**Assignment on 01-08-2024**

**Assignment 1:** Initialize a new Git repository in a directory of your choice. Add a simple text file to the repository and make the first commit.

Here's a step-by-step guide to initialize a new Git repository, add a simple text file, and make the first commit. Since I can't execute commands or manage files directly, I'll provide the commands you would use in a terminal or command prompt.

**1. Initialize a New Git Repository**

Navigate to the directory where you want to create the new repository, or create a new directory for it.

# Navigate to the directory of your choice

cd /path/to/your/directory

# Or create a new directory and navigate into it

mkdir my-new-repo

cd my-new-repo

# Initialize a new Git repository

git init

**2. Create a Simple Text File**

Create a text file using a text editor or directly from the command line.

# Create a new text file

echo "This is a simple text file." > file.txt

3. **Add the File to the Repository**

# Add the new file to the staging area

git add file.txt

4. **Make the First Commit**

# Commit the file to the repository

git commit -m "Initial commit with a simple text file."

**5. Check the Status**

You can check the status of your repository to ensure everything is correctly staged and committed.

# Check the status of the repository

git status

**Assignment 2:** Branch Creation and Switching Create a new branch named 'feature' and switch to it. Make changes in the 'feature' branch and commit them.

Here's a step-by-step guide to creating a new branch named feature, switching to it, making changes, and committing those changes.

**1. Create a New Branch Named 'Feature'**

To create a new branch, use the git branch command, followed by the branch name.

# Create a new branch named 'feature'

git branch feature

**2. Switch to the 'Feature' Branch**

To switch to the newly created branch, use the git checkout or git switch command.

# Switch to the 'feature' branch

git checkout feature

**3. Make Changes in the 'Feature' Branch**

Now that you're on the feature branch, make some changes. For example, let's add a new line to the file.txt file.

# Append a new line to the file.txt

echo "This is a new line added in the feature branch." >> file.txt

**4. Commit the Changes**

Add the modified file to the staging area and commit the changes.

# Add the modified file to the staging area

git add file.txt

# Commit the changes

git commit -m "Added a new line in the feature branch."

**5. Verify the Branch and Commit**

You can verify that you're on the correct branch and that the commit was successful.

# Check the current branch

git branch

# Check the commit history

git log –oneline

**Assignment 3:** Feature Branches and Hotfixes \_x000D\_# Confidential - RP **Assignment 3:** Feature Branches and Hotfixes Create a 'hotfix' branch to fix an issue in the main code. Merge the 'hotfix' branch into 'main' ensuring that the issue is resolved has context menu

To address the scenario where you need to create a hotfix branch, fix an issue in the main code, and then merge the hotfix branch back into the main branch, follow these steps:

**1. Create a 'Hotfix' Branch**

Start by creating a new branch named hotfix from the main branch.

# Ensure you are on the main branch

git checkout main

# Create and switch to the 'hotfix' branch

git checkout -b hotfix

**2. Fix the Issue in the 'Hotfix' Branch**

Make the necessary changes to resolve the issue. For example, let's assume the issue is in a file named main\_code.txt.

# Edit the file to fix the issue

echo "Fixed the issue in the hotfix branch." >> main\_code.txt

# Add and commit the changes

git add main\_code.txt

git commit -m "Fixed issue in the main\_code.txt in the hotfix branch."

3. **Switch Back to the 'Main' Branch**

# Switch back to the main branch

git checkout main

**4. Merge the 'Hotfix' Branch into 'Main'**

Now, merge the changes from the hotfix branch into the main branch.

# Merge the 'hotfix' branch into 'main'

git merge hotfix

**5. Resolve Any Merge Conflicts (if necessary)**

If there are any conflicts during the merge, Git will notify you. You’ll need to manually resolve the conflicts and then complete the merge.

# After resolving conflicts

git add conflicted\_file.txt

git commit -m "Resolved merge conflicts and completed hotfix."

**6. Delete the 'Hotfix' Branch (Optional)**

After the hotfix has been successfully merged, you can delete the hotfix branch if you no longer need it.

# Delete the hotfix branch

git branch -d hotfix

**7. Verify the Merge**

Check the log to ensure the hotfix commit is part of the main branch history.

# View the commit history

git log --oneline