Experiment No. 03:

Creating and Managing GitHub Repositories for Collaborative Development with Pull Requests, Code Reviews, and Project Management Tools

Aim:

To understand the fundamental features of GitHub by creating and managing repositories, connecting them with local Git repositories, and collaborating effectively using pull requests, code reviews, and GitHub's project management tools.

Objective:

- ✧ Create a GitHub repository.
- ✧ Connect it to a local Git repository.
- ✧ Collaborate on a project using pull requests and code reviews.
- ✧ Explore GitHub's project management tools.

Step-by-Step Instructions

---

1. Creating a GitHub Repository

- Log in to https://github.com.
- Click on the "+" sign > New repository.
- Enter repository name (e.g., demo-project).
- Choose Public or Private.
- (Optional) Initialize with a README.
- Click Create repository.

---

2. Setting Up Local Git Repository

a. Install Git (if not already installed):

```
sudo apt install git     # For Ubuntu/Linux
git --version           # To check installation
```

b. Configure Git (once per system):

```
git config --global user.name "Your Name"
git config --global user.email "your_email@example.com"
```

c. Clone the GitHub repository to local system:

```
git clone https://github.com/username/demo-project.gitcd demo-project
```

OR, if you already have a local project:

```
cd existing-project
git init
git remote add origin https://github.com/username/demo-project.git
git add .
git commit -m "Initial commit"
git push -u origin master
```

---

3. Working with Remote Repositories

Command	Purpose
git remote -v	Check remote repository URL
git push	Push local commits to GitHub
git pull	Fetch + merge changes from GitHub
git fetch	Fetch changes only (manual merge needed)

```
git remote -v
git add .
git commit -m "Added new feature"
git push
```

---

4. Collaborating via Pull Requests and Code Reviews

a. Forking a Repository:

Collaborator visits the repository → clicks Fork to make a copy in their GitHub account.

b. Clone forked repo locally:

```
git clone https://github.com/collaborator/forked-repo.git
```

c. Create a new branch and make changes:

```
git checkout -b new-feature# Make changes to code
git add .
git commit -m "Added new feature"
git push origin new-feature
```

d. Create a Pull Request:

- Go to GitHub → your fork → click Compare & pull request.
- Add description, reviewers, and click Create pull request.

e. Code Review:

- ✓ Original repository owner reviews the pull request.
- ✓ Add comments or suggest changes.
- ✓ Merge the pull request if approved.

---

5. GitHub Issues and Projects for Project Management

a. Create Issues:

Go to the repository → Issues tab → New Issue.

Enter bug/feature description and submit.

b. Use Projects:

- Go to Projects tab → New project.
- Choose Board or Table layout.
- Add cards (tasks) from issues or manually.
- Use columns like *To do*, *In Progress*, *Done*.

Expected Outcome:

By the end of this experiment, students will be able to:

- ✓ Create and manage repositories on GitHub.
- ✓ Connect a local Git repository to a remote GitHub repository.
- ✓ Perform version control operations using Git commands (push, pull, fetch, etc.).
- ✓ Collaborate on projects using pull requests and code reviews.
- ✓ Utilize GitHub Issues and Projects for tracking tasks and managing team collaboration.

Conclusion:

You have now learned:

- ✓ How to create and manage GitHub repositories.
- ✓ Work with remote repositories using Git commands.
- ✓ Collaborate with others using pull requests and code reviews.
- ✓ Use GitHub Issues and Projects for effective project tracking.

Experiment No. 3: Practical with Examples

Example Project:
Repository Name: simple-calculator
Project Description: A Python-based calculator that supports basic arithmetic operations.

Step-by-Step Practical with Examples

Step 1: Create a GitHub Repository

Action:

Log in to GitHub.
Click **New Repository**.
Fill the form:

- **Repository name**: simple-calculator
- **Description**: A Python-based calculator
- **Visibility**: Public
- **Check**: Add a README file

Click **Create repository**

Result: Your repository URL will be:

Output:
https://github.com/your-username/simple-calculator

Step 2: Clone the Repository to Local System
Command:

```
git clone https://github.com/your-username/simple-calculator.gitcd simple-calculator
```

Output:
A folder named simple-calculator will be created with a README file.

Step 3: Create a Python File Locally

File Name: calculator.py

Content:

```
def add(a, b):
   return a + b
def subtract(a, b):
   return a - b
print("Sum:", add(10, 4))print("Difference:", subtract(10, 4))
```

Output:
Save this file inside the simple-calculator folder.

Step 4: Commit and Push Changes to GitHub

Commands:

```
git add calculator.py
git commit -m "Added add and subtract functions"
git push origin main
```

Output:

Result: calculator.py is now uploaded to your GitHub repository.

Step 5: Collaborator Forks the Repository

Action by Collaborator:

Open the original repo:
https://github.com/your-username/simple-calculator
Click **Fork** (top right).

Output:

Result:
They get their own copy at:
https://github.com/collaborator-username/simple-calculator

Step 6: Collaborator Adds a New Feature

Commands by Collaborator:

```
git clone https://github.com/collaborator-username/simple-calculator.git
cd simple-calculator
git checkout -b multiply-feature
```

(is used to **create a new branch and switch to it immediately**.)

Edit calculator.py**:**

```
def multiply(a, b):
   return a * b
print("Product:", multiply(10, 4))
```

Push Changes:

```
git add calculator.py
git commit -m "Added multiply function"
git push origin multiply-feature
```

Step 7: Collaborator Creates a Pull Request

Action:

Visit: https://github.com/collaborator-username/simple-calculator

GitHub shows option: **Compare & pull request(PR)**
Add message:
Title: "Added multiply function"
Description: This PR adds a multiply function to calculator.py.
Click **Create pull request**

Step 8: Owner Reviews and Merges the PR

Action by Owner:

Go to **Pull Requests** tab
Click the new PR → review code → approve it
Click **Merge pull request** → Confirm

Step 9: Owner Updates Local Repo

Commands:

```
git pull origin main
```

Output:

Result:
Owner's local copy is now updated with the multiply feature.

Check Locally: Step-by-step:

- Open terminal and go to your project folder:

```
cd simple-calculator
```

- Ensure you're on the **main branch**:

```
git checkout main
```

- Pull latest changes (in case of a recent merge):

```
git pull origin main
```

- Open the file to check:

```
cat calculator.py
```

You will see the contents printed in the terminal.

Optional (Using Code Editor):

Open the folder in VS Code or any editor:

```
code .
```

Step 10: Use GitHub Issues and Projects

Create an Issue
Go to **Issues → New Issue**
Title: *Add division function*

Description: Implement a divide(a, b) function and handle division by zero.
Click **Submit new issue**

Create a Project Board

Go to **Projects → New project**
Template: **Board**
Name: Calculator Development
Add columns: *To Do*, *In Progress*, *Done*
Add the issue as a card to *To Do*

Summary Table:

Step	Action	Example
1	Create repo	simple-calculator
2	Clone repo	git clone <repo-url>
3	Add file	calculator.py
4	Push code	git push origin main
5	Fork repo	Collaborator forks your repo
6	New branch	git checkout -b multiply-feature
7	Pull Request	"Added multiply function"
8	Review & Merge	Merge via GitHub UI
9	Sync local	git pull origin main
10	Issues/Projects	Track features, bugs
