**GitHub Workflow**

1. **Summary**

Branching is an important aspect of development which allows divergence from the main line of development so work can continue.

1. **Commits**

* Ensure on first commit a .gitignore file is included.
* Comment each commit with a #[Task Number] - where [Task Number] is the ID of a relevant JIRA Task

**e.g.**

#CI-31 Fixed spelling error on details page and updated button style.

* Stage only the files relevant to one fix at a time. i.e. create two commits each with their own Task references

**e.g.**

**Bad:**

Commit1 - Fixed #CI-31, #CI-32

**Good:**

Commit1 - #CI-31 fixed

Commit2 - #CI-32 fixed

* Commit message should be descriptive

1. **Naming**

* Follow branch-per-feature and branch-per-release principles.
* Delete feature and bug branches once they have been merged into **master**
* Pull changes from **master** into feature branches often. **Reason**: Syncing changes from **master** often during development ensures that merge conflicts are kept to a minimum.
* Features and change controls should be prefixed with "feat/"
* bugs should be prefixed with "bug/"
* Branches that will contain both features and bugs in it should be prefixed with "feat/"

1. **Guidelines**

* Do Send pull requests to a fellow developer before merging feature / bug branches into **master**. This promotes a strong code-review culture. Note that branches should be pushed to the remote before initiating a pull request.
* CONSIDER Updating master directly with database changes and push the changes early. This allows other developers the opportunity to pull the database changes and continue working with their features.
* DO Stash changes before switching branches if work done is not in a stable state. e.g. when switching branches to make changes to the database. You can retain multiple stashes but bear in mind that stashes are only available on your local machine and cannot be pushed to the remote repository.

1. **Branch Structure**

* From Master create branch for particular release
* Create Feature or Bug branch from that particular release
* Merge Feature branch with Release branch once feature is implemented and tested and seems to be stable
* Merge Release branch with master once release stable
* Delete feature branches

Master

Release1

feat/login

feat/registration