**Branching Strategy**

* Use a single branch for all development.
* Create feature branches from the main branch.
* Merge your feature branch back into the main branch when you’re ready to deploy.
* Delete your feature branch after it’s merged.
* Keep your commits small and focused.
* Write descriptive commit messages.
* Make sure your code works before committing.
* Don’t merge broken code.
* Avoid unnecessary merges.
* Rebase instead of merging.

**Master:(For Production Server)**

* The Master branch stores the official release.
* Tag all commits in the master branch with a version number.
* Production build can happen only from the Master branch.
* This branch will be restricted and accepts only MR.

**Develop:(For UAT Server)**

* The develop branch serves as an integration branch for features.
* Test env build will happen from this branch.
* After code review, the MR option will be enabled.
* This branch will be restricted and accepts only MR.

**Feature: (Created by dev team)**

* Feature branches are generally created off to the latest develop branch.
* Feature branches use develop as their parent branch.
* The feature branch should never interact directly with the master.
* When you’re done with the development work on the feature, the next step is to merge the feature branch into develop and it will be deleted.
* Development env build will happen from this branch.

**Best Practices:**

* All the branches need to be restricted except feature branch.
* Code can be merged only using Merge Request (MR).
* All code will be reviewed before merging.
* Always maintain only Master, Develop, and current sprint branches.