Git Assignment 7

Problem Statement 7

Git Ignore and Stash

- Utilize a .gitignore file to exclude specific files or directories from version control.
- Use the git stash command to temporarily save changes and switch between different branches.

GitHub link for Assignment 7

https://github.com/RahulRGolabhavi/UST-GIT-Assignment-7

Steps to Complete the Assignment

Step 1: Use .gitignore to Exclude Files and Directories

1. **Create a .gitignore File**:
   - In your project root directory, create a .gitignore file:

     ```
     touch .gitignore
     ```

2. **Add Patterns to .gitignore**:
   - Edit the .gitignore file and add the files or directories you want to exclude. For example:

   ```
   # Ignore all .log files
   *.log
   # Ignore the build directory
   /build/
   # Ignore temporary files
   *.tmp
   ```

3. **Verify Ignored Files**:
   - After adding patterns to .gitignore, Git will ignore matching files or directories. To check, run:

   ```
   git status
   ```

   - Files matching the .gitignore patterns will not appear in the output.

4. **Stage and Commit the .gitignore File**:
   - Add and commit the .gitignore file to the repository:

   ```
   git add .gitignore
   git commit -m "Add .gitignore to exclude unnecessary files"
   ```

Step 2: Use git stash to Save Temporary Changes

1. **Make Changes to Files**:
   - Modify or create files in your working directory. For example:

   ```
   echo "Temporary changes" > temp.txt
   ```

2. **View the Changes**:
    - Run the following command to view your uncommitted changes:

```
git status
```

3. **Stash the Changes**:
    - Save your uncommitted changes temporarily using git stash:

```
git stash
```

    - The working directory will now be clean, and the changes are saved in the stash.

4. **List Stashed Changes**:
    - To view the list of stashed changes, run:

```
git stash list
```

    - You will see the stashes with names like stash@{0}.

5. **Apply Stashed Changes**:
    - To reapply the stashed changes, use:

```
git stash apply
```

6. **Pop Stashed Changes**:
    - To reapply the changes and remove the stash from the list, use:

```
git stash pop
```

7. **Drop a Specific Stash**:
    - If you no longer need a stash, you can remove it:

```
git stash drop stash@{0}
```

8. **Clear All Stashes**:
    - To delete all stashes, use:

```
git stash clear
```

Step 3: Switch Between Branches with Stash

1. **Create a New Branch**:
    - Before switching branches, create a new branch if needed:

```
git checkout -b feature-branch
```

2. **Switch Branches with Uncommitted Changes**:
    - If you have uncommitted changes and want to switch to another branch, stash the changes:

```
git stash
git checkout main
```

3. **Apply Changes to the New Branch**:
    - Once on the desired branch, reapply the stashed changes:

```
git stash apply
```