Task 2: Git (Version Control)

Git Commands Practice Documentation

1. **Setting Up Global Configuration**
   - **Command:**

     ```
     $ git config --global user.name "jeel kukadiya"
     ```

   - **Description:** Sets the global Git username for all repositories.
   - **Command:**

     ```
     $ git config --global user.email "jeelkukadiya161104@gmail.com"
     ```

   - **Description:** Sets the global Git email for all repositories.
   - **Command:**

     ```
     $ git config --global core.editor "code --wait"
     ```

   - **Description:** Configures Visual Studio Code as the default editor for Git commands.
   - **Command:**

     ```
     $ git config --global -e
     ```

   - **Description:** Opens the global configuration file in the default editor.
   - **Command:**

     ```
     $ git config --global core.autocrlf true
     ```

   - **Description:** Ensures consistent line endings across operating systems.

2. **Creating a New Git Repository**
   - **Command:**

     ```
     $ mkdir project

     $ cd project

     $ git init
     ```

   - **Description:** Creates a new directory, navigates into it, and initializes an empty Git repository.
   - **Output:**

```
Initialized empty Git repository in C:/Users/jeelk/project/.git/
```

   - **Command:**

     ```
     $ ls -a
     ```

   - **Description:** Lists all files, including hidden files (e.g., .git).

3. **Creating and Managing Files**
    - **Command:**

      ```
      $ echo hello > file1.txt

      $ echo hello > file2.txt
      ```

    - **Description:** Creates two text files (file1.txt and file2.txt) with the content hello.
    - **Command:**

      ```
      $ git status
      ```

    - **Description:** Shows the status of the working directory and staging area.
    - **Output:**

```
Untracked files:

 file1.txt

 file2.txt
```

- **Command:**

  ```
  $ git add .
  ```

- **Description:** Adds all untracked files to the staging area.
- **Command:**

  ```
  $ git commit
  ```

- **Description:** Opens the configured editor to write a commit message. Finalizes the commit of staged changes.
- **Command:**

  ```
  $ git commit -m "Initial commit."
  ```

- **Description:** Commits the staged changes with the message Initial commit..

4. **Modifying Files and Staging Changes**
    - **Command:**

      ```
      $ echo world >> file2.txt

      $ git status
      ```

    - **Description:** Appends world to file2.txt and checks the status.
    - **Output:**

```
Changes not staged for commit:

 modified: file2.txt
```

- **Command:**

```
$ git add file2.txt
```

- **Description:** Stages changes made to file2.txt.
- **Command:**

```
$ git commit -am "Fix the bug that prevented the user from signing up."
```

- **Description:** Stages and commits all changes with a message.

5. **Deleting Files**
   - **Command:**

```
$ rm file2.txt

$ git status
```

   - **Description:** Deletes file2.txt and checks the status.
   - **Command:**

```
$ git add file2.txt

$ git commit -m "Remove unused code."
```

   - **Description:** Stages and commits the deletion of file2.txt.

6. **Renaming Files**
   - **Command:**

```
$ mv file1.txt main.js

$ git add main.js

$ git commit -m "Refactor code."
```

   - **Description:** Renames file1.txt to main.js, stages the change, and commits it.

7. **Ignoring Files**
   - **Command:**

```
$ mkdir logs

$ echo "logs/" > .gitignore

$ git add .gitignore

$ git commit -m "Add gitignore."
```

   - **Description:** Creates a .gitignore file to exclude the logs/ directory from version control.

8. **Removing Files from Version Control**

- **Command:**

```
$ git rm --cached -r bin/

$ git commit -m "Remove the bin directory that was accidently committed."
```

- **Description:** Removes the bin/ directory from the repository while keeping it locally.

9. **Checking the Repository State**
   - **Command:**

   ```
   $ git ls-files
   ```

   - **Description:** Lists all tracked files in the repository.

10. **Additional Commands**
    - **Command:**
      - $ss git rm -h

- **Description:** Displays help for the git rm command.

Summary

This documentation includes essential Git commands for configuring a user, initializing a repository, managing files, and maintaining a clean repository state. These commands provide a foundational understanding of Git for version control.