**Branching and Merging Strategy**

* The master branch will hold the current code in UAT for all clients.
* It is assumed that the code changes for a sprint will be sent for UAT at the end of the sprint.
* At the start of every sprint, a release branch is created from the master branch. The release branch name will be in the format
  + <Project Abbreviation>-Release-<version in the format x.x.x>
* A new branch should be created for each story from the release branch. Story branch name should be the story number from Jira with an optional description after a hyphen. For example - LMS-1100 or LMS-1100-New Bonus Game
* An update of the release branch has to be taken before a new story branch is created.
* All code pertaining to a story should be pushed to the relevant story branch only. This can be in the form of multiple commits as well, so long as the commits don’t cause a build failure.
* Once the development is complete, developers will generate a pull request for code review and merging of the story branch to the release branch.
* While creating Merge request developer should follow below steps
  + Ensure that there are no merge conflicts. The developer can first merge the updated release branch to the story branch to ensure this.
  + Enter the title of the merge request, that can be the story name with story title.
* Developer can add WIPtag before the title of the request, to avoid the merge into release branch after review. WIP means work in progress and the developer does not want his branch to be merged in the following merge request but only reviewed.
  + In comments for the merge request, developer can add FIXES-(story number) to close his/her JIRA story automatically after the merge. For example - FIXES-LMS-1100
  + Developers should select the checkbox saying “Remove source branch” while submitting the request. This will ensure that the story branch is deleted automatically after the merge.
* At the end of every sprint, as the code is sent for UAT, the current release branch will be merged to the master branch and a new release branch should be created with the version incremented.
* If a UAT release is being planned in the **middle** of the sprint cycle, the release branch has to be tagged.
  + The tag name should be in the format
    - R<Release version>-<Client>-<Client>
  + The tag should be an annotated tag, created with the following command format
    - git tag -a <tag-name> -m "tag comments listing relevant features”
* For production fixes and other patches which are not planned for a sprint cycle but are still released to UAT or production, the following hot-fix strategy should be followed
  + A hot-fix branch should be created with name in the format
    - Hot-Fix-<Jira Story Number>
  + The hot-fix branch will be merged with master once the development is complete and the code is tested.
  + The hot-fix branch will be merged with the current release branch immediately as well.

