

Abstract:

This lab report contains the elementary concepts of git and github in enhancing version control and collaborative coding practices and collaboration in software development.

The research also explored its influence on software development and its effectiveness in education. Additionally, the study delved into security practices and identified emerging trends in the software development community. Through this basic concept of git and github, showcasing the improvement in code traceability, collaborative workflows.

This experiment contributes to a better understanding of the challenges associated with version control in collaborative coding projects. The report aims to familiarize the reader with the purpose of using version control system and collaborative platforms for managing code.

Introduction:

Git and GitHub represent integral components of modern software development, revolutionizing version control and collaborative coding practices.

Git, a distributed version control system and GitHub, a web-based platform for hosting Git repositories, have become integral in modern development workflows.

Together, Git and GitHub have become synonymous with best practices in the software development, code management and project efficiency.

Materials:

- ① computer
- ② Git version control system
- ③ GitHub online platform
- ④ Text editor.

Activity :

① Activity 1: Create a git repo

1. first, I have to create a directory which I want as my repository in a location:

\$ mkdir Report1

2. Initialize the directory as a repository:

\$ git init

\$ git config -- global init.defaultBranch main

\$ git branch -m main

3. Have to use config to add my name and email:

\$ git config -- global user.name "Arno".

\$ git config -- global user.email

"Arnobanik1652001@gmail.com"

4. Then create a txt file in my folder

1) Report 1

11) Report 1 (Part 2)

5. Then provide the file cat to print text:

Print "I am Arno".

6. Add this text file to main branch and commit this:

```
$ git add.
```

```
$ git commit -m "A file. Report 1 is added"
```

7. To track this file in GitHub we have to add an url:

```
$ git remote add origin http://github.com/  
Orenibarak/report1.git
```

```
$ git branch -M main
```

```
$ git push -u origin main
```

8. Then adding another text file to the main branch:

```
$ git add.
```

```
$ git commit -m "A file: Report 1 (Part 2) added"
```

```
$ git push -u origin main
```

Activity - 2: Create New Branches
and merge to main

1. Now, I have to create new branch

```
$ git checkout -b branch1
```

```
$ git add.
```

```
$ git commit -m "a file added to Branch 1"
```

```
$ git push -u origin branch1
```


\$ git push -u origin main [switched to main]

\$ git push -u origin New Branch

\$ git checkout main

switched into branch main'

\$ git merge ~~new~~ Branch 1

\$ git push -u origin main

~~\$ git pull~~

~~How, which is created 5 min.~~

18

Git Command List:

1) git config --global user.name (Set the name that will be attached to user commits and tags)

2) git config --global user.email "xyz@gmail.com" (Set the email address that will be attached to user commits and tags.)

3) git status (Displays the status of working directory. Option included new, staged & modified files)

4) git add [file] (Add a file to the staging area)

5) git diff [File] (Shows changes between working directory & staging area)

- 6) `git checkout [File]` (Discard changes in working directory)
- 7) `git commit` (Create a new commit from changes added to the staging area. The commit have a message)
- 8) `git rm [File]` (Removes file from working directory and staging area)
- 9) `git branch` (List all branches in repository)
- 10) `git branch & branch-name` (Create new branch, referencing the current Head)
- 11) `git merge [branch-name]`: Integrates changes from one branch into another.
- 12) `git log` : (list commit history of current branch)
- 13.) `git log ref` (list commits that are represented on the current branch and not merged into ref.)
- 14) `git tag` (list all tags)
- 14) `git tag -a [name]` (Create a tag object named "name" for current commit.)

- 15) `git tag -d [name]` (Remove a tag from local repository)
- 16) `git fetch [remote]` (Fetch changes from the remote, but not update tracking branches)
- 17) `git pull [remote]` (fetch changes from the remote and merge current branch with its upstream)
- 18) `git push -u [remote] [branch]` (Push local branch to remote repository)
- 19) `git stash drop` (discards most recent stashed file)
- 20) `git stash save` (Stores modified tracked files)
- 21) `mkdir project1` (create new folder project1)
- 22) `git init` : (Creates a new git repository)
- 23) `git help` (displays all the necessary info about git commands)
- 24) `git help` (displays all the necessary information about git commands)
- 25) `git rebase` (onebranch and replay them on an alternate branch)

Discussion:

During this lab, when I first started using git & github. I ran into a few common problems. ~~Syntax errors~~ One big challenge was inappropriate syntax ~~usage~~ use. I discovered that failing to ~~enclose to commit~~ ~~commit~~ messages in quotation marks also when specifying branch names ~~because of~~ the space. Also misplaced punctuation, incorrect spacing or missing syntax, spelling mistakes resulted in failed operations. So, I should be careful for ^{these} syntax details for ~~successful~~ successful execution.

Conclusion:

In conclusion, my exploration at git and github has been both challenging and rewarding. Despite initial difficulties, these tools significantly enhance code management and collaboration. git and github are now essential for efficient software development.