Practical No: 01

Practical Title: Demonstrate how to create GitHub account.

Roll No:

Date: **Remark:**

---

1. Open the GitHub Website

- Go to **https://github.com** in your web browser.

2. Click "Sign up"

- On the top-right corner, click **Sign up**.

3. Enter Your Email Address

- Type a valid email that you can access.
- Click **Continue**.

4. Create a Password

- Enter a strong password.
- Click **Continue**.

5. Choose a Username

- Pick a unique username (no spaces).
- Your username will appear in your profile and repository URLs.
- Click **Continue**.

6. Verify Your Account

- GitHub may ask you to solve a CAPTCHA puzzle (verification).
- Complete the verification and click **Continue**.

7. Enter Email Verification Code

- Check your email inbox for a **6-digit GitHub verification code**.
- Enter the code on GitHub to confirm your email.

8. Choose Your Account Type

- Select **Free** (recommended for students and general users).
- Click **Continue**.

9. Personalize Your Experience (optional)

- GitHub may ask a few questions like:
    - How much experience do you have?
    - What will you use GitHub for?

10. Account Creation Complete

- You will be redirected to the GitHub dashboard.
- Your GitHub account is now created and ready to use.

Practical No: 02

Practical Title: Exploring Git Commands through Collaborative Coding • Setting Up Git Repository • Creating and Committing Changes • Branching and Merging.

Roll No:

Date: **Remark:**

---

1. Setting Up Git Repository:

• Initialize a new repository

git init

• Check the status of your files

git status

• Configure your username and email

git config --global user.name "Mitali "
git config --global user.email "mitalishinde2811@gmail.com"

• Clone a remote repository

git clone https://github.com/28Mitali/Mitali-.git

• Add a remote origin

git remote add origin https://github.com/28Mitali/Mitali-.git

2. Creating and Committing Changes

• Add a single file

git add filename.txt

• Add all changed files

git add .

• Commit your changes

git commit -m "Your commit message"

• View commit history

git log

• Push changes to GitHub

git push origin main

3. Branching and Merging

• Create a new branch

git branch feature

• Switch to another branch

git checkout feature

• Create and switch to a new branch

git checkout -b feature

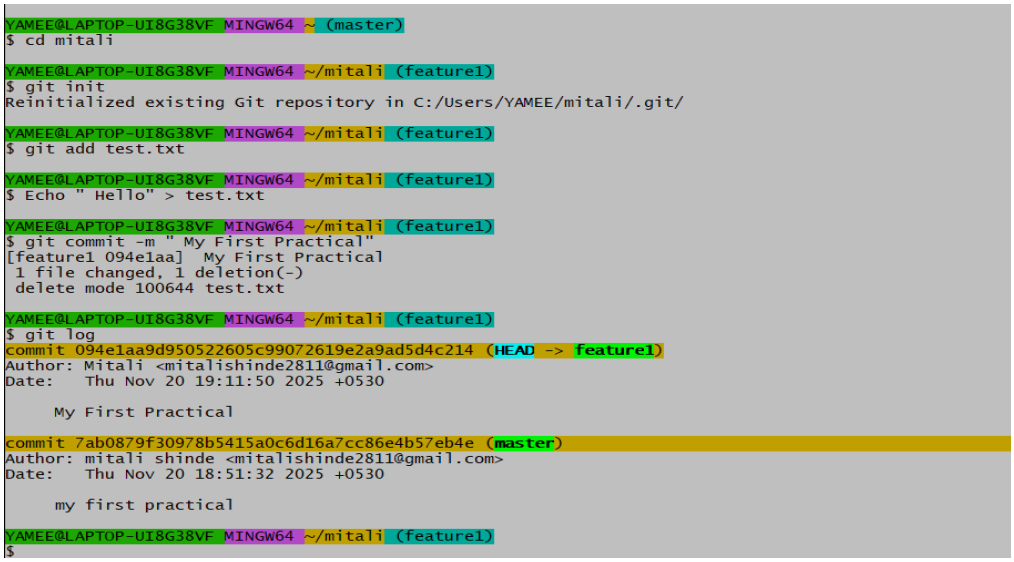
• Merge a branch into main

git checkout main
git merge feature

• Delete a merged branch

git branch -d feature

//OutPut:



```
AMEE@LAPTOP-UI8G38VF MINGW64 ~/mitali (feature1)
 git remote add origin https://github.com/28Mitali/Mitali-.git

AMEE@LAPTOP-UI8G38VF MINGW64 ~/mitali (feature1)
 git push origin master
nfo: please complete authentication in your browser...
numerating objects: 4, done.
ounting objects: 100% (4/4), done.
elta compression using up to 4 threads
ompressing objects: 100% (2/2), done.
riting objects: 100% (4/4), 263 bytes | 131.00 KiB/s, done.
otal 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
emote:
emote: Create a pull request for 'master' on GitHub by visiting:
emote:      https://github.com/28Mitali/Mitali-/pull/new/master
emote:
o https://github.com/28Mitali/Mitali-.git
* [new branch]      master -> master

AMEE@LAPTOP-UI8G38VF MINGW64 ~/mitali (feature1)
 git branch feature1
atal: a branch named 'feature1' already exists
```

```
YAMEE@LAPTOP-UI8G38VF MINGW64 ~/mitali (feature1)
$ git branch feature

YAMEE@LAPTOP-UI8G38VF MINGW64 ~/mitali (feature1)
$ git checkout -b feature
fatal: a branch named 'feature' already exists

YAMEE@LAPTOP-UI8G38VF MINGW64 ~/mitali (feature1)
$ git checkout -b feature-user
Switched to a new branch 'feature-user'

YAMEE@LAPTOP-UI8G38VF MINGW64 ~/mitali (feature-user)
$ git switch feature
D       New.txt.txt
Switched to branch 'feature'

YAMEE@LAPTOP-UI8G38VF MINGW64 ~/mitali (feature)
$ git checkout -b feature
fatal: a branch named 'feature' already exists

YAMEE@LAPTOP-UI8G38VF MINGW64 ~/mitali (feature)
$ git checkout main
error: pathspec 'main' did not match any file(s) known to git

YAMEE@LAPTOP-UI8G38VF MINGW64 ~/mitali (feature)
$ git merage feature-user
git: 'merage' is not a git command. See 'git --help'.

The most similar command is
        merge

YAMEE@LAPTOP-UI8G38VF MINGW64 ~/mitali (feature)
$ git merge featutre-user
merge: featutre-user - not something we can merge

YAMEE@LAPTOP-UI8G38VF MINGW64 ~/mitali (feature)
$ Echo " Hello Friends" > user.txt

YAMEE@LAPTOP-UI8G38VF MINGW64 ~/mitali (feature)
$ git add user.txt
warning: in the working copy of 'user.txt', LF will be replaced by CRLF the next time Git touches it

YAMEE@LAPTOP-UI8G38VF MINGW64 ~/mitali (feature)
$ git commit -m " Branching & Mergeing with user.txt"
[feature 4db6ad3]  Branching & Mergeing with user.txt
 1 file changed, 1 insertion(+)
 create mode 100644 user.txt

YAMEE@LAPTOP-UI8G38VF MINGW64 ~/mitali (feature)
$ git checkout master
error: The following untracked working tree files would be overwritten by checkout:
        test.txt
Please move or remove them before you switch branches.
Aborting

YAMEE@LAPTOP-UI8G38VF MINGW64 ~/mitali (feature)
$ git merge feature-user
Already up to date.
```

```
AMEE@LAPTOP-UI8G38VF MINGW64 ~/mitali (feature)
 git branch -d feature
rror: cannot delete branch 'feature' used by worktree at 'C:/Users/YAMEE/mitali'

AMEE@LAPTOP-UI8G38VF MINGW64 ~/mitali (feature)
 git push origin master
verything up-to-date

AMEE@LAPTOP-UI8G38VF MINGW64 ~/mitali (feature)
 git branch -d feature-user
eleted branch feature-user (was 094e1aa).

AMEE@LAPTOP-UI8G38VF MINGW64 ~/mitali (feature)
 ^C

AMEE@LAPTOP-UI8G38VF MINGW64 ~/mitali (feature)
```

Practical No: 03

Practical Title: Implement GitHub Operations using Git • Cloning a Repository • Making Changes and Creating a Branch • Push/Pull Changes to GitHub

Roll No:

Date: **Remark:**

---

1. Cloning a Repository

Cloning means downloading a GitHub repository to your local system.

Command:
git clone https://github.com/username/repository.git

2. Making Changes and Creating a Branch

• Move into the project folder

cd repository

• Create a new branch

git checkout -b feature-branch

• Add files to staging

git add .

• Commit your changes

git commit -m "Added new features"

3. Push/Pull Changes to GitHub

• Push your branch to GitHub

git push origin feature-branch

• Switch to main branch

git checkout main

• Pull latest changes from GitHub

git pull origin main

• Merge your branch into main

git merge feature-branch

• Push the updated main branch

git push origin main

//Ouput:

```
YAMEE@LAPTOP-UI8G38VF MINGW64 ~/mitali (feature-branch)
$ git checkout -b feature-branch
fatal: a branch named 'feature-branch' already exists

YAMEE@LAPTOP-UI8G38VF MINGW64 ~/mitali (feature-branch)
$ Echo " new file" > feature.txt

YAMEE@LAPTOP-UI8G38VF MINGW64 ~/mitali (feature-branch)
$ git add feature.txt
warning: in the working copy of 'feature.txt', LF will be replaced by CRLF the next time Git touches it

YAMEE@LAPTOP-UI8G38VF MINGW64 ~/mitali (feature-branch)
$ git commit -m " Additional information feature.txt"
[feature-branch 06f42d3]  Additional information feature.txt
 1 file changed, 1 insertion(+)
 create mode 100644 feature.txt

YAMEE@LAPTOP-UI8G38VF MINGW64 ~/mitali (feature-branch)
$ git push origin feature-branch
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (8/8), 759 bytes | 759.00 KiB/s, done.
Total 8 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
remote:
remote: Create a pull request for 'feature-branch' on GitHub by visiting:
remote:      https://github.com/28Mitali/Mitali-/pull/new/feature-branch
remote:
To https://github.com/28Mitali/Mitali-.git
 * [new branch]      feature-branch -> feature-branch

YAMEE@LAPTOP-UI8G38VF MINGW64 ~/mitali (feature-branch)
$ git pull origin main
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 1 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 863 bytes | 9.00 KiB/s, done.
From https://github.com/28Mitali/Mitali-
 * branch            main       -> FETCH_HEAD
 * [new branch]      main       -> origin/main
fatal: refusing to merge unrelated histories

YAMEE@LAPTOP-UI8G38VF MINGW64 ~/mitali (feature-branch)
$ git merge feature-branch
Already up to date.

YAMEE@LAPTOP-UI8G38VF MINGW64 ~/mitali (feature-branch)
$ git push origin main
error: src refspec main does not match any
error: failed to push some refs to 'https://github.com/28Mitali/Mitali-.git'

YAMEE@LAPTOP-UI8G38VF MINGW64 ~/mitali (feature-branch)
$ git push origin feature-branch
Everything up-to-date
```

Practical No: 04

Practical Title: Create account on docker step by steps.

Roll No:

Date: **Remark:**

- Open your browser and go to https://hub.docker.com/.
- Click Sign Up (or Create Account).
- Fill the sign-up form:

  Username: choose a unique username (no spaces; it will be part of your image names).

  Email: enter a valid email you can access.

  Password: pick a strong password.

  Accept the Terms of Service / Privacy Policy.
- Click Create Account or Sign Up.
- Open your email inbox (the one you used to sign up).
- Find the Docker verification email and click the verification link to activate your account.
- Log in to Docker Hub (https://hub.docker.com/) with your new credentials to confirm it works.
- Docker Account Created Successfully.

Practical No: 05

Practical Title: Demonstrate a practical on Version Control Tools.

Roll No:

Date: **Remark:**

Step 1: Create a New Folder

1. Create a folder named **GitPractice** on your desktop.
2. Open **Git Bash** inside this folder.

   mkdir GitPractice

   cd GitPractice

Step 2: Initialize Git Repository

git init

This command creates an empty Git repository.

Step 3: Create a Sample File

Create a file sample.txt a

Echo “This is my first version” > sample.txt

Save the file inside **GitPractice**.

Step 4: Stage Files

git add sample.txt

or to stage all files:

git add .

Step 5: Commit the Changes

git commit -m "Initial commit with sample file"

Step 6: Create a Branch

git checkout -b feature-update

This creates and switches to a new branch.

Step 7: Make Changes in the File

Edit sample.txt:

This is the updated content in the feature branch.

Step 8: Stage and Commit Again

git add sample.txt

git commit -m "Updated sample.txt in feature branch"

Step 9: Merge the Branch into Main

Switch back to main:

git checkout main

Merge:

git merge feature-update

Step 10: Connect to GitHub

1. Create a new GitHub repository (do not add README).
2. Copy the repo link.
3. Connect local repo with GitHub:

git remote add origin **https://github.com/28Mitali/Mitali-.git**

Step 11: Push Changes to GitHub

git push origin main

Step 12: Pull Changes from GitHub (if updates done online)

git pull origin main

Practical No: 06

Practical Title: Create a merge request on gitlab and Review the merge request.

Roll No:

Date: **Remark:**

---

1. Creating a Merge Request on GitLab

A Merge Request (MR) is used to propose changes from a branch to another branch (usually main or master).

Step 1: Install Git: Ensure Git is installed on your local machine

Step 2: Clone the Repository: Clone the project repository from GitLab to your local environment:

git clone https://gitlab.com/users/sign_in

Step 3: Create and Push a Feature Branch

git checkout -b feature-branch

Make your changes → stage → commit:

git add .

git commit -m "Added new feature"

Push the branch to GitLab:

git push -u origin feature-branch

Step 4: Open GitLab Project

1. Go to https://gitlab.com
2. Open your project repository
3. You will see a message:

"A new branch has been pushed. Create a merge request?"

Click Create merge request

or

Go to

Repository → Branches → feature-branch → New merge request

Step 5: Fill Merge Request Details

You will see an MR form:

Important fields to fill:

- Source Branch: feature-branch
- Target Branch: main
- Title: Clear summary (e.g., Add new login feature)
- Description: Explain what you changed
- Assignee: Person who will review
- Labels / Milestones (optional)

Click Submit merge request.

2. Reviewing a Merge Request in GitLab

Step 1: Open the Merge Request

Go to:

Menu → Merge Requests → Assigned to me

Open the MR.

Step 2: Review Code Changes

Go to the Changes tab.

Step 3: Approve or Request Changes

On the right side, click:

Approve

If everything looks correct.

Request Changes

If developer must fix something.

Step 4: Merge the Request

Once approved, click:

Merge

This will merge the branch into main.

Select merge method:

- Merge commit (default)
- Squash commits
- Rebase and merge

Practical No: 07

Practical Title: To study docker file intructions, build an image for a sample web application using docker file.

Roll No:

Date: **Remark:**

---

1. What is a Dockerfile?

A **Dockerfile** is a text file containing a set of instructions used by Docker to automatically **build an image**.

2. Common Dockerfile Instructions

Instruction	Description
FROM	Sets the base image
COPY	Copies files from local machine into container
ADD	Similar to COPY, but can fetch from URLs
RUN	Runs commands during image build (install packages etc.)
WORKDIR	Sets working directory inside container
EXPOSE	Tells Docker which port the container will use
CMD	Default command to run when container starts
ENTRYPOINT	Main entry script for the container
ENV	Sets environment variables
MAINTAINER	Author details (deprecated)

---

3. Sample Web Application (index.html)

Create a folder:

```
mywebapp/
 ├── index.html
 └── Dockerfile
```

index.html

```
<html>
  <body>
    <h1>Welcome to My Docker Web App</h1>
```

```
  </body>
</html>
```

4. Create the Dockerfile

Dockerfile

```
# Step 1: Use a lightweight web server
FROM nginx:latest

# Step 2: Copy HTML file to Nginx directory
COPY index.html /usr/share/nginx/html/

# Step 3: Expose port
EXPOSE 80

# Step 4: Start Nginx
CMD ["nginx", "-g", "daemon off;"]
```

Explanation of instructions used:

- FROM nginx:latest → uses official Nginx server as base
- COPY index.html ... → adds your webpage to the container
- EXPOSE 80 → tells Docker this app runs on port 80
- CMD [...] → runs Nginx in foreground

5. Build Docker Image

Go to project folder and run:

```
docker build -t mywebapp .
```

This creates an image named **mywebapp**.

6. Run Docker Container

```
docker run -d -p 8080:80 mywebapp
```

Now open your browser and visit:

http://localhost:8080

Your web app will be displayed.

7. Verify Running Container

```
docker ps
```