System Requirements Specification Index

For

Git Tags - Versioning

Version 1.0

IIHT Pvt. Ltd.

fullstack@iiht.com

TABLE OF CONTENTS

Git Tags - Versioning

System Requirements Specification

1 PROJECT ABSTRACT

This document outlines the structure for a **Git Tags - Versioning git assessment** designed to evaluate the candidate's proficiency in using Git commands, integrating these commands within a Java application, and managing the build process with Maven. The assessment involves executing specified Git commands, verifying their correctness through a Java application, and using Maven to build and test the application.

2 ASSESSMENT OBJECTIVES

The objective of this assessment is to test the candidate’s ability to utilize git commands effectively with a project environment.

3 ASSESSMENT TASKS

1. Open the terminal in the parent folder. Ensure that the folder path matches the email ID you used for the assessment.
2. Initialize an empty Git repository.
3. Create a file named README.md with the content # TaskManager.
4. Create a Python file task_manager.py and add the function create_task() with the content as:
```
def create_task():
    return 'Task Created'
```
5. Stage both files (README.md and task_manager.py) and commit them with the message "Initial commit: Added README and basic task manager script"
6. After the initial commit, create a tag v1.0 with the message "v1.0: First stable release of TaskManager".
7. Use git show v1.0 to view the commit details associated with the tag v1.0.
8. Modify task_manager.py by adding a new function list_tasks():
```
def list_tasks():
    return ['Task 1', 'Task 2']
```
9. Stage and commit the changes with the message "Added list_tasks feature".
10. Create a tag v1.1 with the message "v1.1: Added list_tasks feature".
11. Use git show v1.1 to view the commit details associated with the tag v1.1
12. Modify task_manager.py to fix the bug in create_task() by changing the return value from "Task Created" to "Task has been created".
13. Stage and commit the changes with the message "Fixed bug in create_task function".
14. Force tag v1.1 again with the updated message "v1.1: Bug fix in create_task function".
15. Use git show v1.1 to view the updated commit details associated with the re-tagged v1.1.
16. Checkout to the v1.0 tag and check the status of the repository.
17. Append "Version 1.0: Initial release with task creation feature" to the README.md.
18. Stage and commit the changes with the message "Added version info for v1.0".
19. Use git log -1 to view the commit history for the most recent commit.

20. Checkout to the main branch and use git reflog to check the reflog for any branch movements and commits.

4 EXECUTION STEPS TO FOLLOW

1. To open the command terminal, you need to go to the Application menu (Three horizontal lines at left top) -> Terminal -> New Terminal and then change the path to utils folder.
2. Once you perform all tasks, please open another terminal with the root address (path with project name).
3. To run your project use command:
   mvn clean install exec:java -Dexec.mainClass="mainapp.MyApp" -DskipTests=true
4. To test your project, use the command
   mvn test
5. If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use CTRL+Shift+B-command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.
6. These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.
7. You need to use **CTRL+Shift+B** - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.