Name - Dhruv D Vaidya
Batch - T23
Roll no - 117

Experiment 2 : Github Account

Aim :

To study and implement making a GIT account.

Theory :

1. Version Control System (VCS)

A **Version Control System (VCS)** is a software tool that helps manage changes to source code or any set of files over time. It allows multiple people to work on the same project without interfering with each other's changes, maintaining a history of modifications, and enabling easy collaboration.

Key Concepts in Version Control:

- **Versioning:** Keeping track of changes made to files. Each change is recorded as a version, allowing you to revert to previous states if necessary.
- **Repository (Repo):** A storage location for your project, including all the files and the history of changes made to them. Repositories can be local (on your computer) or remote (on a server like GitHub).
- **Commit:** A snapshot of changes saved to the repository. Each commit has a unique identifier (hash) that allows you to track specific changes.
- **Branching:** Creating a separate line of development from the main codebase. This is useful for developing new features, fixing bugs, or experimenting without affecting the stable code.
- **Merging:** Integrating changes from one branch into another, typically from a feature branch back into the main branch.

Types of Version Control Systems:

1. **Centralized Version Control System (CVCS):**

- Example: Subversion (SVN)
- A single central repository is used, and developers pull the latest version to work on. Changes are committed directly to the central repository.

2. **Distributed Version Control System (DVCS):**
   - Example: Git, Mercurial
     Every developer has a full copy of the repository, including its history.
   - Changes can be made locally and pushed to a shared repository when ready.

---

2. Git

Git is a distributed version control system created by Linus Torvalds in 2005 to manage the Linux kernel's source code. It is designed to handle projects of all sizes with speed, efficiency, and flexibility.

Core Concepts in Git:

- **Repository (Repo):** The main directory that contains your project files and a hidden .git folder that tracks all changes, commits, branches, etc.
- **Working Directory:** The current state of your files. Changes you make here are not tracked until you add and commit them.
- **Staging Area (Index):** A place where you prepare changes before committing them. It allows you to control which changes get included in the next commit.
- **Commit:** A snapshot of the changes added to the staging area. Each commit has a unique SHA-1 hash and a commit message describing the changes.
- **Branches:** Separate lines of development. The default branch is usually main or master, but you can create branches for new features or bug fixes.
- **Merge:** The process of combining changes from one branch into another. This can lead to merge conflicts if the same parts of the files were modified differently.
-

- **Remote Repositories:** Copies of your repository hosted on a server (like GitHub, GitLab, Bitbucket) that allow for collaboration and backup.

Basic Git Workflow:

Clone a Repository:

```
git clone <repository_url>
```

1. This copies the remote repository to your local machine.

Check Repository Status:

```
git status
```

2. This shows which files have been modified, staged, or are untracked.

Add Changes to Staging Area:

```
git add <file_name>
```

Or add all changes:

```
git add .
```

Commit Changes:

```
git commit -m "Your commit message"
```

Push Changes to Remote Repository:

```
git push origin <branch_name>
```

Pull Changes from Remote Repository:

```
git pull origin <branch_name>
```

---

3. GitHub

GitHub is a web-based platform that uses Git for version control and provides a collaborative environment for developers. It hosts your code repositories online, allowing others to contribute, review, and manage code.

Key Features of GitHub:

- **Repositories:** Host and manage your code projects in the cloud.
- **Forks:** Create your own copy of someone else's repository to experiment with changes without affecting the original.
- **Pull Requests (PRs):** Propose changes to a repository. Other collaborators can review your changes before they are merged into the main project.
- **Issues:** Track bugs, tasks, and feature requests within a project.
- **Actions:** Automate workflows like testing, building, and deploying code with GitHub Actions.
- **GitHub Pages:** Host static websites directly from your GitHub repository.

---

4. Creating a GitHub Account

Step-by-Step Process:

1. **Visit GitHub:**

   Go to https://github.com.

2. **Sign Up:**

   - Click on the **"Sign up"** button.
   - Enter your desired **username**, **email address**, and **password**.
   - Choose your plan (there's a free plan available for individual developers and open-source projects).

3. **Verify Email:**
   GitHub will send a confirmation email. Open your inbox and click the verification link.

4. **Set Up Git:**

   - Install Git on your computer: Download Git.

Configure Git with your GitHub credentials:

```
git config --global user.name "Your Name"
git config --global user.email "your_email@example.com"
```

◯

5. **Create a New Repository:**

   ◯ Click the **"+"** button at the top-right corner → **New repository**.

   ◯ Choose a name, description, and set the repository to public or private.

   ◯ Optionally, initialize it with a README, .gitignore, or license file.

---

5. Example Git Workflow with GitHub

Scenario: You want to add a new feature to a project.

Clone the Repository:

```
git clone https://github.com/username/repository.git
```

Create a New Branch:

```
git checkout -b feature/new-feature
```

1. **Make Changes:**
   Edit files, add new code, or fix bugs.

Stage Changes:

```
git add .
```

Commit Changes:

```
git commit -m "Add new feature"
```

Push Changes to GitHub:

```
git push origin feature/new-feature
```

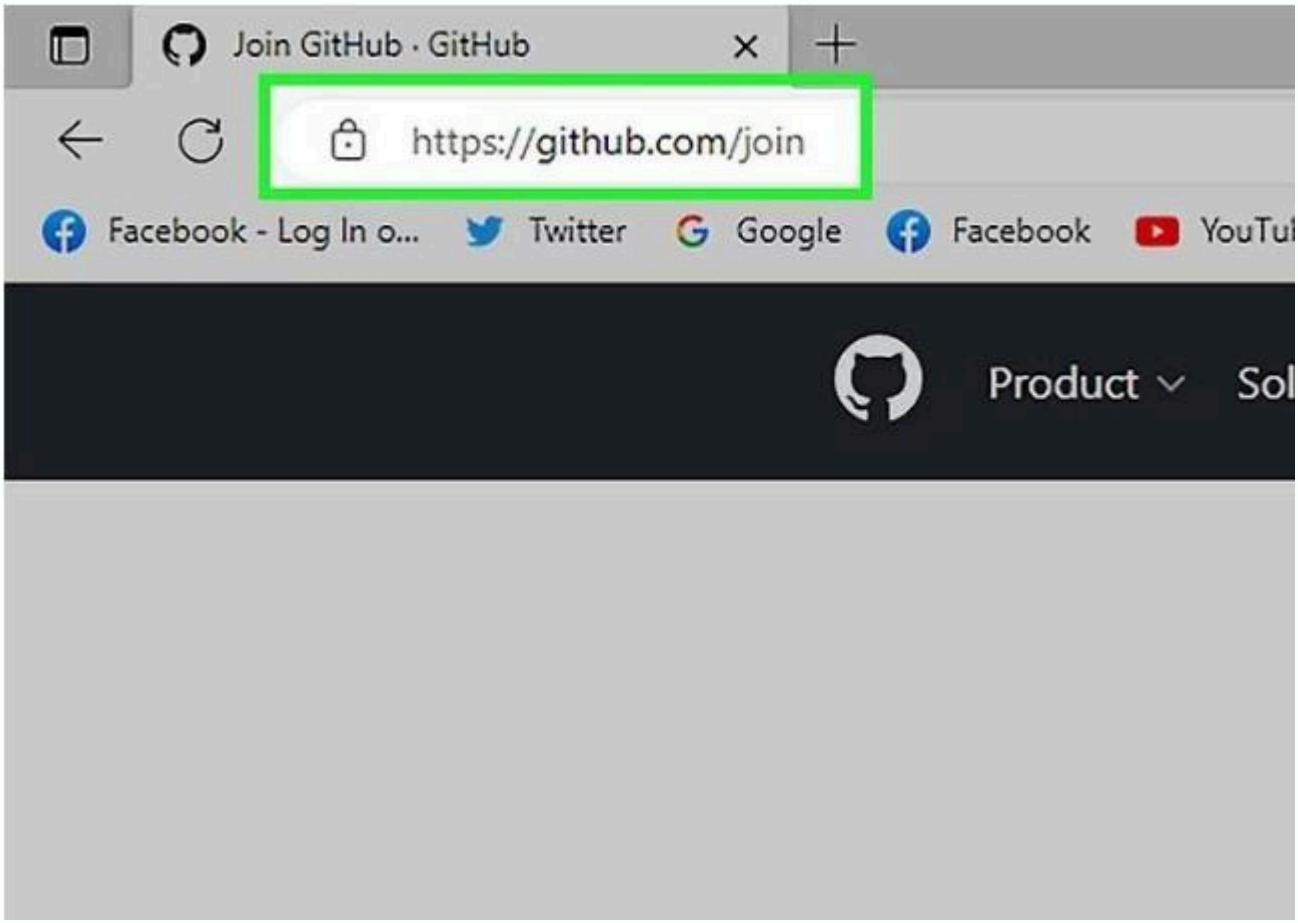
2. **Create a Pull Request:**
    - ◯ Go to the repository on GitHub.
    - ◯ Click on **"Compare & pull request"**.
    - ◯ Add a description of your changes and submit the pull request.
3. **Review and Merge:**
   Collaborators review the pull request, suggest changes, and eventually merge it into the main branch.

---

6. Advanced Git Concepts

- **Rebase:** A way to move or combine a sequence of commits to a new base commit.
- **Cherry-Pick:** Apply the changes from a specific commit to another branch.
- **Stash:** Temporarily save changes that you're not ready to commit.
- **Tags:** Mark specific points in history as important (e.g., releases).

Screenshots:



Join GitHub

First, let's create your user account

Username *

wikihowneveconcepts

Email address *

Password *

••••••••••

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. Learn more.

Email preferences

Send me occasional product updates, announcements, and offers.

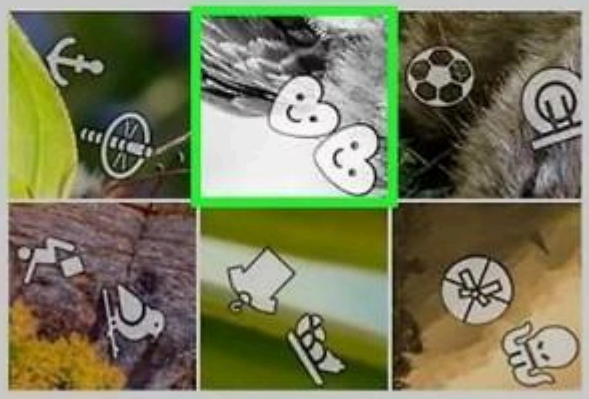
Verify your account

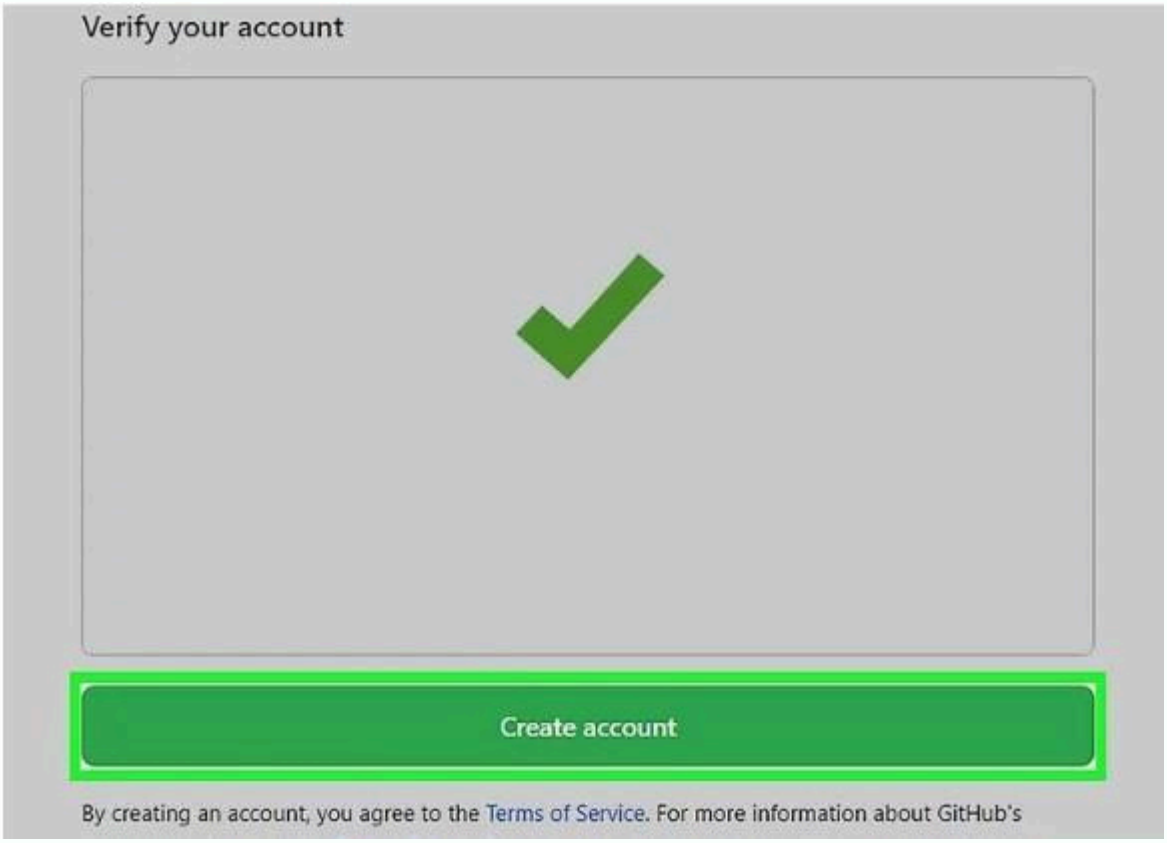
Email preferences

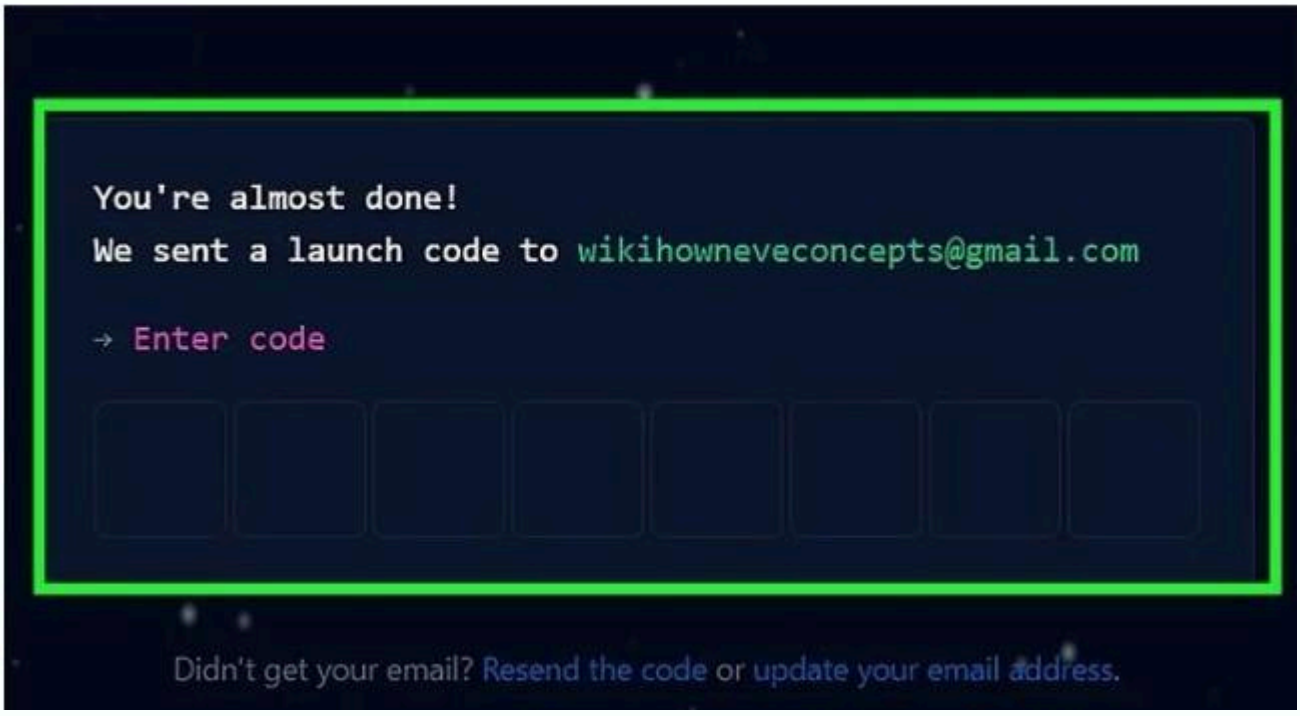
Send me occasional product updates, announcements, and offers.

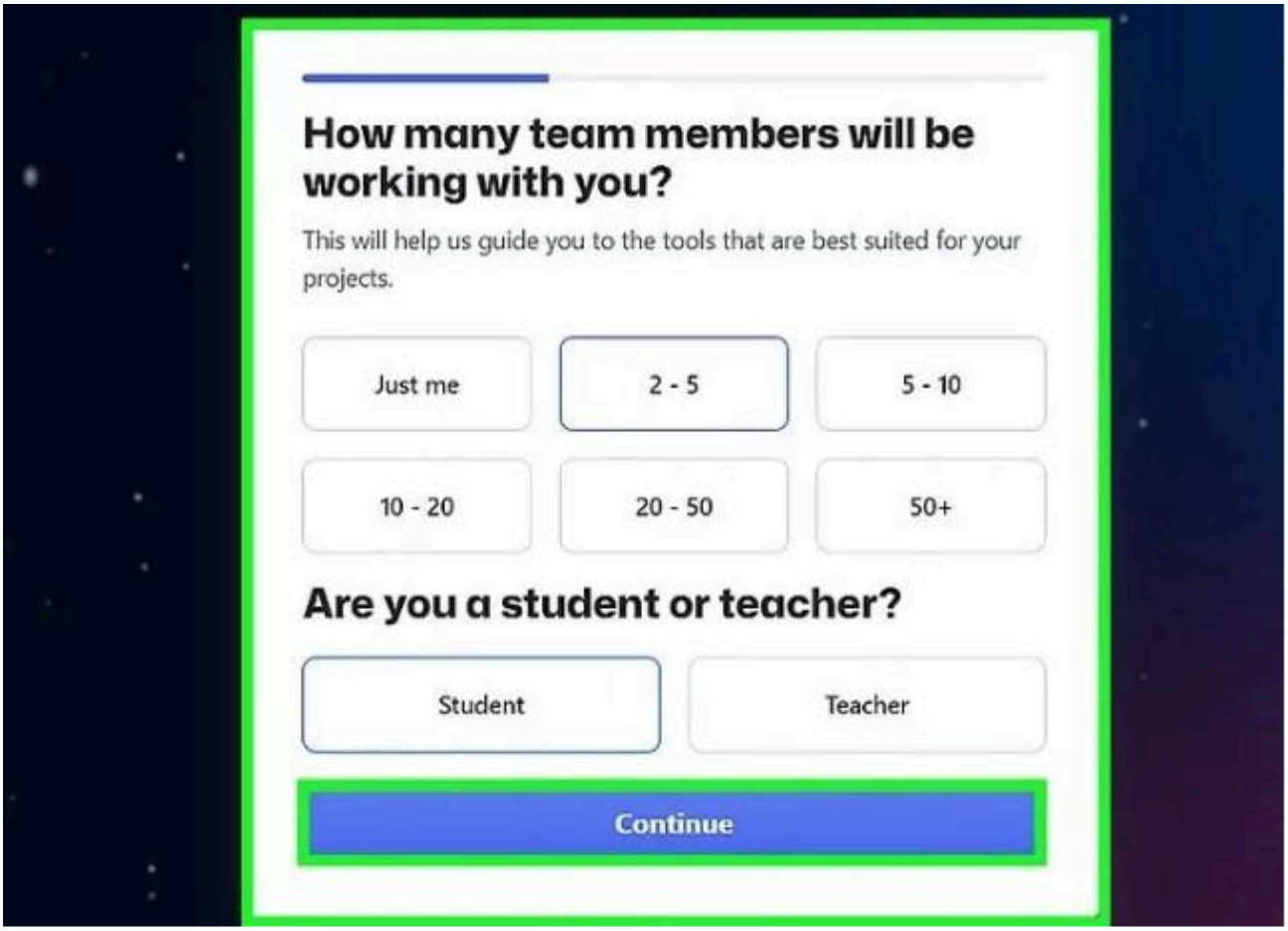
Verify your account

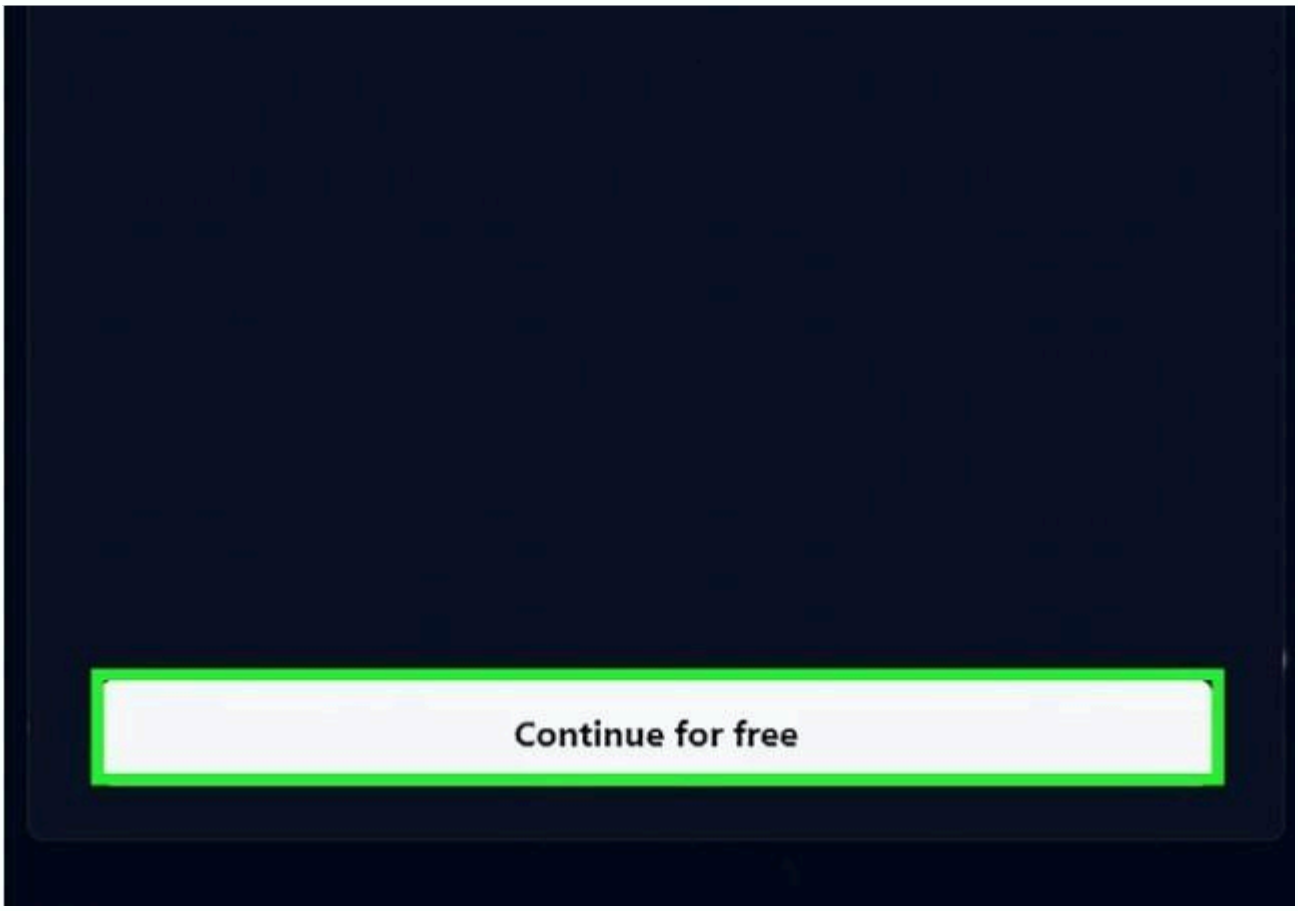
Pick one square that shows two identical objects.




Verify your account
Create account
By creating an account, you agree to the Terms of Service. For more information about GitHub's


You're almost done!
We sent a launch code to wikihowneveconcepts@gmail.com
→ Enter code
Didn't get your email? Resend the code or update your email address.


How many team members will be working with you?
This will help us guide you to the tools that are best suited for your projects.
Just me
2 - 5
5 - 10
10 - 20
20 - 50
50+
Are you a student or teacher?
Student
Teacher
Continue


Continue for free

Conclusion :

Hence we studied and implemented creating a Github account.