ASSIGNMENT: GIT AND GITHUB

Q1. Explain what version control is and its importance in software development.

Solution:

- *Version control is a system that records changes to a file or set of files over time. It allows you to track the history of changes, collaborate with others, and manage multiple versions of a project.*
- *Version control is commonly used in software development, but it can also be applied to any context where files need to be tracked and shared.*
- *Version control system (VCS), also known as Source Code Management (SCM) is a fundamental tool used in software development and other collaborative projects to manage and track changes to files over time.*
- *VCSs are an integral part of the development process and a must-have tool for both individual developers and teams as well.*
- *Main needs for a Version control system are as below:*
    - *Tracking History changes*
    - *Collaboration or Parallel development*
    - *Backup and Disaster Recovery*
    - *Code Reviews and Quality Assurance*
    - *Experimentation and Rollback*
    - *Release Management*
    - *Traceability and Documentation*
    - *Conflict Resolution*
    - *Open Source and Collaboration Beyond Teams*

Q2. Explain the Git Workflow, including the staging area, working directory, and repository.

Solution:

- *Git is a VCS, which means it helps developers manage changes to their code over time. It allows you to track modifications, collaborate with others, and maintain a history of the project.*
- *Git has three main states that your files can reside in committed, modified, and stage. Committed means that the data is safely stored in your local storage. Modified means that you have changed the file but have not committed it to your remote storage yet. Staged means that you have marked a modified file in its current version to go into your next commit snapshot.*
- *When working with Git, understanding the concepts of the working directory, staging area, and Git repository is crucial. These three components play a significant role in managing the versions of your project and collaborating with others efficiently.*
- *The working directory, also known as the working tree, refers to the directory on your local machine where you are actively making changes to your project files. It contains all the files and directories that make up your project. You can think of it as your playground, where you create, modify, and delete files freely.*
- *We can use "git add" command to add files from working directory to staging area.*
- *The staging area, also called the index, acts as a middle ground between your working directory and the Git repository. It serves as a holding area for changes that you want to include in your next commit. When you make changes to your files in the working*

directory, you need to explicitly add them to the staging area before they become part of a commit.

- *We can use "git commit" command to commit files from staging area to Local repository.*
- *The staging area allows you to selectively choose which changes you want to include in your next commit. For example, if you have made modifications to multiple files, but only want to commit some of them, you can add only those specific files to the staging area.*
- *The Git repository contains all the commits, branches, and tags associated with your project. It is like a database that stores the complete history of your project. Whenever you make a commit, Git creates a new snapshot of your project's state and adds it to the repository.*
- *Repository is the Central storage for all files and their history.*
- *We can use "git push" command to push changes from Local repository to Remote repository.*
- *We can use "git pull" command to pull changes from Remote repository to Local repository.*

Q3. Explain what .gitignore is and why it's important in version control

Solution:

- *Version control is a critical aspect of modern software development, allowing developers to track changes, collaborate seamlessly, and revert to previous states of their projects.*
- *Git, one of the most widely used version control systems, provides a handy tool called .gitignore to exclude certain files and directories from being tracked.*
- *Git Ignore is an important feature in Git that allows you to specify which files and directories should be excluded from being tracked in a repository.*
- *The .gitignore file is a simple text file that tells Git which files or directories to ignore in a project. It can be placed at the root of your repository or in any subdirectory.*
- *Each line in the .gitignore file specifies a pattern that matches one or more files.*
- *When Git sees a file that matches a pattern in the .gitignore file, it will not track changes to that file. This means the file will not be added to the staging area, committed or included in the version history.*
- *Purpose and Benefits of Using .gitignore:*
    - *Keeping large or unnecessary files out of the repository.*
    - *Avoiding the inclusion of sensitive information (e.g., API keys).*
    - *Excluding files that are specific to a developer's local environment.*
    - *Improving collaboration by ensuring that all team members ignore the same files reduces confusion and conflicts.*

Q4. Briefly explain what GitHub is and how it facilitates collaboration and version control also name some alternatives to GitHub.

Solution:

- ***GitHub** is a web-based platform that hosts Git repositories, providing developers with tools for version control and collaboration.*
- *Whether you are working on a small personal project or a large enterprise application, GitHub can streamline your workflow and enhance productivity.*
- *GitHub combines Git, a powerful version control system, with features that facilitate collaboration and project management.*
- *Here are some key points to understand about GitHub:*

- ***Version Control**: GitHub uses Git, a distributed version control system that tracks changes to your code. Git allows multiple developers to work on a project simultaneously without overwriting each other's changes. It keeps a history of all changes, making it easy to revert to previous versions if needed.*
- ***Collaboration**: GitHub is designed for collaborative work. It allows multiple people to contribute to a project, review code, discuss issues, and merge changes efficiently. This makes it an ideal platform for open-source projects and team-based development.*
- ***Project Management**: GitHub provides tools for managing your projects, such as issues, pull requests, and project boards. These features help you keep track of tasks, bugs, and enhancements, making it easier to manage your project's workflow.*
- *Here are the some of the GitHub alternatives:*
- ***GitLab** is a web-based DevOps lifecycle tool that provides a Git repository manager with features such as issue tracking and CI/CD pipeline.*
- ***Bitbucket** is a Git repository management solution designed for professional teams, particularly those using Atlassian products.*
- ***SourceForge** is a web-based service that offers software developers a centralized online location to control and manage free and open-source software projects.*
- ***AWS CodeCommit** is a fully managed source control service that makes it easy for teams to host secure and scalable Git repositories.*
- ***GitKraken** is a Git GUI and powerful Git client that integrates with major Git repository hosting services.*
- ***Gitea** is a lightweight, self-hosted Git service offering repository hosting and collaboration tools.*

Q5. Describe the process of contributing to any open-source project on GitHub in a step-by-step manner.

Solution:

- *First step is Finding a repository to contribute. So, go to GitHub and click on explore. Then, click on Topics. Click on the topic of your interest.*
- *Now you have a list of repositories according to your interest. Go through each and every repository and try to shortlist 2 to 3 repositories to which you are willing to contribute.*
- *Then, try to understand the codebase, guidelines to make contributions, writing and lining style and project structure.*
- *Fork the desired repository first, and then enter on create fork.*
- *Then, create a folder for the repository you want to contribute to.*
- *Open the terminal/command prompt (If you are using Windows) and type git clone for cloning the repository (git clone "the link of the repository which is a fork").*
- *Make the changes in other repositories as you want to change.*
- *Type these commands in the repository as mentioned below.*
- ***git clone {URL}:** makes a clone or copy of that repository in a new directory at another location.*
- ***cd**: To change this current working directory.*
- ***git remote add upstream**: specifies the remote repository from which we want to pull changes or updates.*
- ***git remote**: to manage the remote repositories associated with our local Git repository.*
- ***git pull upstream main**: to fetch and merge changes from the upstream repository's main branch into our local repository's current branch*

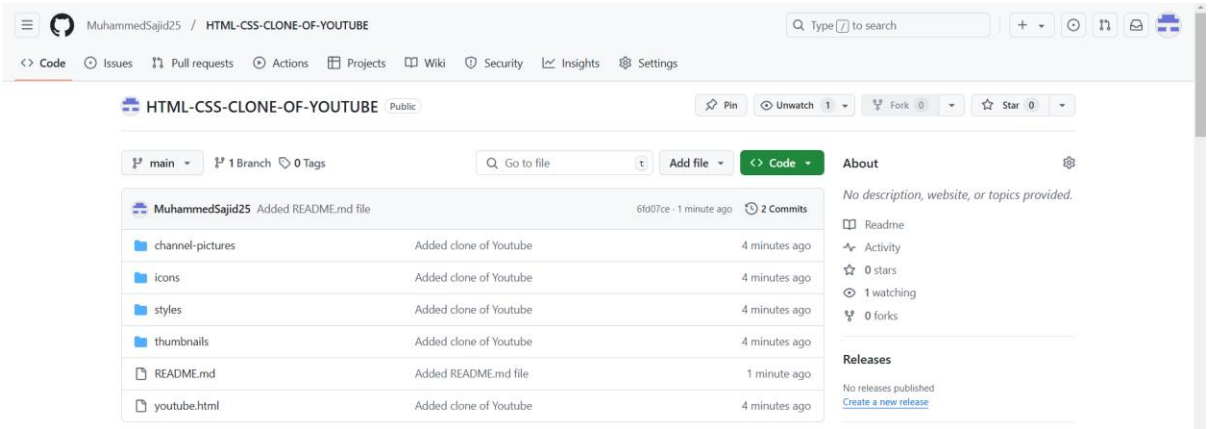
- ***git checkout -b {newbranchname}:*** *to create a new branch and switch to it at the same time*
- ***git branch -a****: to list all the branches in our local repository as well as the remote repositories that are currently available.*
- ***git add . :*** *to stage all changes in the current directory and its subdirectories for the next commit*
- ***git commit -m {message}:*** *creates a new commit with a message that describes the changes we've made*
- ***git remote -v:*** *to display a list of all the remote repositories that are currently associated with our local Git repository*
- ***git push -u origin {newbranchname****}: to push our local changes to the origin remote repository and set it as the default upstream branch for the current branch.*
- *Enter the compare & pull request.*
- *Leave a comment and then enter Create pull request, and your first pull request is generated.*
- *Now the author will check the changes you made, and if he finds them correct, he will accept your pull request; if they don't find them correct, they will suggest that you make the changes in your commit.*

Q6. Deploy Tailwind projects named Youtube, slack, and Gmail clones on GitHub pages and share the deployed link of those three. Expected output - Live hosted URL Link of your deployed respective website with GitHub pages.

***Note - For this task, you can use HTML and CSS for now. Tailwind CSS is not required at this stage.**

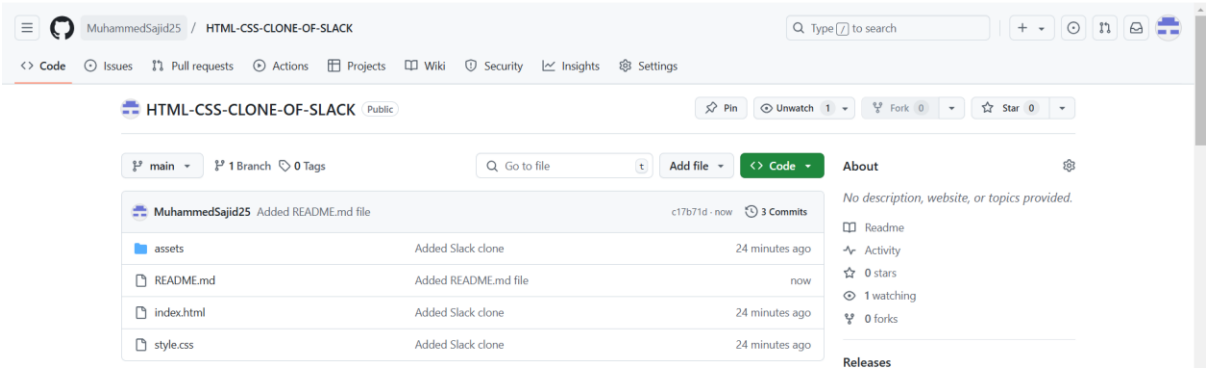
Solution:

Youtube clone:



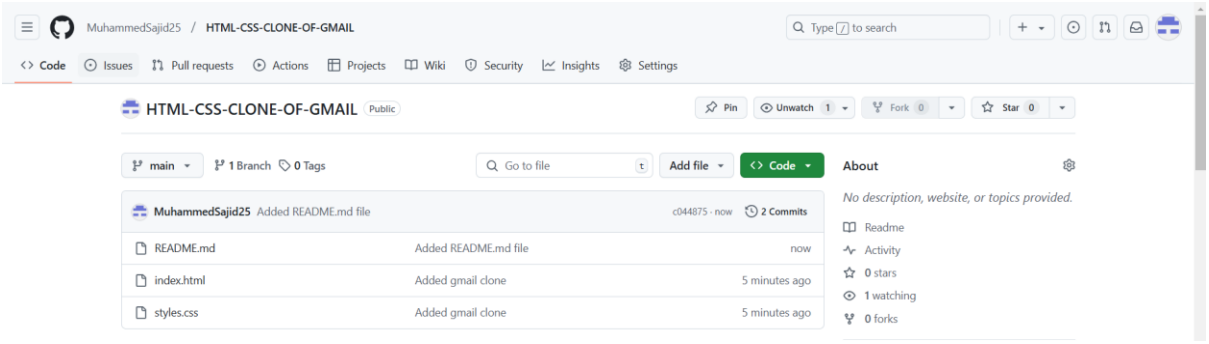
https://github.com/MuhammedSajid25/HTML-CSS-CLONE-OF-YOUTUBE.git

Slack clone:



https://github.com/MuhammedSajid25/HTML-CSS-CLONE-OF-SLACK.git

Gmail clone:



https://github.com/MuhammedSajid25/HTML-CSS-CLONE-OF-GMAIL.git