master without reverting dev to the release state.

## Branch Creation:

Issues must be created before creating a branch, and every branch should address a single issue, where practical. This prevents duplicate work on the same issue and allows for issue management.

## Naming Convention:

The name of a branch should be succinct, but appropriately descriptive. Be cognizant of other currently existing branches and avoid using a similar name. Branches will be named with a prefix to categorize what kind of issue it aims to address. For instance, “fix-branchName”, or “feat-branchName”, or “doc-branchName”, where “fix” is for fixing an issue, “feat” is a new feature, and “doc” is for documentation.

As the project progresses, the need for new prefixes may be identified. Prefixes will be defined in the GitHub Wiki. Using the prefixes along with the name will help with getting a modest understanding of what the branch aims to address.

## Merging:

To merge changes or features into dev and hotfix, it must be reviewed and tested. Merging can occur after three individuals have provided a review of the pull request. The developer who created the pull request should not provide a review but is expected to participate in discussion and resolution of issues found during the review.

To merge changes into master, similar testing and reviews must occur as described for merges into dev. Outside of extraordinary circumstances, only dev may be merged into master.

Procedure for merges into master, dev, or hotfix branches depends on the nature of the change. Some changes may not be inherently testable (updates to comments, documentation, etc.). Review is required for all changes and changes to the code require test results.

After a change/feature branch is merged into dev, remove that branch.