IP01: Configuration Management System

In this project, test cases were designed to demonstrate how the following situations are handled in the Configuration Management System, GitHub.

1. Code checkout
2. Code check-in
3. Code snapshot (or release point)
4. Code in development by 2 users, first one checks out code, second checks out code, makes changes and checks in, first makes changes and needs to check in
5. Code changes need to be rolled back to a previous level
6. One developer is working on a new version, one is fixing bugs in a previous release

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Functional Area | Test Scenario | Steps | Expected Results | Actual Results | Pass/Fail |
| 1 a | Code checkout | Verify that user is able to check out the code from CMS to their local machine | 1. Login to the GitHub and copy the Repository URL 2. Use the git clone command to check out the code to local machine   git clone <name url> | 1. Login to GitHub is successful 2. User is able to clone the repo | Same as expected | Pass |
| 1 b | Code checkout | Verify that user is able to check out a specific branch from CMS | 1. Login to the GitHub and copy the Repository URL 2. Use the git clone command to check out the code to local machine   git checkout -b test <name of remote>/test | 1. Login to GitHub is successful 2. User is able to clone the repo | Same as expected | Pass |
| 2 a | Code check-in | Verify that user is able to check in the code with a new branch to the CMS system | 1. Login to the GitHub and copy the Repository URL 2. Use the git clone command to check out the code to local machine   git clone <url>   1. Now checkout a branch from the main using command – git checkout -b <new\_branch> 2. Now make changes to the newly created branch and commit the code using command – git commit -m “message” 3. Now check-in the code by pushing the new branch to the CMS | 1. User is able to clone an existing branch from the GITHUB repo to their development system. 2. User can publish his/her working branch to GitHub | Same as expected | Pass |
| 2 b | Code check-in | Verify that user is able to check in the code to an existing branch in CMS system | 1. Login to the GitHub and copy the Repository URL 2. Use the git clone command to check out the code to local machine   git clone <url>   1. Now checkout a branch from the main using command – git checkout -b <existing\_branch> 2. Now make changes to the existing branch and commit the code using command – git commit -m “message” 3. Now check-in the code by pushing the existing branch to the CMS | 1. User is able to clone a existing branch from the GITHUB repo to his/her development system. 2. User is able to publish his/her working branch to GitHub | Same as expected | Pass |
| 3 a | Code snapshot (or release point) | Verify user is able to create release from a given branch | 1. Clone the GitHub repo into the local development environment 2. Use the following command to create a release tag – git tag -a <tag> 3. Now push the release to GitHub issuing the following command - git push --follow-tags origin <release\_version> | 1. User is able to crate tag from a branch 2. Once code to Pushed to the GitHub, User can view the release 3. All other GitHub user can view and pull the release from GitHub | Same as expected | Pass |
| 3 b | Code snapshot (or release point) | Verify that user is able to edit and update release | 1. Follow the Steps in 3a to create a new release 2. Now edit the created release 3. Verify all the updated texts are saved in the GitHub | 1. User is able to edit an existing release | Same as expected | Pass |
| 4 a | Code in development by 2 users, first one checks out code, second checks out code, makes changes and checks in, first makes changes and needs to check in | Verify that both users can work and check-in their changes without any issue when there is no conflict | 1. User1 and user2 checks out the main branch code 2. User1 creates new branch and updates his/her work and create the MR to the main branch 3. User2 creates new branch and updates his/her work 4. User1 MR is reviewed is merged into the main branch 5. Now user2 creates the MR to main branch | 1. Since user1 and user2 worked in different code areas, so there is no such code conflict, and both can check-in their code | Same as expected | Pass |
| 4b | Code in development by 2 users, first one checks out code, second checks out code, makes changes and checks in, first makes changes and needs to check in | Verify that both users can work and check in their changes without any issue when there is code conflict | 1. User1 and user2 checks out the main branch code 2. User1 creates new branch and updates his/her work and create the MR to the main branch 3. User2 uses that same new branch as User1 and updates his/her work 4. User1 MR is reviewed is merged into the main branch 5. Now user2 creates the MR to main branch | 1. Since user1 and user2 worked in on the same code area, so there is a code conflict. First the conflict needs to be resolved, before the code can be checked in | Same as expected | Pass |
| 5a | Code changes need to be rolled back to a previous level | Verify that the users can rollback any specific commit changes | 1. Clone the Repo and update changes to the repo and commit the changes and push to the GitHub 2. Now revert the last commit using the following command - git revert --no-edit <commit-sha> | 1. User is able to roll back the specific commit | Same as expected | Pass |
| 5b | Code changes need to be rolled back to a previous level | Verify that the users can rollback multiple commit changes | 1. Clone the Repo and update changes to the repo and commit the changes and push to the GitHub 2. Now revert the multiple commits using the following command - git revert --no-edit <commit-sha> | 1. User is able to roll back all the commits | Same as expected | Pass |
| 6 | One developer is working on a new version, one is fixing bugs in a previous release | Verify that both the users working on different versions can commit the changes without any conflict | 1. Developer A Clone the Repo and update changes to the repo and commit the changes and push to the GitHub 2. Developer B Clone the Repo and update changes to the repo and commit the changes and push to the GitHub on branch B | 1. Developer A commits are only reflected on their working branch 2. Similarly, Developer B commits are reflected in branch B only | Same as expected | Pass |