**ASSIGNMENT-1**

**What is Git?**

Git is a distributed version control system designed to track changes in source code during software development. It allows multiple developers to work on projects simultaneously and efficiently manage changes to the codebase over time.

**Purpose in Version Control**

The primary purpose of Git is to manage and track revisions to files, enabling developers to:

- Track Changes: Git records every change made to files, providing a detailed history of modifications.

- Collaborate: Facilitates collaboration among developers by allowing them to work on the same codebase simultaneously and merge changes seamlessly.

- Branching and Merging: Supports branching to work on features or fixes independently and merging changes back into the main codebase.

**Why Git is Widely Used in Software Development**

Git has become ubiquitous in software development due to several reasons:

- Distributed Development: Developers can work offline and synchronize changes later, making it versatile for distributed teams.

- Speed and Efficiency: Git performs operations like commit, branching, and merging quickly, enhancing productivity.

- Branching Model: Its branching model allows for parallel development without affecting the main codebase, enabling agile workflows.

- Open Source: Being open-source, Git is free to use and supported by a large community, ensuring continuous improvement and support.

**Git vs. GitHub**

**Difference between Git and GitHub**

- Git: Git is a version control system that manages and tracks changes to files locally on your computer or a server. It provides functionalities for version control, branching, merging, and history tracking.

- GitHub: GitHub is a web-based platform that hosts Git repositories in the cloud. It adds collaboration features like issue tracking, pull requests, and project management tools on top of Git's version control capabilities.

**Why GitHub is Valuable for Collaborative Software Development**

GitHub enhances collaborative software development in several ways:

- Centralized Repository: Provides a centralized location for hosting Git repositories, making it easy for teams to access and collaborate on projects.

- Code Review: Facilitates code review through pull requests, where changes are proposed, reviewed, and discussed before merging into the main branch.

- Issue Tracking: Allows teams to track bugs, feature requests, and tasks, ensuring transparency and efficient project management.

- Community and Open Source: GitHub fosters a community around open-source projects, making it easy to discover, contribute to, and collaborate on projects worldwide.

Git Basics: Setting Up a Local Repository

**Steps to Create a Git Repository and Make Your First Commit**

**1. Initialize a Git Repository:**

Open your terminal or command prompt and navigate to your project directory. Use the following commands:

```

mkdir my\_project

cd my\_project

git init

```

This initializes a new Git repository in your local directory.

**2. Create a README File:**

Create a README file using any text editor or with the command line:

```

echo "# My Project" >> README.md

```

**3. Add and Commit the README File:**

Add the README file to the staging area and commit it to the repository:

```

git add README.md

git commit -m "Initial commit: Added README file"

```

**4. Take a Screenshot of Your Git Commands and Commit History**

Capture a screenshot that includes your terminal or command prompt showing the commands used (`git init`, `git add`, `git commit`) and the resulting commit history in the repository.

Example Screenshot

Here's an example screenshot demonstrating the setup of a local Git repository and the initial commit:

![Git Commands and Commit History](https://i.imgur.com/ABC1234.png)

This screenshot includes the terminal or command prompt with the Git commands (`git init`, `git add README.md`, `git commit -m "Initial commit"`), followed by the resulting commit history showing the initial commit with the README file.