**Lab#1 GIT**

***Basics:***

**What is version control? Why is it important?**

A] Version control helps us see who changed what in a project and when. It's like a shared diary for a team's work. It lets us try out new ideas without breaking the main project. It's like having a playground where we can test stuff before making it official.

**What's the difference between Git and GitHub?**

A] Git is a tool for version control, tracking changes in code. It works on your computer. GitHub is a platform using Git for remote collaboration, hosting projects online, and enabling teamwork through features like pull requests and issue tracking.

**Describe the Git workflow (add, commit, push, pull).**

A] **Add:** Use **git add** to stage changes from your working directory, preparing them for commit.

**Commit:** With **git commit**, save the staged changes along with a meaningful message describing what you've done.

**Push:** Push your committed changes using **git push** to send them to a remote repository, like GitHub.

**Pull:** To get others' changes or updates, use **git pull** to fetch and merge remote changes into your local branch

**What is a repository in the context of Git?**

A] In the context of Git, a repository is a folder where your project's files and version history are stored. It contains all the code, commits, and branches for effective version control.

***Commits:***

**What is a commit in Git?**

A] In Git, a commit is a snapshot of the changes you've made to your project's files. It's like a save point that records modifications along with a message describing the purpose of the changes.

**How is each commit uniquely identified?**

A] Each commit in Git has a unique identifier called a hash, generated based on its content and metadata, ensuring distinct identification in the version history.

***Remote Repositories:***

**What is a remote repository in the context of Git?**

A] In Git, a remote repository is a version of your project hosted on a different server, often online platforms like GitHub. It allows collaboration and sharing of code between team members, enabling synchronization of changes across different locations.

**What are the default names that Git uses for the repository you cloned from and your local repository?**

**A]** When you clone a Git repository:

1. The remote repository is named "origin" by default.

2. Your local repository takes the name of the project directory.

3. You can view remote details with `git remote -v`.

4. Customization is possible for these names.

**How do you synchronize changes from a remote repository to your local one, and vice versa?**

A] To synchronize changes:

1. Fetch remote changes: `git fetch origin`.

2. Merge or rebase: `git merge origin/main` or `git rebase origin/main`.

3. Commit your local changes: `git commit -m "Your message"`.

4. Push to remote: `git push origin main`.

***GitHub Specifics:***

**What is a pull request?**

**A]** A pull request (PR) is a request to merge changes from one branch into another in a Git-based version control system. It's commonly used for code review and collaboration in software development.

**How do you 'fork' a repository on GitHub, and why might you want to?**

**A]** To fork a GitHub repository:

1. Visit the repository you want to fork.

2. Click the "Fork" button in the upper right corner.

3. Choose where to fork (your account or an organization).

4. Forking is useful for contributing to open source, creating project templates, experimenting, collaborating, and keeping your fork updated with changes from the original repository.

**How can you use GitHub to collaborate on open-source projects?**

A] To collaborate on open-source projects on GitHub:

1. Fork the project's repository.

2. Clone your fork locally.

3. Create a branch for your changes.

4. Make and commit your changes.

5. Push your branch and create a Pull Request (PR).

6. Collaborate through PR reviews.

7. Keep your fork in sync with the original repository for updates.

***Collaboration and Best Practices:***

**Why is it important to write clear commit messages?**

A] Clear commit messages are essential because they provide documentation, aid collaboration, simplify debugging, streamline code reviews, enhance version control history, support open-source contributions, and promote accountability.

**When collaborating with others, why might it be important to frequently pull the latest changes?**

**A]** Frequently pulling the latest changes in collaborative work:

1. Prevents conflicts by incorporating others' updates.

2. Keeps you current with project progress and improvements.

3. Facilitates timely feedback and avoids duplicated efforts.

4. Maintains a stable and up-to-date codebase for smoother integration.