# **Git Workflow Cheat Sheet (Windows + VS Code + PowerShell)**

Run all commands from your project root (folder with app.py).  
 Make sure your venv is activated for Python work:  
 . .\.venv\Scripts\Activate.ps1

## **0) One-time setup (if needed)**

git init

git remote add origin https://github.com/<your-username>/planception.git

git branch -M main

git push -u origin main

## **1) Daily update flow (the 80% you’ll use)**

git status # see what changed

git add . # stage everything (or list files explicitly)

git commit -m "feat: brief message of what changed"

git push origin main # upload to GitHub

**Good commit messages**

* feat: add edit-task route
* fix: correct overdue date comparison
* docs: add SRS and diagrams
* test: add flash message test
* refactor: split repo/service logic

## **2) Pull latest changes (stay in sync)**

git pull origin main # bring remote changes to your local

If you get a merge conflict, open the file in VS Code, resolve the <<<<<<< markers, then:

git add .

git commit -m "merge: resolve conflicts in X"

git push

## **3) Feature branches (recommended for V2 changes)**

# create & switch to a new branch

git checkout -b feature/edit-task

# do your work... add/commit as usual

git add .

git commit -m "feat: edit task form + route"

# push the branch

git push -u origin feature/edit-task

When finished, either merge in GitHub via Pull Request, or locally:

git checkout main

git pull origin main

git merge feature/edit-task

git push origin main

## **4) See history & differences**

git log --oneline --graph --decorate --all

git diff # see unstaged changes

git diff --staged # see what you’re about to commit

## **5) Undo/rollback (safely)**

**Unstage a file (keep changes):**

git reset HEAD <file>

**Amend the last commit message (no new commit):**

git commit --amend -m "fix: better message"

git push --force-with-lease # only if you already pushed

**Revert a bad commit (make a new inverse commit):**

git log --oneline # find the SHA (e.g., abc123)

git revert abc123

git push

**Throw away local changes in a file (careful!):**

git checkout -- <file>

## **6) Stash work-in-progress (WIP)**

If you need to quickly switch branches but keep current edits:

git stash push -m "wip: editing forms"

git checkout main

# later…

git checkout feature/edit-task

git stash list

git stash pop # reapplies the latest stash

## **7) Tags (mark versions for your report/demo)**

git tag -a v1.0.0 -m "V1: MVP scaffold"

git push origin v1.0.0

git tag -a v1.1.0 -m "V1.1: docs & diagrams"

git push origin v1.1.0

## **8) .gitignore (keep junk out of Git)**

Create a .gitignore in the project root:

# Python

\_\_pycache\_\_/

\*.pyc

# Virtual env

.venv/

# Flask/SQLite

instance/

\*.sqlite

# Coverage/pytest caches

htmlcov/

.coverage

.pytest\_cache/

# OS/Editor

.DS\_Store

Thumbs.db

.vscode/

Note: You *do* want instance/ in .gitignore (it contains your local DB). Your prof doesn’t need your local data file.

## **9) Handy aliases (optional, speeds you up)**

git config --global alias.st "status -sb"

git config --global alias.lg "log --oneline --graph --decorate --all"

Now you can run:

git st

git lg

## **10) Typical V2 branch workflow (example)**

# start V2 (edit & sorting)

git checkout -b feature/edit-and-sorting

# code… then:

git add .

git commit -m "feat: add edit form and route"

git commit -m "feat: add sorting by due date then priority"

git push -u origin feature/edit-and-sorting

# open PR on GitHub (optional) or merge locally:

git checkout main

git pull origin main

git merge feature/edit-and-sorting

git push

git tag -a v2.0.0 -m "V2: editing + sorting"

git push origin v2.0.0

### **Quick FAQ**

* **“Why does python -m pytest work but pytest fails?”** Environment/path quirk. Use python -m pytest -q, or add a pytest.ini with pythonpath=..
* **“I pushed to the wrong branch.”** git checkout main → git merge <that-branch> → git push → optionally delete remote branch.
* **“I messed up a commit I already pushed.”** Prefer git revert <sha> (safe). Avoid rewriting published history unless you’re sure: git push --force-with-lease.