# System Requirements Specification Index

For

# Git Revert And Undo Changes

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

# TABLE OF CONTENTS

# Revert And Undo Changes

## System Requirements Specification

## 1 PROJECT ABSTRACT

This document outlines the structure for a **Revert And Undo Changes 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. Initialize an empty Git repository inside the history_repo folder.
2. Create a file called history_file.txt with some initial content like "Initial commit message."
3. Add and commit the file to the repository with the comment “First commit.”
4. Make a change to the history_file.txt (e.g., "Second commit message").
5. Stage and commit this change with the comment “Second commit.”
6. Make another change to history_file.txt (e.g., "Third commit message").
7. Stage and commit this change with the comment “Third commit.”
8. Use the git log command to view the commit history and identify the commit hash of the second commit.
9. Use the git rebase -i HEAD~3 command to interactively rebase the last three commits and change the commit message for the second commit.
10. Rebase the last three commits and change the order of the commits (make the third commit the first one).
11. Amend the "First commit" message and use rebase again to apply the changes to the history.
12. Use the git rebase master feature_branch command to rebase the feature_branch onto the master branch.
13. Resolve any conflicts that may occur during the rebase process.
14. Push the rebased feature_branch to the remote repository, using force-push if necessary.
15. Create a new branch called rebase_branch from master.
16. Make a commit in rebase_branch with a message like "Rebase branch commit."
17. Rebase rebase_branch onto master using the appropriate Git command.
18. Push both the rebase_branch and master to the remote repository.

- Initialize an empty Git repository inside the revert_repo folder.
- Create a file called revert_file.txt and add some initial content, such as "Initial content."
- Add and commit the revert_file.txt file with the comment “Initial commit.”
- Make a change to revert_file.txt (e.g., "Added some content in commit 2").

 Stage and commit this change with the comment “Second commit.”
 Make another change to revert_file.txt (e.g., "Another change in commit 3").
 Stage and commit this change with the comment “Third commit.”
 Use the git log command to view the commit history and identify the commit hash for the second commit.
 Revert the changes made in the second commit using the git revert <commit_hash_of_second_commit> command.
 Resolve any conflicts that may occur during the revert process (if applicable).
 Commit the revert changes (if Git didn’t auto-generate the message, add it manually).
 Verify the contents of revert_file.txt to confirm the second commit's changes were successfully reverted.
 Make a new change to revert_file.txt (e.g., "Final content after revert").
 Stage and commit the change with the message “Final commit after revert.”
 Use git log to confirm the commit history, ensuring the revert commit and the final commit are both present.
 Push the changes to a remote repository (e.g., GitHub).
 Create a new branch called feature_branch from the master branch.
 Make changes in feature_branch (e.g., add "Changes in feature branch").
 Stage and commit the changes in feature_branch with the message “Changes in feature branch.”
 Revert the changes made in the feature_branch and commit the revert.
 Push both the feature_branch and master branches to the remote repository.

# 4 EXECUTION STEPS TO FOLLOW

1. All git commands must be executed only in the utils folder.
2. 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.
3. Once you perform all tasks, please open another terminal with the root address (path with project name).
4. To run your project use command:
   **mvn clean install exec:java -Dexec.mainClass="mainapp.MyApp" -DskipTests=true**
5. To test your project, use the command
   **mvn test**
6. 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.
7. 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.
8. 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.