CSC 3400 Software Engineering

Assignment: Git and Version Control Fundamentals

Overview

This assignment will introduce you to the essential concepts and practices of version control using Git. You will learn to manage code changes, collaborate with others, and maintain project history through hands-on exercises.

Learning Objectives

By completing this assignment, you will be able to:

- Set up and configure Git for your development environment
- Create and manage Git repositories
- Track changes using commits, branches, and merges
- Collaborate using remote repositories
- Resolve merge conflicts
- Use Git best practices for software development

Prerequisites

- Basic command line familiarity
- Text editor of choice
- Git installed on your system
- GitHub account (free)

Assignment Structure

This assignment consists of 5 parts that build upon each other. **Complete them in order.**

Part 1: Git Setup and Basic Operations

Task 1.1: Initial Setup

1. Configure your Git identity:

```
git config --global user.name "Your Full Name"
git config --global user.email "your.email@university.edu"
```

2. Set up a default branch name and editor:

```
git config --global init.defaultBranch main
git config --global core.editor "your-preferred-editor"
```

Task 1.2: Create Your First Repository

1. Create a new directory called `software_project`
2. Initialize it as a Git repository
3. Create a `README.md` file with:
   - Project title: "My Software Engineering Project"
   - Your name and student ID
   - Brief description of what this repository will contain
4. Add and commit the README file with the message "Initial commit: Add project README"

Task 1.3: Basic File Operations

1. Create a simple Python file called `calculator.py` with basic arithmetic functions (add, subtract, multiply, divide)
2. Create a `.gitignore` file that ignores:
   - `*.pyc` files
   - `__pycache__/` directories
   - `.env` files
3. Add both files to staging and commit with message "Add calculator module and gitignore"

Deliverable: Screenshot of `git log --oneline` showing your commits

Part 2: Working with Branches

Task 2.1: Feature Branch Development

1. Create and switch to a new branch called `feature/advanced-operations`
2. Modify `calculator.py` to add two new functions: `power()` and `square_root()`
3. Add input validation to all functions to handle division by zero and negative square roots
4. Commit your changes with message "Add advanced mathematical operations"

Task 2.2: Parallel Development

1. Switch back to the `main` branch
2. Create another branch called `feature/user-interface`
3. Create a new file `main.py` that provides a simple command-line interface for the calculator
4. The interface should allow users to select operations and input numbers
5. Commit with message "Add command-line user interface"

Task 2.3: Branch Management

1. Switch back to `main` branch
2. Merge the `feature/advanced-operations` branch
3. Delete the merged feature branch
4. List all branches (local and remote)

Deliverable:

- Screenshot of branch history using `git log --graph --oneline --all`
- Copy of your `calculator.py` and `main.py` files

Part 3: Remote Repositories and Collaboration

Task 3.1: GitHub Setup

1. Create a new repository on GitHub called `csc3400-git-assignment`
2. Add your local repository as a remote origin
3. Push your `main` branch to GitHub
4. Push the remaining `feature/user-interface` branch

Task 3.2: Collaborative Workflow Simulation

1. Create a new branch called `feature/error-handling`
2. Add comprehensive error handling to your calculator functions
3. Create a `tests.py` file with at least 5 test cases for your calculator functions
4. Push the branch to GitHub
5. Create a pull request on GitHub (but don't merge it yet)
6. Add meaningful comments to your pull request describing the changes

Task 3.3: Repository Management

1. Clone your repository to a different directory (simulating a teammate's environment)
2. From the cloned repository, create a new branch `hotfix/readme-update`
3. Update the README.md with:
    - Installation instructions
    - Usage examples
    - List of features
4. Push the hotfix branch and create another pull request

Deliverable:

- URL to your GitHub repository
- Screenshots of both pull requests

Part 4: Merge Conflicts and Resolution

Task 4.1: Create a Conflict

1. In your original repository, switch to `main` branch
2. Modify the first line of `README.md` to change the project title
3. Commit and push the change
4. In your cloned repository (different directory):
    - Modify the same first line of `README.md` with a different title
    - Commit the change
    - Try to push (this should fail due to conflicts)

Task 4.2: Resolve the Conflict

1. Pull the latest changes from the remote repository
2. Resolve the merge conflict by:
    - Keeping both titles in a meaningful way
    - Adding a subtitle that combines both ideas
3. Complete the merge and push the resolved changes

Task 4.3: Advanced Conflict Resolution

1. Create conflicting changes in your `calculator.py` file from both repositories
2. Practice resolving conflicts using:
    - Command line merge tools

- Your preferred editor/IDE merge resolution tools

3. Document the conflict resolution process in a new file called `CONFLICT_RESOLUTION.md`

Deliverable:

- Screenshot of the conflict resolution process
- Final merged files
- Your conflict resolution documentation

Part 5: Git Best Practices and Workflow

Task 5.1: Git Workflow Documentation

Create a file called `GIT_WORKFLOW.md` that documents:

1. Your team's branching strategy
2. Commit message conventions
3. Code review process
4. Release workflow
5. Common Git commands reference

Task 5.2: Repository Cleanup

1. Merge all completed feature branches
2. Delete merged branches (local and remote)
3. Tag your final version as `v1.0.0`
4. Update README with project status and version information

Deliverable:

- Clean repository with proper tagging
- Complete workflow documentation
- Final project structure

Submission Requirements

What to Submit:

1. **GitHub Repository URL** - Your complete project repository (chaynes4@flsouthern.edu should have access to view!)
2. **Screenshots Portfolio** including:
   - Git log showing commit history
   - Branch visualization
   - Pull requests
   - Conflict resolution process
3. **Final Code Files** - All Python files with proper documentation

Submission Format:

- Create a ZIP file named `LastName_FirstName_GitAssignment.zip`
- Include all required files and screenshots
- Submit via Canvas

Grading Criteria:

- This assignment is Pass/Fail.

Late Submission Policy:

- No late submissions accepted.

Additional Resources

Helpful Tools:

- **Git GUI**: GitKraken, SourceTree, various VSCode Plugins, Sublime Merge, or built-in IDE Git tools

Getting Help:

- Git documentation: `git help <command>`