GIT CASE-STUDY ASSIGNMENT

Q1.
Basic Git Workflow
Objective: Learn the basics of Git, including initializing a repository, adding files, making commits, and pushing to a remote repository.
Scenario: Create a simple project (e.g., a "Hello World" program in any language). Initialize a Git repository, create a few commits as you make changes to the project, and push the changes to GitHub.

Ans.
Steps performed:

Create remote repository in github
Create local repository in local
Navigate to local repository
git init
git add .
git commit -m ‘add java program’
Made changes to file
git add .
git commit -m ‘changes in program’
git remote add origin git@github.com:rajithachavva/case-study-git.git
git push -u origin master
git url: rajithachavva/case-study-git

Q2.
Branching and Merging
Objective: Understand how to create branches, switch between branches, and merge branches.
Scenario: Create a project with a main branch. Add a new feature on a separate branch and merge it back into the main branch. Resolve any merge conflicts that arise.

Ans.

Used same repo as in first question
git switch -c java
Made changes to java branch
git add .
git commit -m ‘changes’
git switch main
git merge java

Q3.
Collaborative Workflow
Objective: Learn how to collaborate on projects using Git, including cloning repositories, creating pull requests, and code reviews.

Scenario: Simulate a team environment where you clone a repository, create a branch to make some changes, push the changes, and open a pull request. Have someone else review and merge the pull request.

Ans:
git clone git@github.com:rajithachavva/case-3.git
git switch -c rajitha
made some changes
git add .
git commit -m 'modified java file'
git push -u origin rajitha
Created a pull request in github to merge rajitha branch into master
Git url: rajithachavva/case-3

Q4.
Reverting Changes
Objective: Learn how to undo changes in Git using commands like git revert, git reset, and git checkout.
Scenario: Make a few changes to a project and commit them. Then, simulate a situation where a change needs to be undone or reverted to an earlier state.

Ans.
git add .
git commit -m 'hello'
git revert HEAD //this uncommitted and discarded the changes
git reset HEAD~1 // this unstages the changes without deleting them from working directory
git reset –soft HEAD~1 // uncommitted the changes but kept them staged
git reset –hard HEAD~1 //this uncommitted and discarded changes
git checkout – file.txt // discards uncommitted changes in a file

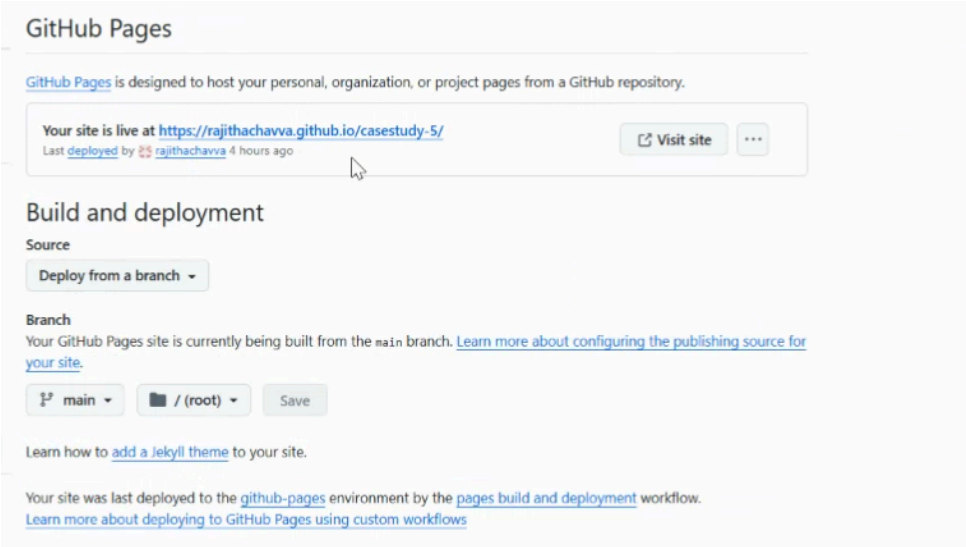
Q5.
Using Git with GitHub Pages
Objective: Practise deploying a simple website using GitHub Pages.
Scenario: Create a static website (e.g., an HTML page) and host it on GitHub Pages. Learn how to push changes and see them reflected on the live website.

Ans:
Created a repo in local
git init
Created a html file
git add .
git commit -m 'iud'
git remote add origin git@github.com:rajithachavva/casestudy-5.git
git push -u origin master

Then navigated to settings in github and then to pages
Selected branch as main
Navigted to url in screenshot and website is hosted there
https://rajithachavva.github.io/casestudy-5/

Made some changes to html file and pushed
git add .
git commit -m “changes”
git push -u origin main
Changes are reflected in url

Git url: https://github.com/rajithachavva/casestudy-5



Q6.
Working with Git Tags
Objective: Understand how to use tags in Git for marking specific points in history, such as releases.
Scenario: Simulate a software release process by tagging specific commits as different versions (e.g., v1.0, v1.1). Practice pushing these tags to a remote repository.

Ans:

git tag -a v1.0 -m ‘this is version 1’
git add .
git commit -m ‘python’
git push -u origin main
Made some more changes
git add .
git commit -m ‘modified java file’
git push -u origin main
git tag -a v1.1 -m ‘version v1.1’
git push –tags

Q7.
Stashing Changes
Objective: Learn how to temporarily save changes using git stash and apply them later.
Scenario: Simulate a situation where you need to switch branches but have uncommitted changes. Use git stash to save your work, switch branches, and then apply the changes back

Ans.

git switch rajitha
Made some changes
git stash
git switch main
git stash apply

Q8.
Exploring Git History
Objective: Learn to explore the commit history using commands like git log and git diff.
Scenario: Practice navigating the commit history to understand how changes evolved over time. Use git log to view commit details and git diff to see differences between commits.

Ans.

git log
git log –oneline
git diff HEAD~1 HEAD

Q9
Git Hooks

Objective: Learn how to automate tasks using Git hooks.

Scenario: Set up a pre-commit hook to check for code formatting or linting before allowing a commit. This can help enforce coding standards.

Ans:

I used checkstyle to check the code
Written below files

Checkstyle.xml, code.java and pre-commit file

Pre-commit file as below to validate code before commit:

After adding above files performed git add . and git commit -m “precommit” and no errors were found with code and also edited java code by adding extra semicolon and then performed git commit and found errors as expected so pre-commit hook is working as expected.

Git url: https://github.com/rajithachavva/precommit-casestudy

Q10.
Git Rebase
Objective: Understand how to use git rebase to integrate changes from one branch into another.
Scenario: Create a feature branch from the main branch, make some commits, then rebase the feature branch onto the main branch to bring it up to date.

Ans.
git switch -c feature
Made some changes
git add .
git commit -m ‘changes’
git switch main
git rebase feature