**3. GIT Basics**

1. Create a repository “training” and share with your trainer
2. Get a copy of this repository to your local machine

**git clone url**

**git clone https://github.com/Anuka-R98/ITPM\_PROJECT.git**

1. Create a new branch ‘oop-concepts’

**cd <repository-name> Change to the repository directory**

**cd training**

**git checkout -b <branch-name> Create a new branch**

**git checkout -b oop-concepts**

1. Create a directory ‘oop-exercise’

**mkdir name**

**mkdir oop-exercise**

1. Add your oop exercise answer to directory ‘oop-exercise’ and push

**git add .**

**git commit -m "<commit-message>"**

**git commit -m “first commit”**

**git push -u origin <branch-name>**

**git push -u origin oop-concepts**

1. Create a pull request from your branch to master and request approval from trainer
2. After pull request is accepted, delete branch ‘oop-exercise’ from both local and remote

**git branch -d oop-concepts**

**git push origin --delete oop-concepts**

1. List down git command you used in the exercise
2. Describe about following commands and practice

|  |  |
| --- | --- |
| git fetch | git fetch is a primary command used to download contents from a remote repository |
| git branch | **command used to create, list, rename, and delete branches** |
| git pull | **command is used to fetch and download content from a remote repository and immediately update the local repository to match that content** |
| git log | **command is used to view the history of committed changes within a Git repository** |
| git status | **command displays the state of the working directory and the staging area** |
| git diff | **command helps see, compare, and understand changes in your project.** |
| git stash | **command takes your uncommitted changes (both staged and upstaged), saves them away for later use, and then reverts them from your working copy** |
| git stash pop | **command helps to remove or throw away the latest or the topmost stash** |

1. What is a merge conflict?

A merge conflict occurs in Git when two branches have made changes to the same line of the same file, and Git is unable to automatically resolve the changes.

This requires manual intervention to choose which changes to keep or merge, and how to resolve the conflict. It happens when merging two branches and Git cannot determine which change should take precedence over the other.

1. Create a new branch “my-branch” from master
2. Checkout to master branch

**git checkout master**

1. create a new text file “example.txt”
2. Add text “this is from master” and push to master branch
3. Checkout to “my-branch”
4. create a new text file “example.txt”
5. Add text “this is from my-branch” and push to my-branch
6. Merge “my-branch” with master
7. Mention the steps followed to resolve the merge conflict

* Identify conflicting files
* Open conflicting file and view changes
* Determine which changes to keep and which to discard
* Edit file, remove conflict markers, and make necessary changes
* Commit the resolved file
* Push the resolved changes (if working in a remote repository).

1. What is the .gitignore file?

The .gitignore file is a configuration file in a Git repository that specifies files and directories that Git should ignore and not track. These can include compiled files, build artifacts, dependencies, log files, and other files that are specific to the development environment and not needed for the project's source code.

1. Rename the answer sheet as “git-basics-<your-name>-<date>” and submit