# Experiment No 11

# Aim: Case study of GitHub and version control

### 1. What is Github?

GitHub is a web-based version control and collaboration platform for software developers. Microsoft, the biggest single contributor to GitHub, acquired the platform for $7.5 billion in 2018. GitHub, which is delivered through a software as a service (SaaS) business model, was started in 2008.

Git is used to store the source code for a project and track the complete history of all changes to that code. It lets developers collaborate on a project more effectively by providing tools for managing possibly conflicting changes from multiple developers.

GitHub allows developers to change, adapt and improve software from its public repositories for free, but it charges for private repositories, offering various paid plans. Each public and private repository contains all of a project's files, as well as each file's revision history. Repositories can have multiple collaborators and can be either public or private.

*Benefits and features of GitHub:

GitHub facilitates collaboration among developers. It also provides distributed version control. Teams of developers can work together in a centralized Git repository and track changes as they go to stay organized.

Other products and features of note include the following:

- **GitHub Gist** lets users share pieces of code or other notes.
- **GitHub Flow** is a lightweight, branch-based workflow for regularly updated deployments.
- **GitHub Pages** are static webpages to host a project, pulling information directly from an individual's or organization's GitHub repository.
- **GitHub Desktop** lets users access GitHub from Windows or Mac desktops, rather than going to GitHub's website.
- **GitHub Student Developer Pack** is a free offering of developer tools for students. It includes cloud resources, programming tools and support, and GitHub access.
- **GitHub Campus Experts** is a program students can use to become leaders at their schools and develop technical communities there.

## 2. What is Version control?

Version control is a system that records changes to files over time so that you can recall specific versions later. It is primarily used in software development but can be applicable to any kind of files.

The main purposes of version control are:

**1. Tracking Changes:** Version control tracks changes made to files, providing a history of modifications, additions, and deletions. Each change is typically accompanied by a description of what was altered and why.

**2. Collaboration:** Multiple developers can work on the same files concurrently without interfering with each other's work. Version control systems allow for merging changes made by different contributors.

**3. Reverting to Previous Versions:** Developers can revert to a previous version of a file or project if a mistake is made, allowing for easy error correction and recovery.

**4. Branching and Merging:** Version control facilitates the creation of branches, which are independent lines of development. Branches allow for experimentation and the implementation of new features without affecting the main project. Merging combines the changes from one branch to another.

**5. Documentation and Annotations:** Developers can provide comments and annotations to describe changes made, aiding in understanding the evolution of the project.

There are various version control systems available, with Git being one of the most widely used in the software development industry. Git, for instance, is distributed, meaning that each user has a complete copy of the repository, allowing for work to be done offline and fostering a decentralized approach to collaboration. Other version control systems include Subversion (SVN) and Mercurial.

**3. Case study of any one application using github**

**Topic:** Development of the "Atom" Text Editor

**Background:**

Atom is a popular, open-source text editor developed by GitHub. It's known for its extensibility, ease of use, and customization features. The development of Atom involves a global community of contributors and relies heavily on GitHub for version control and collaboration.

GitHub Implementation:

**1. Repository Setup:**

GitHub hosts the "Atom" repository, making it accessible to the public. The repository contains all the source code, documentation, issues, and pull requests related to the project.

**2. Collaborative Development:**

A diverse community of developers from around the world contributes to the Atom project.

Contributors can fork the repository, make changes in their own branches, and then submit pull requests to the main Atom repository to propose changes.

**3. Version Control:**

GitHub's version control system tracks all changes made to the Atom codebase over time.

Developers commit code changes to their respective branches and provide descriptive commit messages to explain the purpose of each change.

**4. Pull Requests and Code Review:**

When a contributor wants to submit a change, they create a pull request (PR) on GitHub.

The PR serves as a discussion space where other developers can review the code, provide feedback, and discuss potential improvements.

Continuous integration (CI) tools are used to automate testing, ensuring that proposed changes do not introduce regressions.

**5. Issue Tracking:**

GitHub Issues is used for tracking bugs, feature requests, and enhancements.

Contributors and users can report issues, and maintainers assign, prioritize, and label them accordingly.

Issues often serve as a starting point for contributions.

**6. Branching Strategy:**

Atom's repository uses a branching strategy that includes a "master" branch for stable releases and a "beta" branch for pre-release versions.

Contributors create feature branches to work on specific enhancements or bug fixes.

**7. Continuous Integration/Continuous Deployment (CI/CD):**

GitHub Actions is used for automating various tasks, such as building, testing, and packaging Atom releases.

This ensures that the project maintains a high level of quality and is continuously available for users.

**8. Documentation and Wiki:**

GitHub's Wiki feature is used to maintain comprehensive documentation for Atom, including installation guides, customizations, and API references.

**9. Community Involvement:**

Atom's development thrives on community involvement. GitHub provides a platform for both GitHub employees and external contributors to collaborate and improve the project.

**10. Releases and Changelogs:**

GitHub facilitates the release management process, including the creation of release notes and changelogs, making it easy for users to understand what's new in each version.

By leveraging GitHub for version control, collaboration, and issue tracking, the Atom text editor has become a widely adopted and highly extensible tool for developers and writers alike. The open-source nature of the project and the use of GitHub have fostered a thriving community and continuous improvement of the software.