**Digital Nurture 4.0 Deep Skilling - Java FSE**

**Week 8 – GIT Hands\_On**

**1. Git-HOL**

**Step 1:** Setup your machine with Git Configuration

To create a new repository, signup with GitLab and register your credentials

Login to GitLab and create a “GitDemo” project

1. To check if Git client is installed properly: Open Git bash shell and execute

If output shows Git with its version information that indicates, that Git Client installs properly.

2. To configure user level configuration of user ID and email ID execute

3. To check if the configuration is properly set, execute the following command.

**CODE:**

$ git --version

git version 2.45.0

$ git config --global user.name "harepreya"

$ git config --global user.email "harepreya14@example.com"

$ git config --global --list

user.name=harepreya

user.email=harepreya14@example.com

**Step 2:** Integrate notepad++.exe to Git and make it a default editor

1. To check, if notepad++.exe execute from Git bash

If Git bash could not able to recognize notepad++ command that implies notepad++.exe is note added to the environment path variable.

To add path of notepad++.exe to environment variable, go to control panel -> System -> Advanced System settings. Go to Advanced tab -> Environment variables -> Add path of notepad++.exe to the path user variable by clicking on “Edit”.

**CODE:**

# 1. Check if notepad++ is recognized by Git Bash

notepad++

# If the above command is not found, add the path of notepad++.exe to your Environment Variables manually:

# Example path: C:\Program Files\Notepad++\notepad++.exe

# 2. Once path is set, configure Notepad++ as the default Git editor

git config --global core.editor "'notepad++' -multiInst -notabbar -nosession -noPlugin"

# 3. Verify the editor setting

git config --global –list

**2. Git-HOL**

Create a “.log” file and a log folder in the working directory of Git. Update the .gitignore file in such a way that on committing, these files (.log extensions and log folders) are ignored.

Verify if the git status reflects the same about working directory, local repository and git repository

**CODE:**

mkdir log

touch log/sample.log

touch debug.log

nano .gitignore

\*.log

log/

git status

git add .gitignore

git commit -m "Added .gitignore to ignore log files and log folder"

**3. Git-HOL**

**Branching:**

1. Create a new branch “GitNewBranch”.

2. List all the local and remote branches available in the current trunk. Observe the “\*” mark which denote the current pointing branch.

3. Switch to the newly created branch. Add some files to it with some contents.

4. Commit the changes to the branch.

5. Check the status with “git status” command.

**CODE:**

# 1. Create a new branch “GitNewBranch”

git branch GitNewBranch

# 2. List all local and remote branches

git branch -a

# (The current branch will have a \* in front of it)

# 3. Switch to the newly created branch

git checkout GitNewBranch

# 4. Add some files to it

echo "This is a test file for GitNewBranch" > branch\_file.txt

# 5. Stage and commit the file

git add branch\_file.txt

git commit -m "Added branch\_file.txt in GitNewBranch"

# 6. Check the status

git status

**Merging:**

1. Switch to the master

2. List out all the differences between trunk and branch. These provide the differences in command line interface.

3. List out all the visual differences between master and branch using P4Merge tool.

4. Merge the source branch to the trunk.

5. Observe the logging after merging using “git log –oneline –graph –decorate”

6. Delete the branch after merging with the trunk and observe the git status

**CODE:**

# 1. Switch to master branch

git checkout master

# 2. Show differences between master and GitNewBranch in CLI

git diff GitNewBranch

# 3. Show differences in P4Merge (assuming it's installed and set as mergetool)

git mergetool

# 4. Merge GitNewBranch into master

git merge GitNewBranch

# 5. Observe the commit history graph after merging

git log --oneline --graph --decorate

# 6. Delete the merged branch

git branch -d GitNewBranch

# 7. Check the status

git status