**A picture containing text, sign

Description automatically generated**

**How to use GitHub and GitHub desktop for collaboration.**

By

Tarun Narasimha Varma Muppalla

A04297061

Graduate Assistant – Research

Department of Computer Science

**Abstract**

This document explains the pivotal role of GitHub as a leading platform for collaborative software development. GitHub, built upon Git's distributed version control system, provides a centralized hub for managing code repositories, facilitating seamless collaboration among developers. The platform offers robust features such as issue tracking, pull requests, and code review, enhancing communication and coordination within development teams. GitHub's support for diverse collaboration workflows, coupled with its user-friendly interface, contributes to efficient project organization. Additionally, GitHub serves as a remote repository hosting service, enabling teams to collaborate effortlessly across geographical boundaries. This abstract provides a succinct overview of GitHub's pivotal role in streamlining collaborative workflows, ensuring code quality, and fostering effective communication in the realm of software development.

Table of Contents

1. **GitHub** 
   1. Creating a GitHub account
   2. Create a Repository
   3. Fork a Repository
   4. Creating a Branch
   5. Commit
   6. Pull Requests
   7. Merge changes
2. **GitHub Desktop**
   1. Creating a Repository
   2. Cloning a Repository
   3. Creating a Branch
   4. Create a Pull request
   5. Merge Changes
3. Conclusion
4. References

**1. GitHub**

**1.1. Creating a GitHub account**

1. Open the following link in the browser of your preferred choice: <https://github.com/>
2. Click on sign up and enter all the details it asks you for.

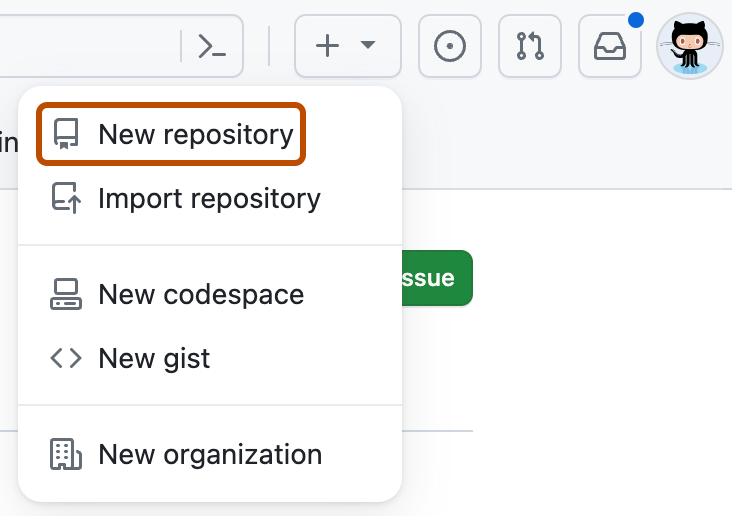
A screenshot of a computer screen

Description automatically generated

1. Next up you would have to verify your email by entering the code that comes to the email address that you had provided. Once you enter the code you have successfully created a GitHub account.

**1.2. Create and use a repository**

1. In the upper-right corner of any page, click on the plus icon, then click **New repository**.



1. Enter your desired "Repository name".
2. In the "Description" box, type a short description.
3. Select whether your repository will be **Public** or **Private**.
4. Select **Add a README file**. It’s best practice to have a README file as you can explain your project in detail within the file.
5. Add .gitignore – you can select what kind of files you want to ignore.
6. Visually once you enter everything it should look something like the screenshot below. Click **Create repository**.

A screenshot of a computer

Description automatically generated

**1.3. Fork a Repository**

You might have a question about how a team can collaborate on the same repository. This is where the concept of “Fork” comes into the picture. Let’s assume a Team lead already created the repository. His/her peers can use the URL (right-click on the repository name and copy link address) of the repository and fork it. This will create a copy on your system.

1. If the repository is a private one, the Repository owner would have to add the list of collaborators.
2. Open the Repository, and go to the **Settings** tab. Select “Collaborators”, and click on “Add people”. Add the usernames of the people you want to provide access to.

A screenshot of a computer

Description automatically generated

1. The invitation can be accepted by the teammates through the notification’s icon in the top right corner of their GitHub account.

A black and white icon

Description automatically generated with medium confidence

1. Once that’s done click on **Fork** and then **Create a Fork.**

A screenshot of a computer

Description automatically generated

Note: You would have to click on **Sync Fork** often to keep your files in sync and up to date with the repository owner (Upstream).

A screenshot of a computer

Description automatically generated

**1.4. Creating a Branch**

By default, your repository has one branch named “main”. You can create additional branches of the main in your repository. This is helpful when you want to add new features to a project without changing the main source of code.

When you create a branch of the main branch, you're making a copy, or snapshot, of the main as it was at that point in time. If someone else made changes to the main branch while you were working on your branch, you could pull in those updates.

1. Click the dropdown menu next to the **main button**.
2. Type a branch name into the text box.
3. Click **Create branch “branch name” from main**.

A screenshot of a computer

Description automatically generated

Now you have two branches, main and feature1. Right now, they look the same. Next, you'll add changes to the new branch.

**1.5. Making and Committing changes**

You can make and save changes to the files in your repository. On GitHub, saved changes are called **commits**. Each commit has an associated commit message, which is a description explaining why a particular change was made. Commit messages capture the history of your changes so that other contributors can understand what you’ve done and why.

1. Under the branch you created, let’s say you made changes to an existing file or added a new file.
2. You can make changes by clicking on the **Pen icon** as shown in the screenshot below.

A screenshot of a computer

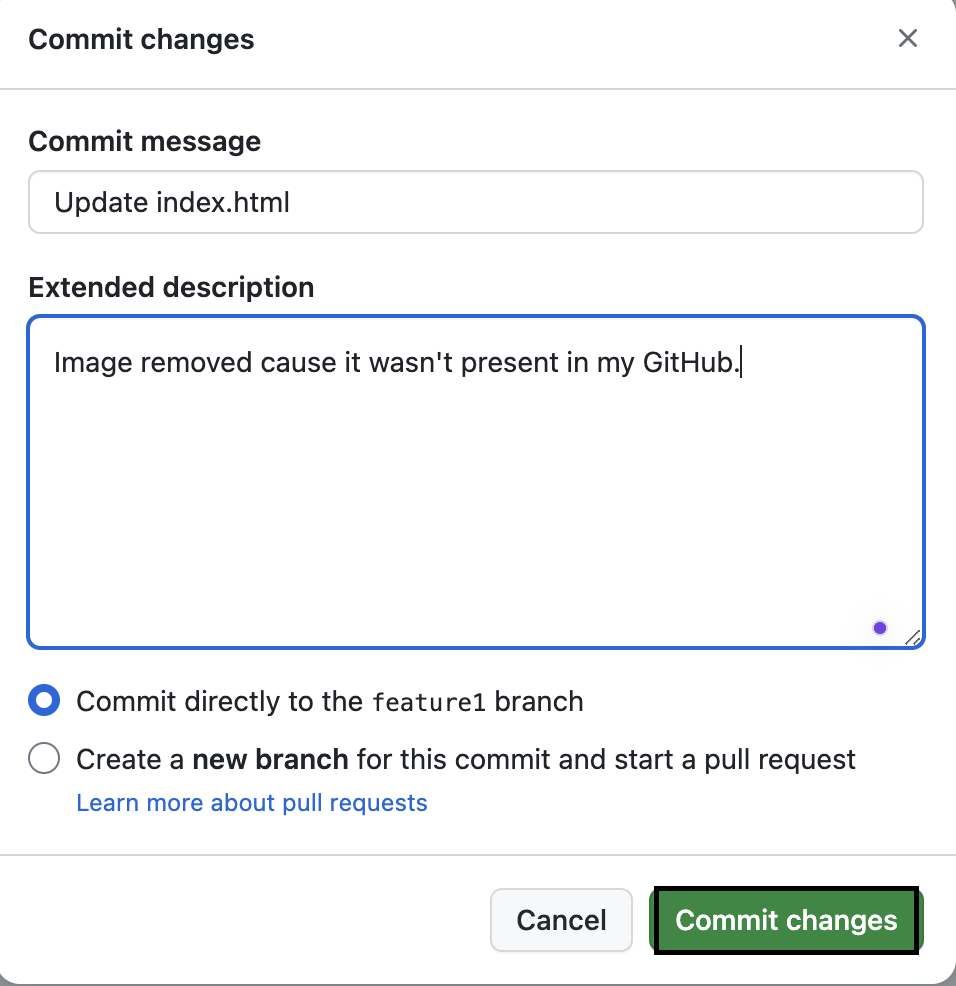
Description automatically generated

1. Click **Commit changes...**.

A screenshot of a computer

Description automatically generated

1. In the "Commit changes" box, write a commit message that describes your changes.



1. Click **Commit changes**.

**1.6. Open a Pull Request**

Now that you have changes in a branch off of the main, you can open a pull request.

Pull requests are the heart of collaboration on GitHub. When you open a pull request, you're proposing your changes and requesting that someone review and pull in your contribution and merge them into their branch. Pull requests show diffs, or differences, of the content from both branches. The changes, additions, and subtractions are shown in different colors.

As soon as you make a commit, you can open a pull request and start a discussion, even before the code is finished. You can even open pull requests in your own repository and merge them yourself. It's a great way to learn the GitHub flow before working on larger projects.

1. Click the **Pull requests** tab of your repository.
2. Click **New pull request**

A screenshot of a computer

Description automatically generated

1. In the **Example Comparisons** box, select the branch you made, in this case it would be feature1 to compare with main (the original).
2. Look over your changes in the diffs on the Compare page, make sure they're what you want to submit.

A screenshot of a computer

Description automatically generated

1. Give your pull request a title and write a brief description of your changes. You can include emojis, drag-and-drop images, and gifs. Click **Create pull request**.

A screenshot of a computer

Description automatically generated

1. Optionally, to the right of your title and description, click the next to **Reviewers**, **Assignees**, **Labels**, **Projects**, or **Milestone** to add any of these options to your pull request. You do not need to add any yet, but these options offer different ways to collaborate using pull requests. For more information, see "[About pull requests](https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/about-pull-requests)."
2. Click **Create pull request**.

**1.7. Merging a Pull request**

In this final step, you will merge your feature1 branch into the main branch. After you merge your pull request the changes will reflect in the main branch.

Sometimes, a pull request may introduce changes to code that conflict with the existing code on main. If there are any conflicts, GitHub will alert you about the conflicting code and prevent merging until the conflicts are resolved. You can make a commit that resolves the conflicts or use comments in the pull request to discuss the conflicts with your team members.

1. Under Pull request tab open the respective pull you want to merge. A screenshot of a computer

   Description automatically generated
2. click **Merge pull request** to merge the changes into main.

A screenshot of a computer

Description automatically generated

1. Click **Confirm merge**. You will receive a message that the request was successfully merged, and the request was closed.
2. Click **Delete branch** if you want to. Now that your pull request is merged and your changes are on main, you can safely delete the feature1 branch. If you want to make more changes to your project, you can always create a new branch and repeat this process.

**2. GitHub desktop**

Use the following link to download and install GitHub desktop: <https://desktop.github.com/>

If you wish to use Git on command line and you want to install it on your VS code then refer to the following tutorial at Timestamp: 7:14: <https://www.youtube.com/watch?v=8Dd7KRpKeaE>

Why do we need it?

* Promotes Collaboration: When a team is trying to collab to develop a code it’s better to use GitHub Desktop app. Using the GitHub desktop app will help ease the process of making changes to your code and repositories.
* Ease of use: You can also revert changes or go back to a previous version by looking under the history tab.
* Use of Editors: You can use Visual Studio Code, which will make implementing your code easier.

**2.1. Creating a Repository**

You can create a new repository on the desktop app by using “Create New Repository” but here we would be using the existing repository.

A screenshot of a computer

Description automatically generated

* 1. **Cloning a repository**

1. If we must work on an existing repository, then we would have to clone the repository by clicking on “Clone Repository”. This will set you up with a copy of all the files on your local system.

A screenshot of a computer

Description automatically generated

1. Enter the URL of the existing repository and click on **Clone**.

A screenshot of a computer

Description automatically generated

1. Now you can open the existing files in Visual Studio. Whatever changes made to the code will automatically reflect on GitHub desktop, unlike the web application where we would have to manually override the code for it to detect changes.
2. Select the file that you would like to commit to the branch, add a description of the change and click on “Commit to main” or any other branch of your choice.

A screenshot of a computer

Description automatically generated

1. By clicking on **Push to origin**, you have published the changes from your local system to the main branch.

A screenshot of a computer

Description automatically generated

**2.3. Create a new Branch on GitHub desktop.**

1. Navigate to Branch, under it select option “New Branch”

A screenshot of a computer

Description automatically generated

1. Let’s say in visual code you added a new file under the new branch you just created, in this case, “featuredemo”. Then you commit this new file into that branch.

A screen shot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

1. To merge the change into main branch. Go to the main branch first and then click on branch and then select “Merge into Current Branch”

A screenshot of a computer

Description automatically generated

1. Select the branch from which you the files to be merged with main. Click on “Create a Merge Commit”

A screenshot of a computer

Description automatically generated

Note: It’s not always safe to directly merge changes into the main branch. As a best practice, we have something called Pull request in GitHub when we are collaborating so the other team members can review your code and suggest changes or validate your code before committing.

**2.4. Create a Pull request**

1. Commit the changes to the branch.

A screenshot of a computer

Description automatically generated

1. Click on “Push origin”
2. Go to GitHub on your web browser click on “New Pull Request”. Select the base branch and the branch you want to compare it with. Click on Create Pull Request.

A screenshot of a computer

Description automatically generated

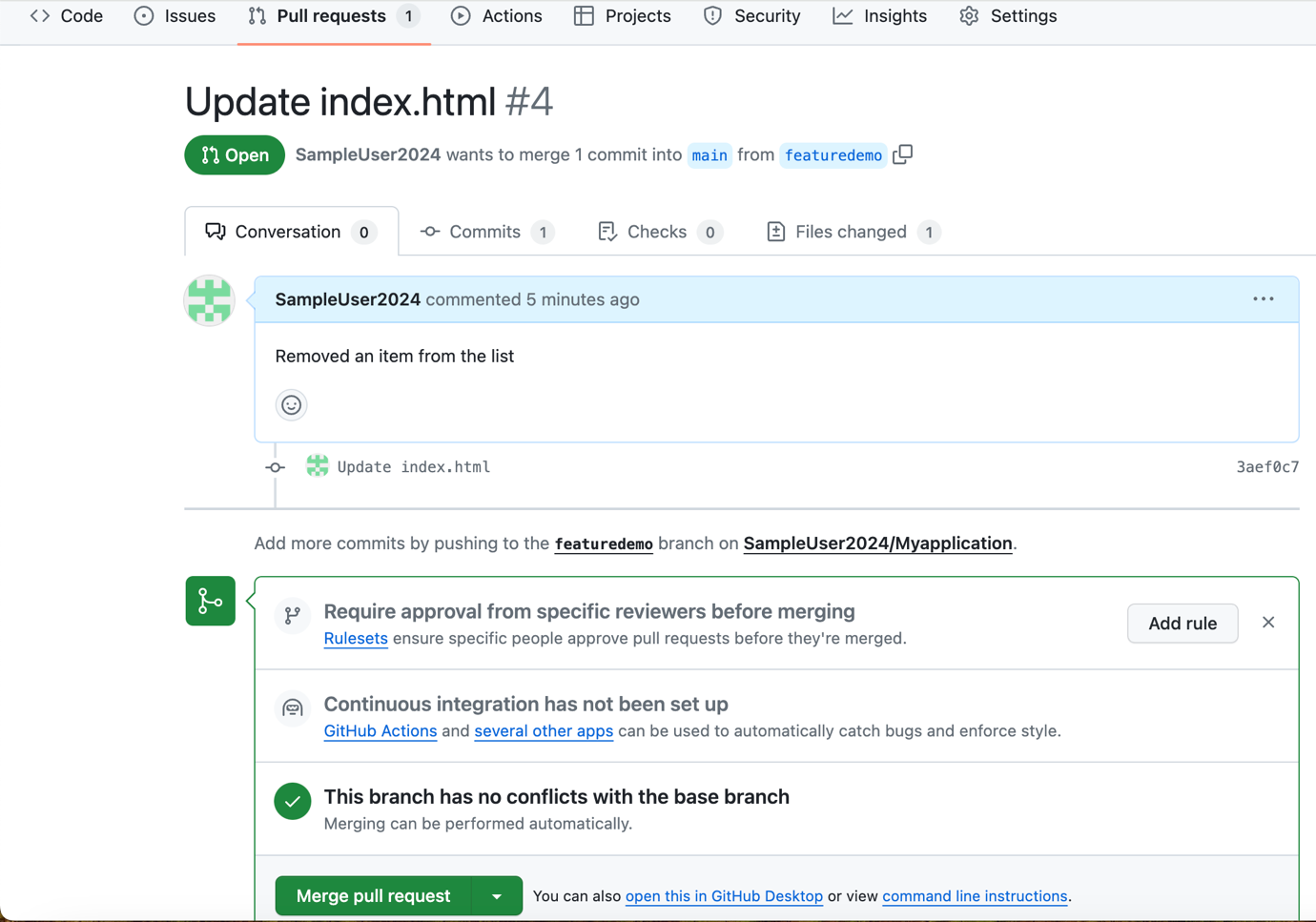
1. Add a description to your Pull Request and then click on Create pull request.

A screenshot of a email

Description automatically generated

**2.5. Merge changes**

1. Under Pull requests tab, select the request. Verify the changes and click on Merge Pull request.



1. If you don’t approve of the changes made, close the pull request by adding the necessary comments.

A screenshot of a computer

Description automatically generated

**Conclusion**

This document serves as a comprehensive guide on how to prepare and leverage GitHub for collaborative software development. By following the outlined steps and best practices, users can establish a strong foundation for effective version control, streamlined collaboration, and efficient project management. The document has covered essential aspects, including setting up repositories, branching strategies, pull requests, code reviews. Emphasizing the importance of clear communication and adherence to collaborative workflows, this guide aims to empower users in maximizing the potential of GitHub. As users embark on their GitHub journey, incorporating these insights will not only enhance their individual proficiency but also contribute to the overall success of collaborative software development endeavors.

**Table of References:**

|  |  |
| --- | --- |
| **Url Description** | **Url** |
| GitHub Link | <https://github.com/> |
| GitHub Desktop App link | <https://desktop.github.com/> |
| Visual Tutorial of How to set up and use GitHub Desktop | <https://www.youtube.com/watch?v=8Dd7KRpKeaE> |
| To better understand Pull request and merge visually | <https://www.youtube.com/watch?v=k5D37W6h56o> |
|  |  |
|  |  |
|  |  |