Ultimate Git Contribution Guide

A hybrid design (professional + developer handbook) - step-by-step instructions to fork, clone, branch, and open a pull request. Designed for absolute beginners and seasoned developers.

Overview

This guide covers everything from zero-git knowledge to creating a professional Pull Request (PR) on GitHub. Follow the steps exactly, use the copyable commands, and apply the checklist before submitting your PR.

Quick Glossary

- Git: version control system for tracking changes and collaborating.
- GitHub: platform hosting git repositories and collaboration tools.
- Repo: repository - the project code and history.
- Fork: your personal copy of someone else's repo on GitHub.
- Clone: local copy of a repo on your computer.
- Origin: remote pointing to your fork.
- Upstream: remote pointing to the original project.
- Branch: isolated workspace for changes.
- Commit: snapshot of changes with a message.
- Push: upload commits to your fork on GitHub.
- Pull Request (PR): request to merge your branch into upstream.

Prerequisites

Install Git and set up your identity. Have a GitHub account. Use a terminal (Command Prompt, PowerShell, or bash) and an editor like VS Code.

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

Step-by-Step Workflow

Follow these steps in order. Replace placeholders like <your-username> and <branch-name> with your information.

Step 1 - Fork the Upstream Repository

1) On GitHub, open the upstream repository page (example: https://github.com/nishant0820/Student-Result-Management-System).

2) Click 'Fork' (top-right). This creates a copy at: https://github.com/<your-username>/Student-Result-Management-System

Step 2 - Clone Your Fork Locally

Open terminal and run:

```
git clone https://github.com/<your-username>/Student-Result-Management-System.git
cd Student-Result-Management-System
```

Now you are inside the local copy of your fork. All work should happen inside this folder.

Step 3 - Add Upstream Remote (Recommended)

Link the original repository so you can keep your fork updated:

```
git                        remote                          add                        upstream
https://github.com/nishant0820/Student-Result-Management-System.git
git fetch upstream
git remote -v
```

Verify you see both 'origin' (your fork) and 'upstream' (original repo).

Step 4 - Create a Feature Branch

Never edit main. Create an isolated feature branch:

```
git checkout -b feat/brief-description
```

Use a clear branch name: feat/, fix/, chore/, docs/ prefixes help maintainers understand intent.

Step 5 - Copy Your Updated Files into the Cloned Folder

If you developed changes elsewhere, copy only the changed files into your cloned repo. Typical files to copy:

- dashboard.py, student.py, result.py, report.py, course.py, create_db.py
- images/ (small icons), reports/
- Any helper scripts you added

Confirm the files in your cloned folder by opening them in your editor and verifying your new code exists.

Step 6 - Stage and Commit Changes

Stage and commit with clear messages:

```
git add .
git status
git commit -m "feat: support multiple courses per student and per-student PDF export"
```

Write concise commit messages. Use present-tense imperative verbs (feat:, fix:, docs:, chore:).

Step 7 - Push Your Branch to Your Fork (origin)

Push the branch so GitHub can host it:

```
git push -u origin feat/brief-description
```

After pushing, GitHub shows 'Compare & pull request' - click it to start the PR.

Step 8 - Create the Pull Request (PR)

On the PR page ensure the following selectors are correct:

- Base repository: original owner (e.g., nishant0820/Student-Result-Management-System)
- Base branch: main (or project's default branch)
- Compare: your fork and branch (e.g., Basava05:feat/...)

Fill PR title and description with clarity. Provide testing steps and attach screenshots for UI

changes.

PR Title and Description Template

Use this as a starting point. Edit to suit the changes you made.

```
PR Title:
feat: multi-course enrollment per student + subject-wise results + PDF export

PR Description:
## Summary
Short description of what and why.

## Changes
- Added enrollment table for many-to-many mapping
- Updated student.py, result.py, report.py

## How to test
1. Run create_db.py
2. Add course(s)
3. Add student and enroll them in multiple courses
4. Add subject results and export PDF

Notes: backward compatible
```

Checklist - What to Check Before PR

- Run code locally and verify flows (add course, add student, add result, export PDF).
- Remove debug prints and console logs.
- No secrets or credentials in code.
- Avoid committing large binaries; use .gitignore.
- Add README or migration notes if DB schema changed.
- Keep commits focused and branch-specific.

Common Mistakes to Avoid

- Editing directly on main or the owner's repo.
- Mixing unrelated changes in one PR.
- Pushing sensitive data or credentials.
- Not syncing with upstream before creating a branch.
- Large files in repo history (use git-lfs if necessary).

Troubleshooting - Common Errors and Fixes

Error: 'nothing to commit, working tree clean' - ensure you copied files into the cloned folder and saved them.

Error: 'src refspec does not match any' - create branch and commit before pushing.

Error: 'permission denied (publickey)' - set up SSH keys or use HTTPS and provide credentials.

Sync with upstream if master/main moved ahead: git fetch upstream; git checkout main; git merge upstream/main; git push origin main.

Commands Cheat Sheet (copyable)

```
git clone <your-fork-url>
git remote add upstream <upstream-url>
```

```
git fetch upstream
git checkout -b feat/short-description
git add .
git commit -m "short clear message"
git push -u origin feat/short-description
```

Final Tips - Write a Strong PR

Explain WHY you made changes, not only WHAT. Add screenshots for UI. Be responsive to reviewer comments and push fixes to the same branch. Offer to split the PR if it's too large.