DIGITAL ASSIGNMENT – 1

SHREYA CHATTERJEE

18BIT0251

SLOT: **B1+TB1**

COURSE CODE: **ITE1008**

COURSE TITLE: **OPEN SOURCE PROGRAMMING**

TOPIC: **GITHUB**

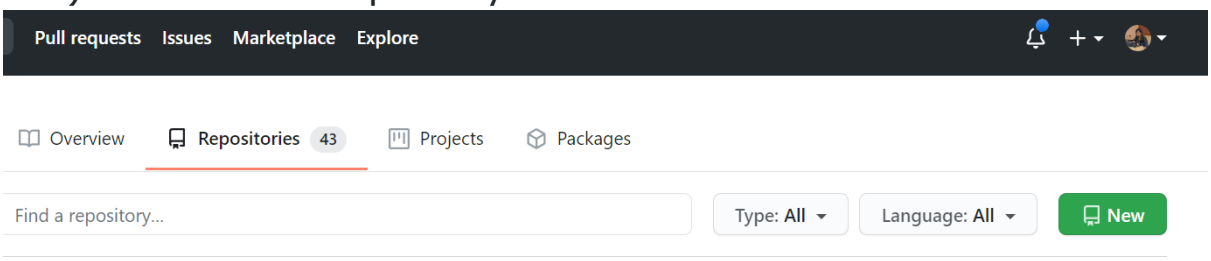
Git is a distributed version control system. GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere.

GitHub can be accessed in the following ways:

1) **GitHub platform**
2) **Git Command Line Interface**
3) **GitHub Desktop**
4) **GitHub Command Line Interface**

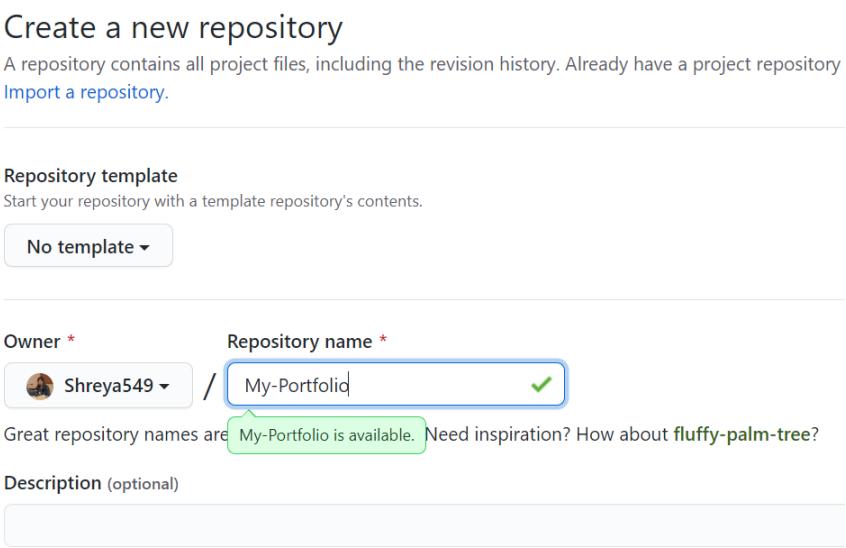
GitHub Platform

1) Create a new Repository

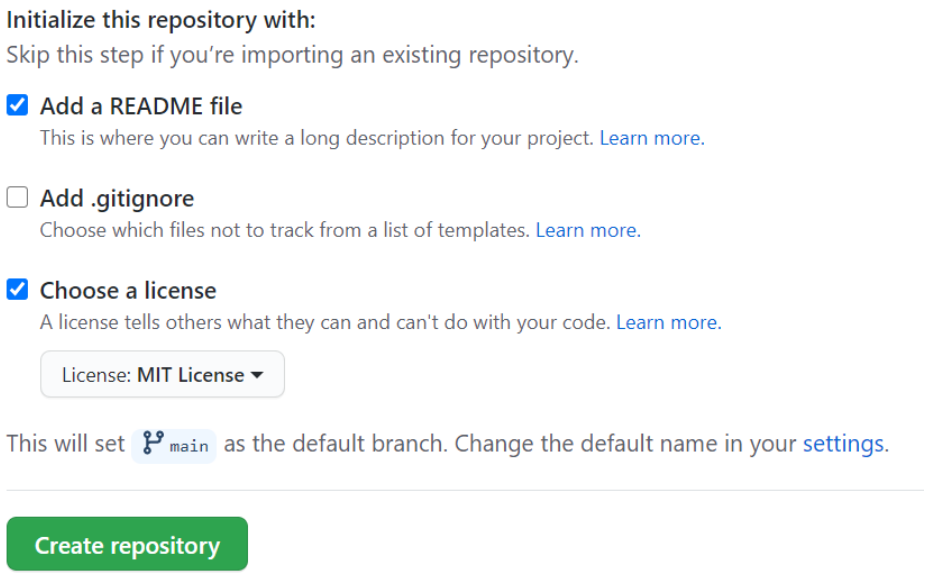


1. Click on New

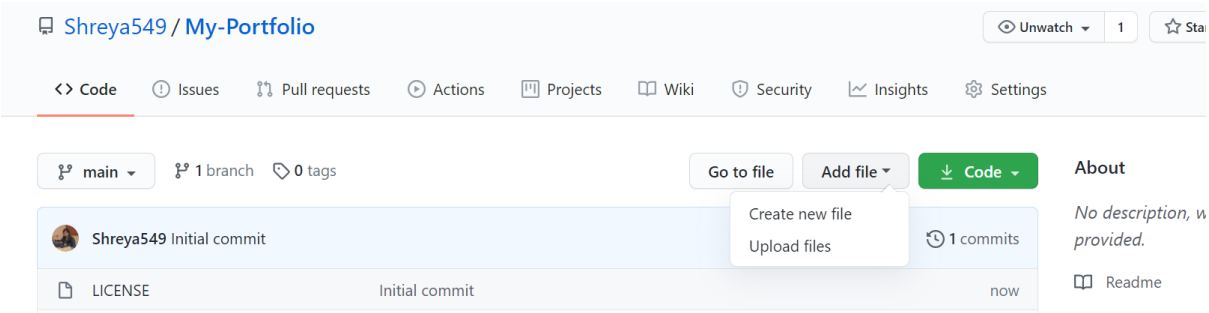
2) Set a name to your repository



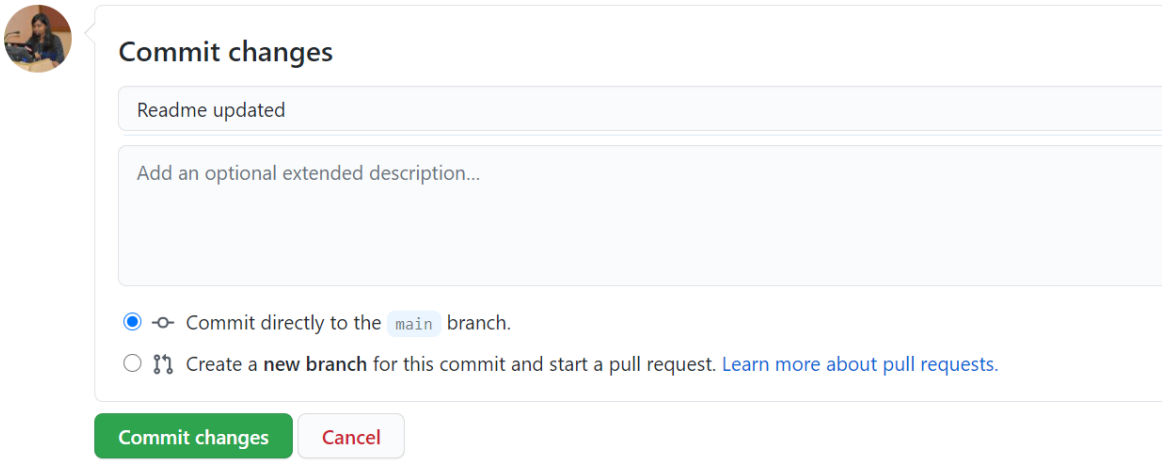
3) Add a README and choose a license



4) Add code to your repository

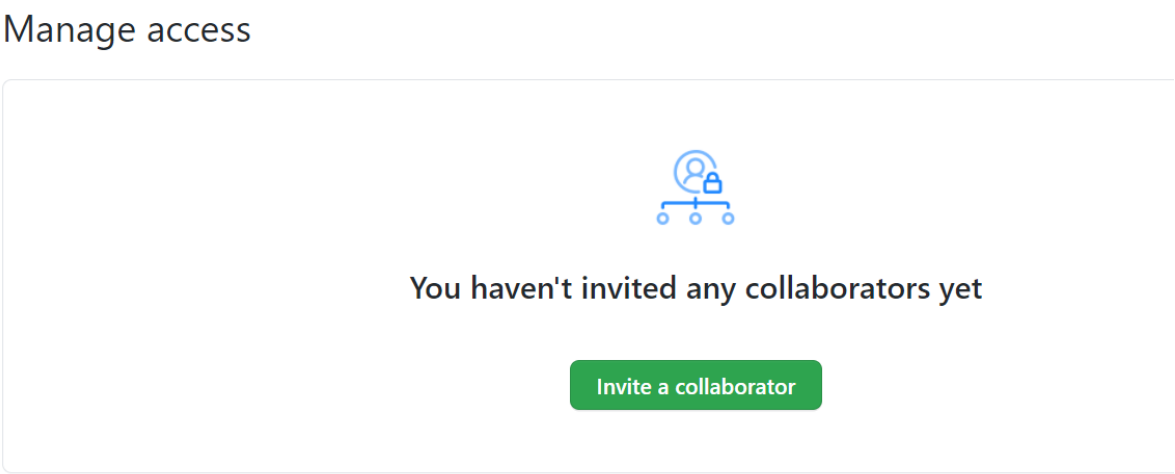


5) Add a commit message

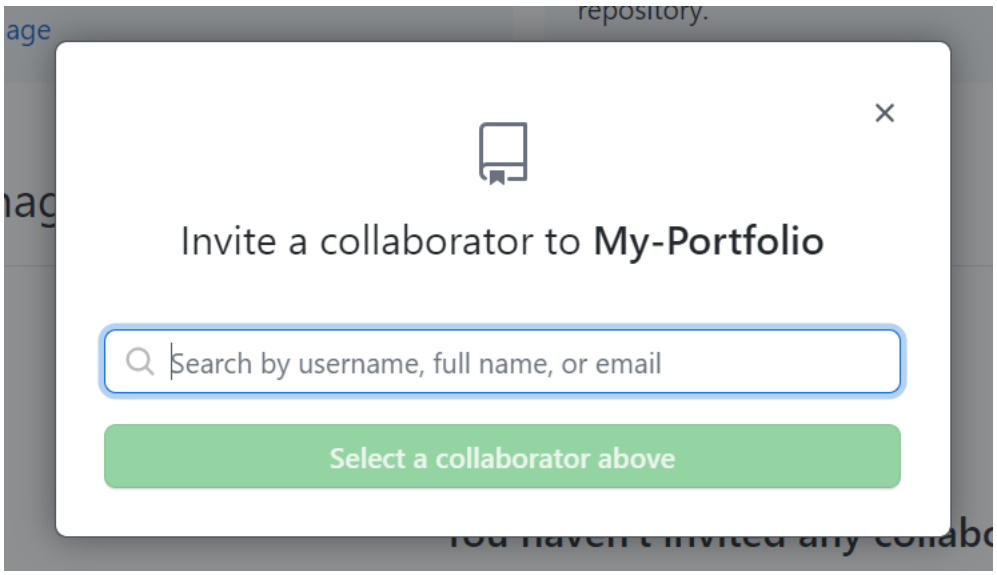


Since GitHub is used for version control, we can add collaborators to our projects as well.

1) Go to Settings >> Manage Access



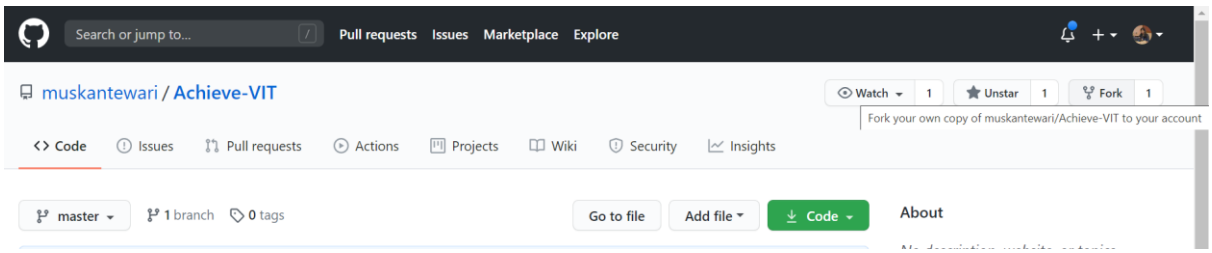
2) Send invitation to a collaborator



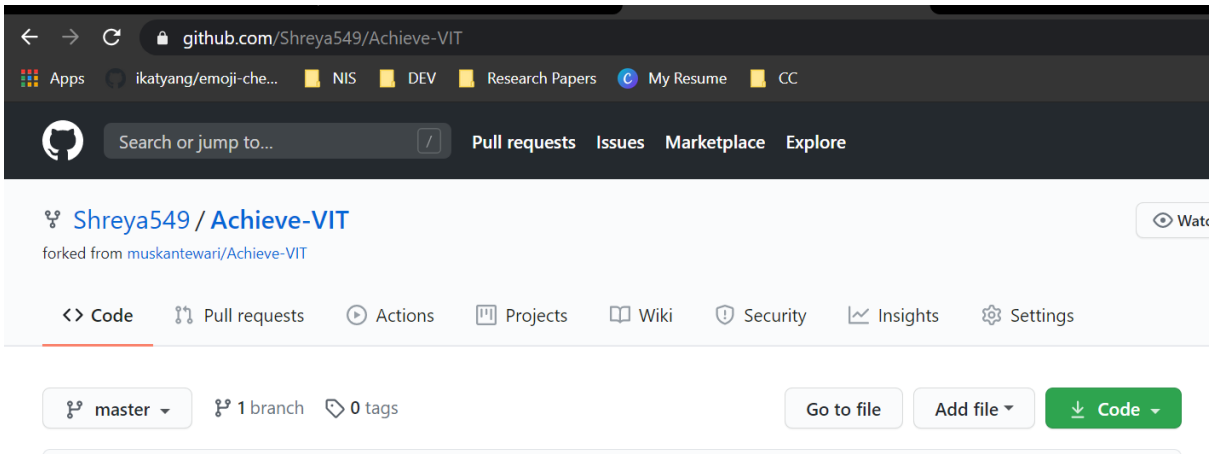
A major advantage of GitHub is the Open Source Community

To make Open Source Contributions on a repository

1) Go to someone else's repository and click on fork



2) GitHub will create a copy of this repository in your own account

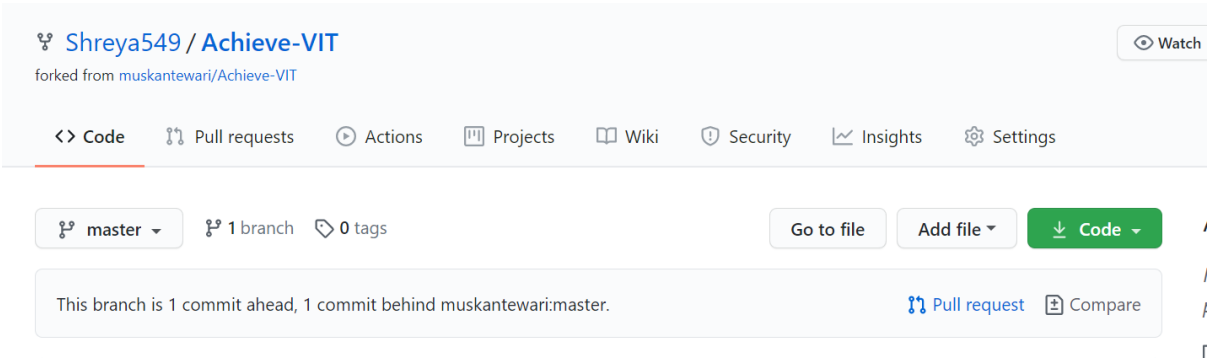


3) Make changes to this code

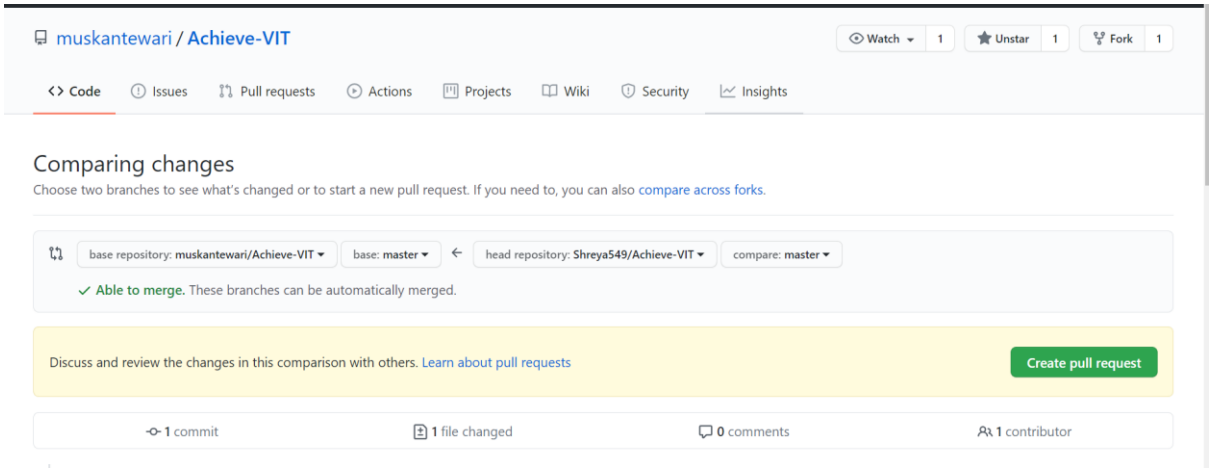
4) Commit the changes

5) Send your code for review to the original Repository owner, by sending a Pull Request

Click on Pull Request



6) Check for merge Conflicts



7) Check additions and deletions

Showing 1 changed file with 2 additions and 2 deletions.

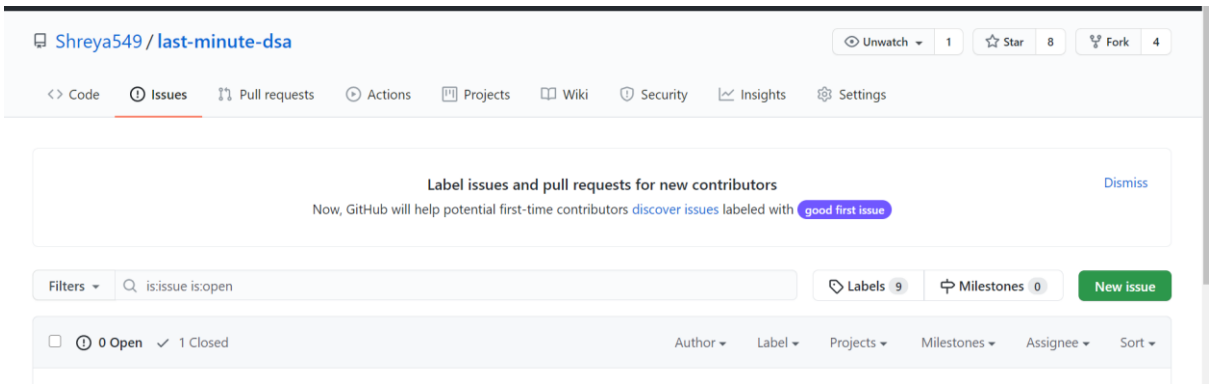
4 js/faculty_login.js

```
@@ -1,5 +1,4 @@
 function login(){
-    console.log('Hii')
     // debugger;
     var data = {
         "empid" : document.getElementById('username').value,
@@ -9,8 +8,9 @@ function login(){
     var xh = new XMLHttpRequest();
     xh.open("POST", "https://achieve-vit.herokuapp.com/accounts/login/", true);
     xh.setRequestHeader('Content-Type', 'application/json');
+    xhr.setRequestHeader('Access-Control-Allow-Origin', '*');
     xh.send(JSON.stringify(data));
-    // xhr.setRequestHeader('Access-Control-Allow-Origin', '*');
+
     xh.onload = function () {
         console.log("HIii")
         console.log(this.responseText)
```

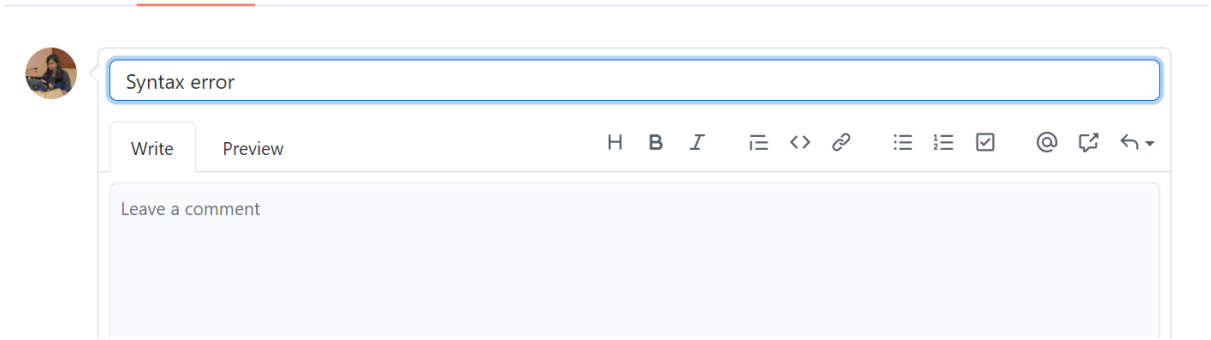
8) Click on Create Pull Request

Since the projects are open sourced, users can raise issues on any repository

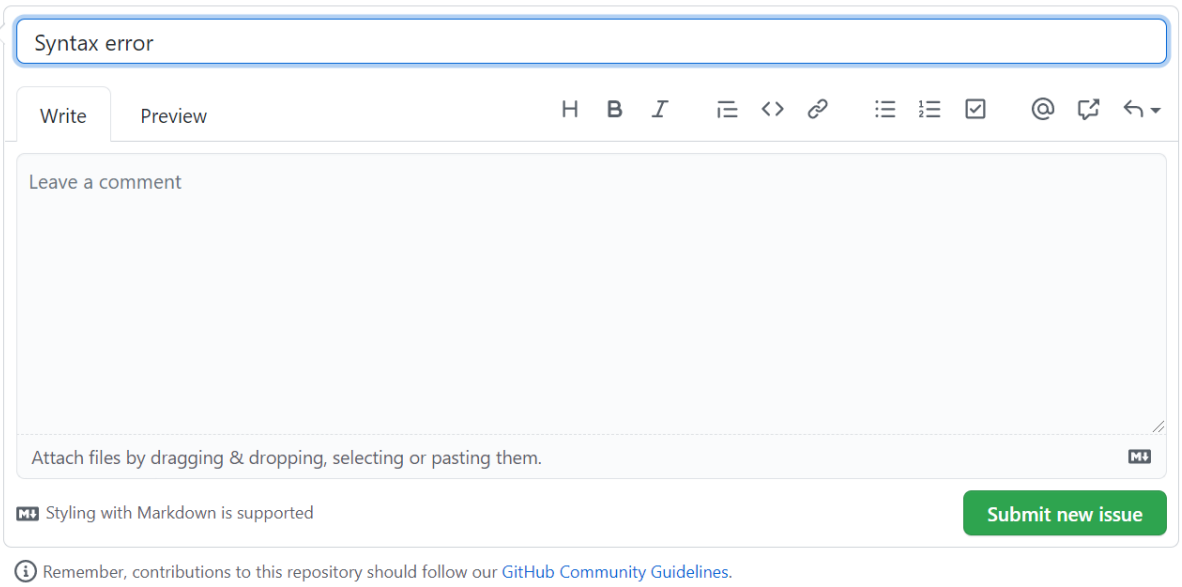
1) Go to Issues >> New Issue



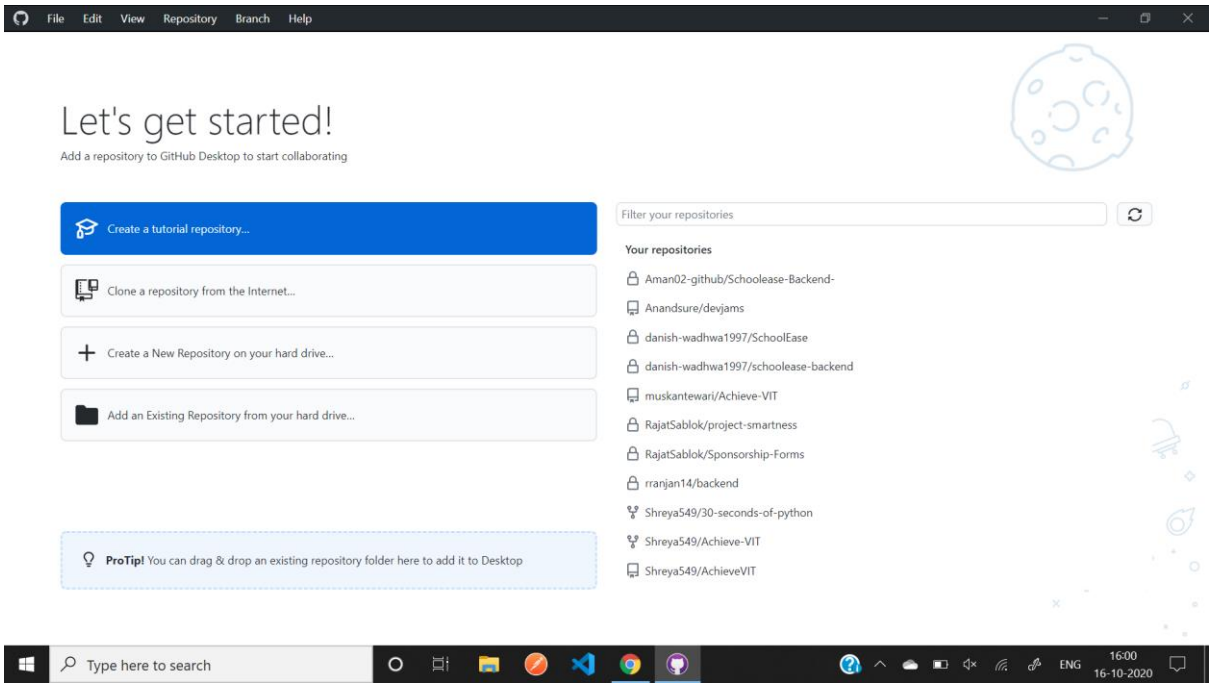
2) Add a title to the issue



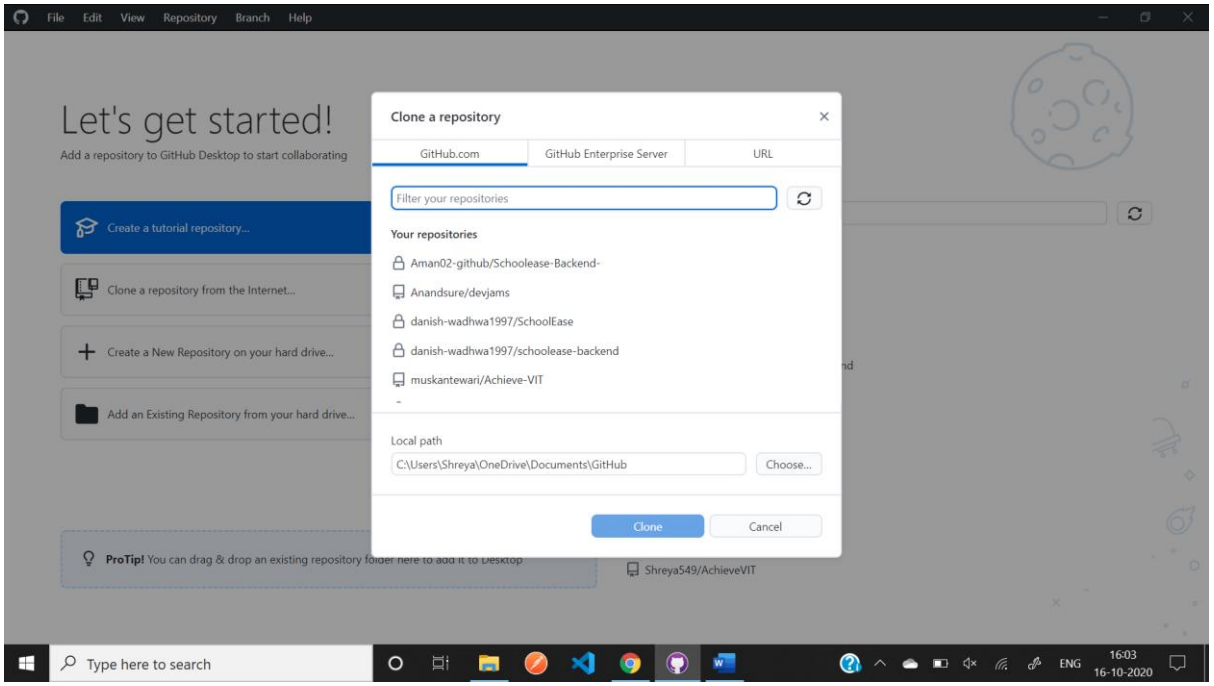
3) Submit Issue



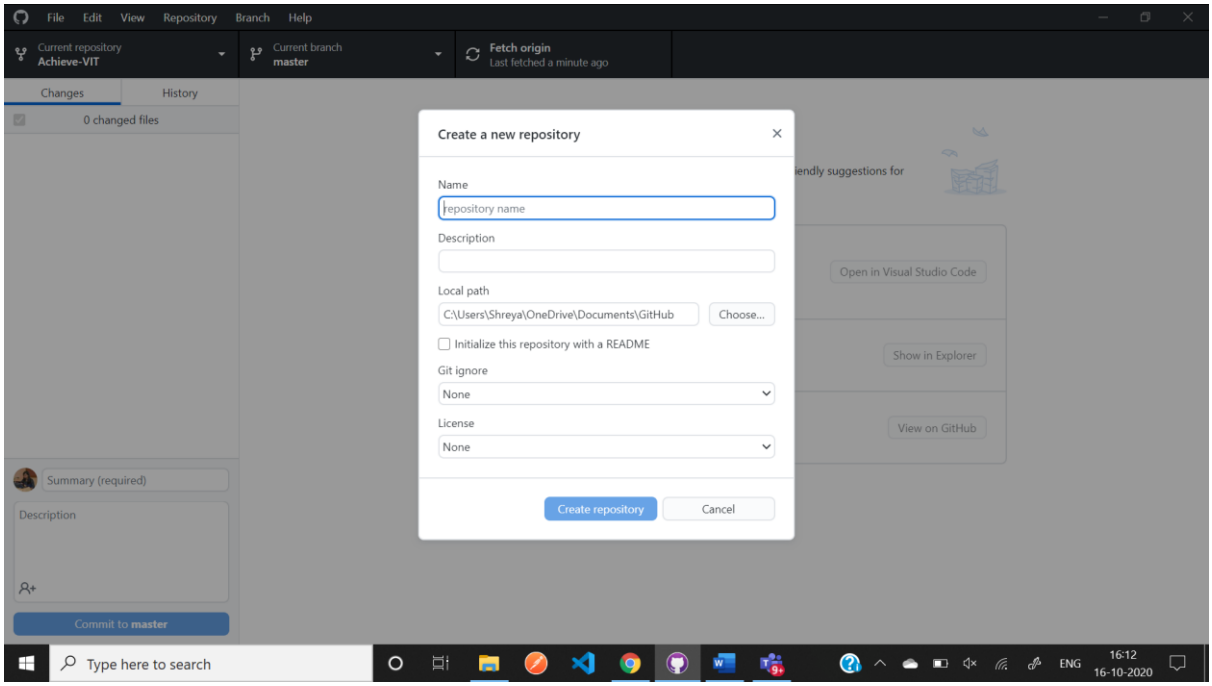
GitHub Desktop



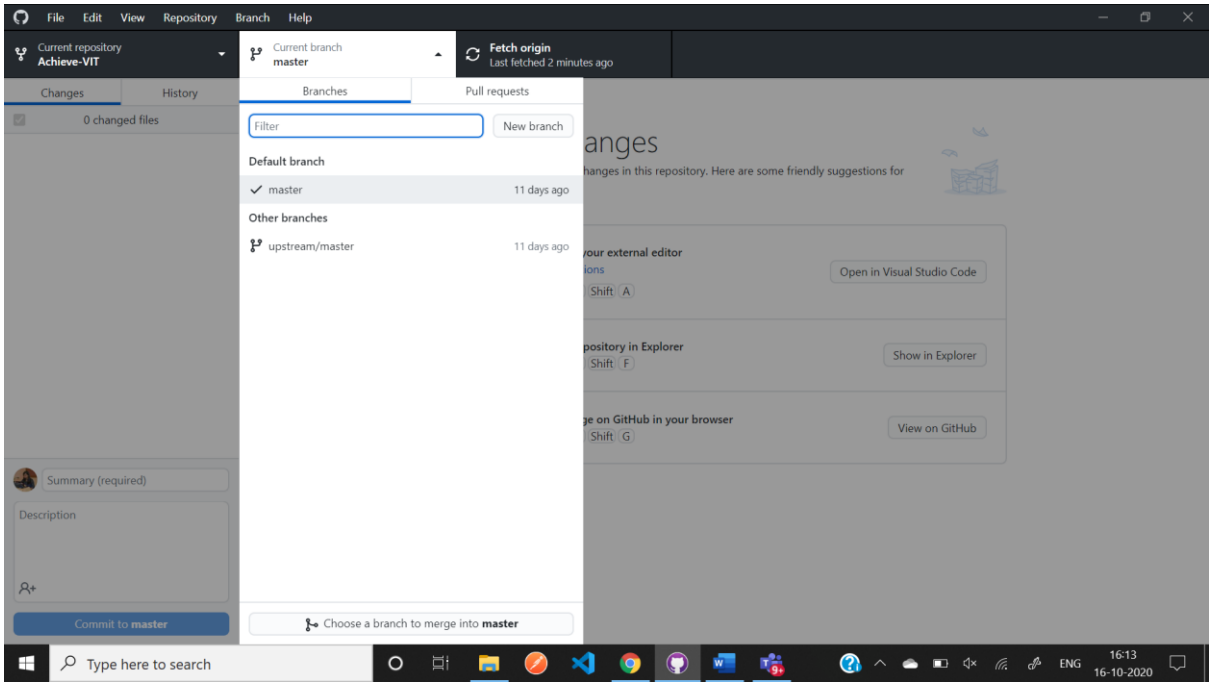
1) Clone an existing repository



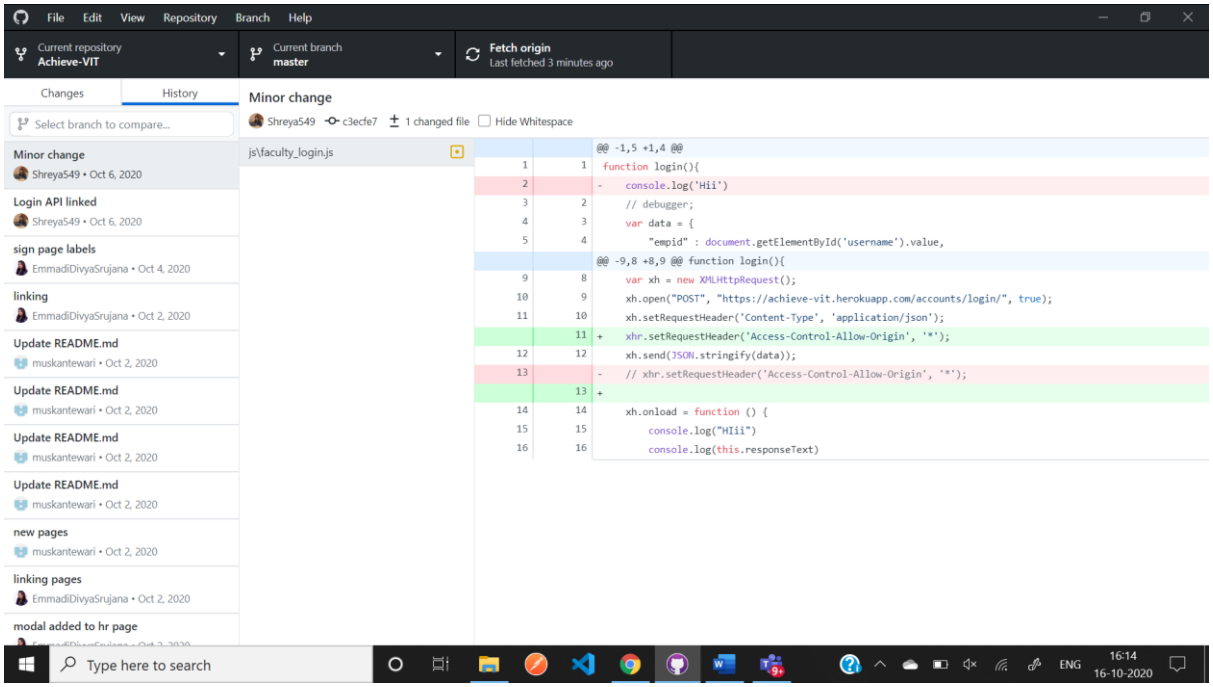
2) Create a new repository



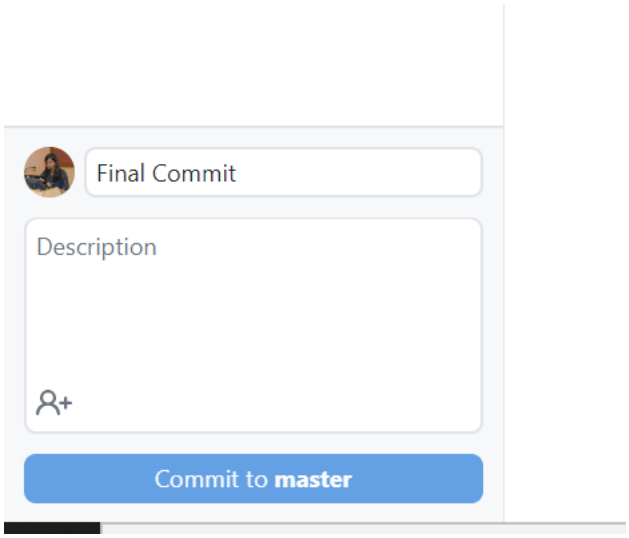
3) Change branch



4) Check commit history



5) Commit after making changes



Git Command Line Interface

After creating a repository or after forking a repository, users can use the command line interface to add or edit their codes.

To use it, users must download 'git' on their systems.

Most of them are given below

Basic Commands

- Clone a Repository

```
git init

 Turn an existing directory into a git repository

git clone https://github.com/Shreya549/OSP-DA.git

 Replace url with your required repository url
```

- Set username and email id

```
   git config --global user.name "FIRST_NAME LAST_NAME"

   git config --global user.email "MY_NAME@example.com"
```

- Pushing to a Repository

```
  git status
    show modified files in working directory, staged for your next commit

  git add .
  git commit -m <commit message>
  git push origin master

   To add just one file or a set of files replace the . with your filename
in first command

  git diff
    diff of what is changed but not staged
  git diff --staged
    diff of what is staged but not yet committed
  git commit -m "[descriptive message]"
    commit your staged content as a new commit snapshot
```

- Change most recent Git commit message

```
git commit --amend
```

- Check where is your Repository remote

```
  git remote -v

  Output should have the repository url similar to this

  origin       https://github.com/Shreya549/OSP-DA.git (fetch)
  origin       https://github.com/Shreya549/OSP-DA.git (push)
```

- Change Remote

```
git remote set-url origin <new git url>
```

- Check logs

```
    git log

   Output should be something similar

    commit 82e2a7c46a96b3b4aaf5acbc0cbc218d118aa922
    Author: <User> <45638240+<User>@users.noreply.github.com>
    Date:   Fri May 15 14:52:32 2020 +0530

    <Commit Message>
```

- Tracking changes

```
  git diff
    To track the changes that are yet to staged.

    git diff --staged
      To track the changes that are staged but not committed.

    git diff HEAD
      To track the changes after committing a file.

    git status
      To know the state of the files in local directory.

    git show
     To show all the changes made in the file for each commit
```

- Ignoring files

```
    Git can ignore specified files from adding into the remote repository
using     gitignore.

    Create .gitignore in the project

     vim .gitignore
```

Add the filename/directory you want to ignore by the git in the gitignore file

```
node_modules
```

now, when you add the files it ignores node_modules directory in your project.

Branches

- Create new branch

```
   git checkout -b newbranch

   Replace newbranch with your branch name
```

- Check current branch

```
   git branch

   This should list all your branches and highlight the current branch in green
```

- Switch to new branch

```
   git checkout newbranch
```

- Push in new branch

```
    git add .
    git commit -m <commit message>
    git push origin <branch name>
```

Update forked Repository with original Repository

These steps are to be followed when your forked repository is few commits behind original repository

1. Check if you have the forked repository added to your remotes.

```
git remote -v
```

If you see the forked repository listed in your remotes you can skip to point 3.

2. Add the forked repository as a remote

```
git remote add upstream https://github.com/whoever/whatever.git
```

3. Fetch changes from forked repository

```
git fetch upstream
```

Any new changes and branches from the original forked repository should now be fetched to your local repository.

4. Rebase your local branch, or merge the changes

You can now choose to either rebase your local branch, or merge the changes from the forked repository into your branch.

Use the following if you want to rebase your branch:

```
git rebase upstream/master
```

Or use the merge command if you want to merge instead:

```
git merge upstream/master
```

Replace "master" with the name of the branch on the forked repository that you want to rebase or merge with.

Your dev branch is X commits behind and Y commits ahead of master fix

```
  git checkout master

  git fetch origin master

  git checkout dev

  git rebase origin/master

  git checkout master

  git merge --no-ff dev

  If you get this message 'Automatic merge failed; fix conflicts and then commit the result'

  Check for merge conflicts in code and fix them (master branch should have required files now)

  For any such message deleted in 'dev and modified in HEAD. Version HEAD of requirements.txt left in tree'
```

```
  File can either be deleted or modified or kept same

  git pull origin master (if warning comes)

  add, commit, push
```

Short hands

```
    git commit -am <message>
      adds and commits the changes in a single command

    git push -u origin <branch name>
      use -u (upstream) for pushing your first commit changes into the remote
repository and later on you avoid origin <branch name>

   git push
      works fine till the last commit

    git diff > difference.txt
      If you are feeling hard to track all the changes on console
      above helps to writes/pipes the differences into specified file
(difference.txt) and you can track the changes easily
```

GitHub Command Line Interface

```
List Issues With GitHub CLI

1. gh issue list
list out the open GitHub Issues for our project

2. gh issue list --state "all"
  gh issue list -s "all"
If we want to list out ALL of the issues we could use the “state”
flag

3. gh issue list --assignee "n8ebel"
Now, maybe we’ve realized that is too many issues to sort through,
so we decide we only want to list out your currently assigned
issues.

4. gh issue status
Next, we want to check in on the status of a couple of the issues we
created yesterday.  Maybe we don’t remember their exact numbers, but
since we created them, we can use the status command to list them at
the terminal
```

5. gh issue list --state "closed"
gh issue list -s "closed"
After checking in on these issues, we still can't find the issue we're looking for, so we might want to check whether it was closed or not.

6. gh issue list --label "bug"
gh issue list -l "bug"
To list out all of our open bugs, we could filter by the "bug" label defined in our GitHub repo

7. gh issue view "15"
Once we've found an issue we want to fix, we might want to assign that issue to ourselves. Currently, we can't do that directly from the command line, but we can quickly open the issue from the command line using the "view" command.

8. gh issue create
We can use the gh issue create command to create a new GitHub Issue directly from the command line.

9. gh issue create -t "Sample Issue Title" -b "Sample issue description"
If you'd like to simplify things a bit, you can specify the issue with the command using additional flags

10. gh pr list
list the open pull requests for our project.

11. gh pr list --state "all"
gh pr list -s "all"
If we want to list out ALL of the pull requests, both open and closed, we could use the "state" flag

12. gh pr status
Next, we want to check in on the status of a couple of the PRs we created yesterday. Maybe we don't remember their exact numbers, but since we created them, we can use the status command to list them at the terminal

13. gh pr list --state "closed"
gh pr list -s "closed"
After checking in on these PRs, we still can't find the pull request we're looking for, so we might want to check whether it was closed or not.

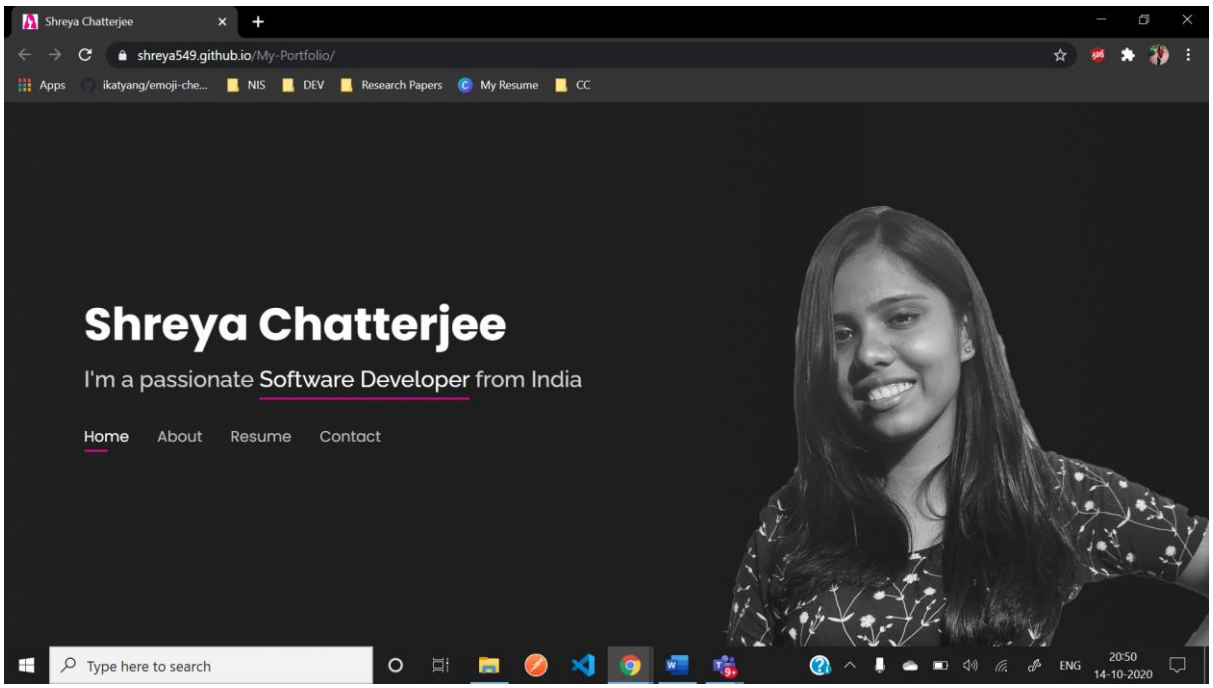
14. gh pr view "14"
Once we've found a PR we want to review, we might want to assign that PR to ourself. Currently, we can't do that directly from the

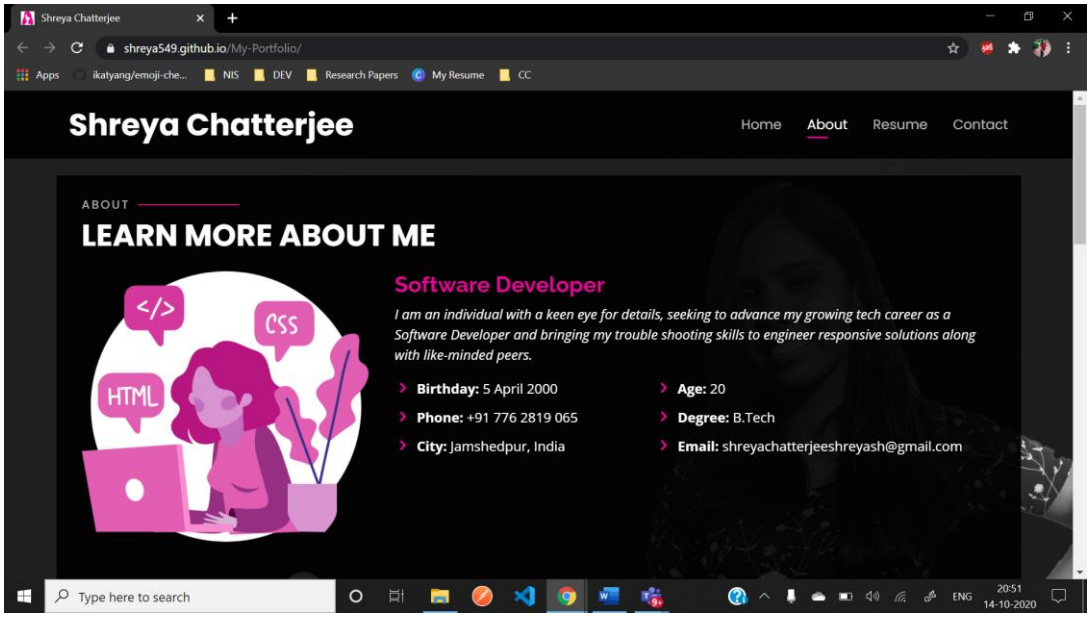
```
command line, but we can quickly open the PR from the command line
using the view command.

15. gh pr checkout
Check out pull requests locally.

16. gh pr create
Create a new pull request.

17. gh pr checks
View your pull requests' checks.

18. gh release create
Create a new release

19. gh alias set
Create a shortcut for a gh command.
```

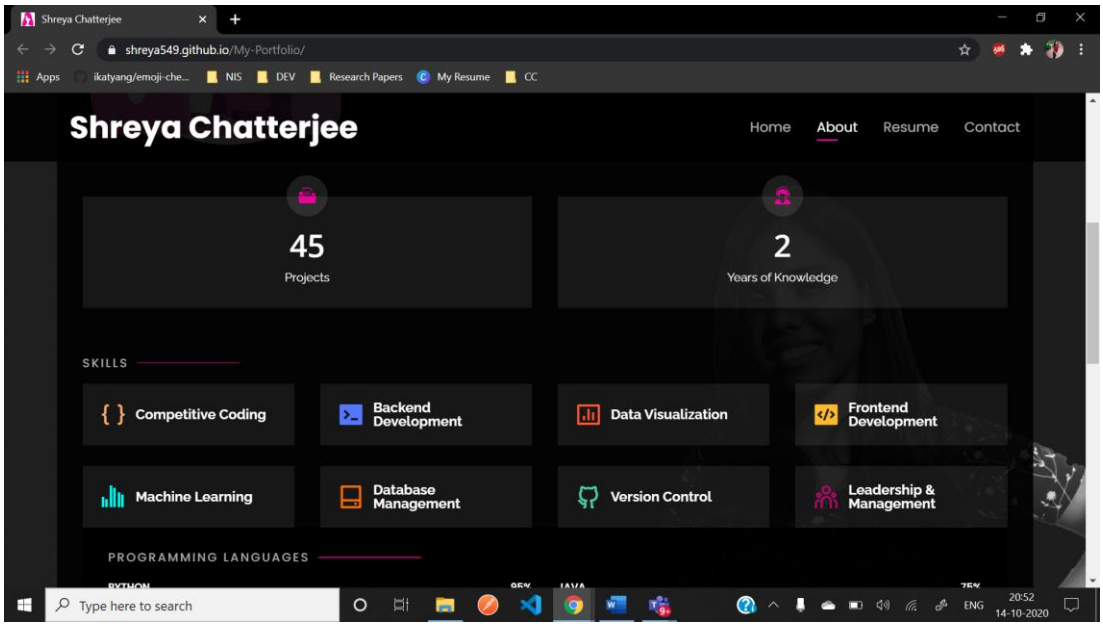
Screenshots of my Portfolio:

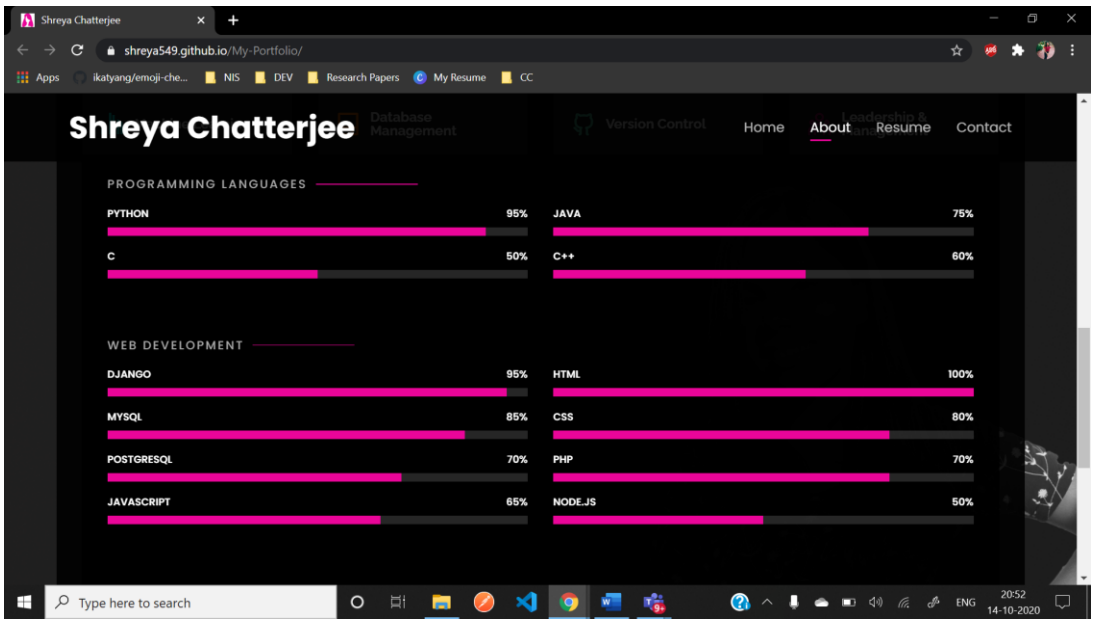
[Link to my portfolio on GitHub Pages]

https://github.com/Shreya549/My-Portfolio




Shreya Chatterjee
shreya549.github.io/My-Portfolio/
Shreya Chatterjee
Home
About
Resume
Contact
ABOUT
LEARN MORE ABOUT ME
Software Developer
I am an individual with a keen eye for details, seeking to advance my growing tech career as a Software Developer and bringing my trouble shooting skills to engineer responsive solutions along with like-minded peers.
Birthday: 5 April 2000
Age: 20
Phone: +91 776 2819 065
Degree: B.Tech
City: Jamshedpur, India
Email: shreyachatterjeeshreyash@gmail.com
Type here to search


Shreya Chatterjee
shreya549.github.io/My-Portfolio/
Shreya Chatterjee
Home
About
Resume
Contact
45
Projects
2
Years of Knowledge
SKILLS
Competitive Coding
Backend Development
Data Visualization
Frontend Development
Machine Learning
Database Management
Version Control
Leadership & Management
PROGRAMMING LANGUAGES
Type here to search


Shreya Chatterjee
shreya549.github.io/My-Portfolio/
Shreya Chatterjee
Home
About
Resume
Contact
PROGRAMMING LANGUAGES
PYTHON 95%
JAVA 75%
C 50%
C++ 60%
WEB DEVELOPMENT
DJANGO 95%
HTML 100%
MYSQL 85%
CSS 80%
POSTGRESQL 70%
PHP 70%
JAVASCRIPT 65%
NODE.JS 50%
Type here to search

Shreya Chatterjee

shreya549.github.io/My-Portfolio/

Apps | ikatyang/emoji-che... | NIS | DEV | Research Papers | My Resume | CC

Shreya Chatterjee

Home | About | Resume | Contact

DATA ANALYTICS

PYTHON	70%
R	50%

TOOLS

GITHUB	90%
POSTMAN	100%
HEROKU	90%

Shreya Chatterjee

shreya549.github.io/My-Portfolio/

Apps | ikatyang/emoji-che... | NIS | DEV | Research Papers | My Resume | CC

Shreya Chatterjee

Home | About | Resume | Contact

RESUME

CHECK MY RESUME

Education

B.TECH INFORMATION TECHNOLOGY

2018-2022

Vellore Institute of Technology, Vellore

CGPA - 9.17

HIGHER SECONDARY CERTIFICATE

2016-2018

Professional Experience

LEAD BACKEND DEVELOPER

2020 - Present

SchoolEase

- Lead in the Server-side Development of the Web-Application.
- Delegate tasks to the other members of the backend team and provide counsel on all aspects of the project.
- Supervise the assessment of all the REST - APIs in order to

Shreya Chatterjee

shreya549.github.io/My-Portfolio/

Apps | ikatyang/emoji-che... | NIS | DEV | Research Papers | My Resume | CC

Shreya Chatterjee

Home | About | Resume | Contact

CGPA - 9.17

HIGHER SECONDARY CERTIFICATE

2016-2018

Little Flower School, Jamshedpur (CISCE)

Percentage - 91.5%

SENIOR SECONDARY CERTIFICATE

2005-2016

Little Flower School, Jamshedpur (CISCE)

Percentage - 95.8%

- Lead in the Server-side Development of the Web-Application.
- Delegate tasks to the other members of the backend team and provide counsel on all aspects of the project.
- Supervise the assessment of all the REST - APIs in order to ensure accuracy of the code.

FINANCE DIRECTOR

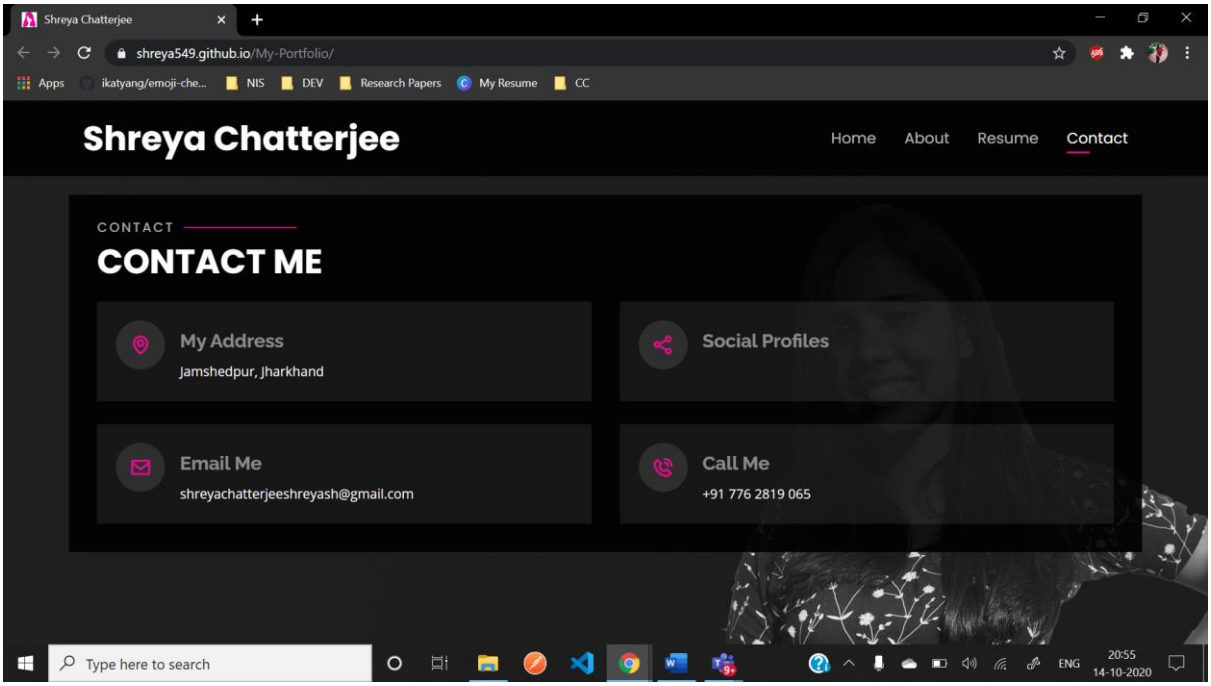
2020 - Present

CodeChef-VIT

- Ensure that the Finance is kept clean.
- Create the budget and manage the flow of resources to organise technical events throughout the year
- Organise and Manage events hosted by CodeChef-VIT.

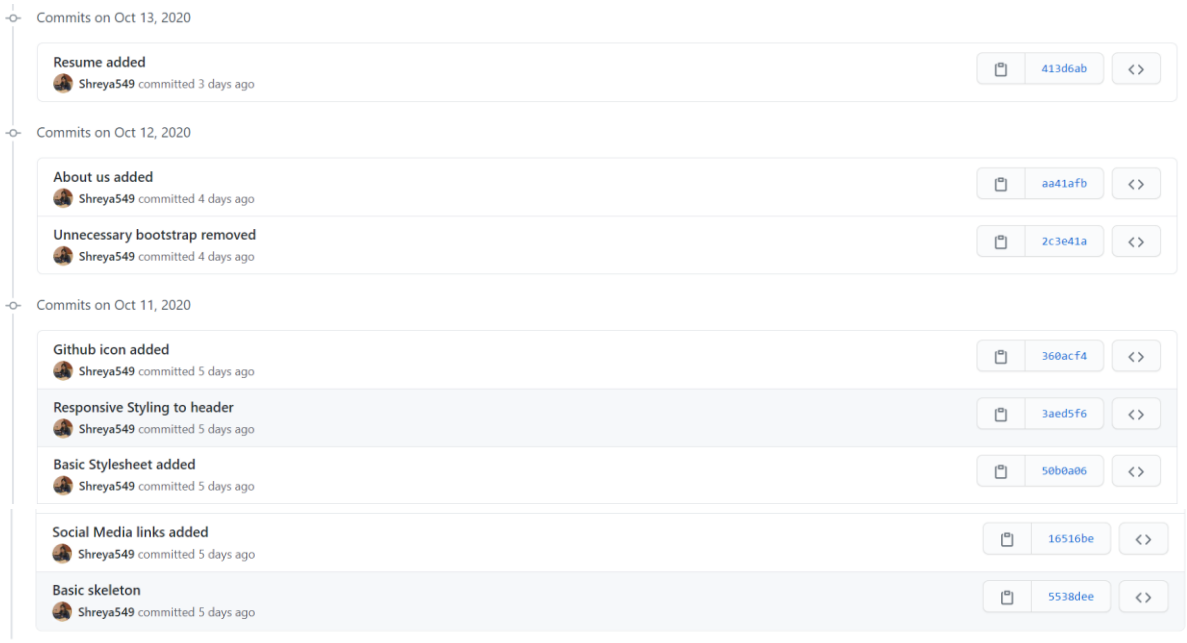
Type here to search

ENG 20:55 14-10-2020



Commit History

Graph fixed Shreya549 committed 2 days ago		a116678
Readme created Shreya549 committed 3 days ago	Verified	d763a22
Css changes Shreya549 committed 3 days ago		0ac12ee
Minor changes in index Shreya549 committed 3 days ago		7a894ca
CSS for Contact Me added Shreya549 committed 3 days ago		8eaa213
CSS for resume added Shreya549 committed 3 days ago		3903cda
CSS changed Shreya549 committed 3 days ago		69b8f3a
About Section added Shreya549 committed 3 days ago		c832c99
Css changed Shreya549 committed 3 days ago		daeb494
Contact me added Shreya549 committed 3 days ago		918f423
JS linked Shreya549 committed 3 days ago		9175819
JavaScript added Shreya549 committed 3 days ago		1b2c0fe
favicon added Shreya549 committed 3 days ago		5ce416d
Images added Shreya549 committed 3 days ago		a5c9925
Section css added Shreya549 committed 3 days ago		b68af4d
Bootstrap files Shreya549 committed 3 days ago		415d658
Linking the js files Shreya549 committed 3 days ago		9571cb5



Pros and Cons of GitHub:

Pros

- It has a large community which can easily contribute to open-source projects
- Collaborating on projects is easier with GitHub
- GitHub's intuitive design allows easy navigation and good user – experience
- Users can create issues or open pull requests to contribute on open source projects
- GitHub provides a good UI for checking Commit History
- GitHub provides an option to create project boards
- GitHub also supports creation of organisations and classrooms
- GitHub pages provide easy-hosting for Static Sites
- Provides an OAuth for Authentication on other platforms
- Provides a profile README.md

Cons

- Doesn't offer very good API development
- Slightly expensive for those in search of private repository
- It does not support first party CI/CD lifecycle

- Repositories can host information upto 1 GB and file size cannot exceed 100 MB.
- Reviewing large pull requests can be tedious and it can be tough to identify recent changes (e.g. a one line change) in new files or files with lots of changes
- Reversing merge operations and solving merge conflicts are difficult on GitHub

Features to be added to GitHub:

- First party CI/CD
- Dark themed UI
- Full-fledged GitHub CLI
- Organization README.md
- Hosting of Dynamic Sites on GitHub pages
- Time tracking of commits
- Importing Project from GitLab or BitBucket

Comparison between other Version Control Applications

	GitHub	**GitLab**	**BitBucket**	**Beanstalk**
Started In	2008	2011	2008	2012
Description	GitHub is the best place to share code with friends, co-workers, classmates, and complete strangers	GitLab offers git repository management, code reviews, issue tracking, activity feeds and wikis.	Bitbucket gives teams one place to plan projects, collaborate on code, test and deploy, all with free private Git repositories.	A single process to commit code, review with the team, and deploy the final result to your customers.
Total Users	40,000,000	100,000	30,000	-
Integrated CI/CD	No	Yes	Yes	Yes
Branch Permissions	No	Yes	Yes	Yes

CLI	Yes	No	Yes	Yes
Used by	Airbnb, Netflix, Reddit	Alibaba Group, GO-JEK, Craftbase	Figma, Trivago, Paypal	Accenture, Duda, Crowdkeep