Digital Assignment-1
Open Source Programming- ITE-1008

VIT®

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Name : Anshuman Arora
Registration Number :- 18BIT0059
Slot- B2
Faculty- Prof. Jayakumar S

Github Link: - https://github.com/anshuman-arora99/OSP-DA-1

Hosted Website: - https://anshuman-arora99.github.io/OSP-DA-1/

Write down the step by step process of GitHub working methodology and different ways to access GitHub.

Step 1 Create a repository

A repository is usually used to organize a single project. Repositories can contain folders and files, images, videos, spreadsheets, and data sets – anything your project needs. We recommend including a README, or a file with information about your project. GitHub makes it easy to add one at the same time you create your new repository. It also offers other common options such as a license file.

Step 2 Create a branch

When you're working on a project, you're going to have a bunch of different features or ideas in progress at any given time – some of which are ready to go, and others which are not. Branching exists to help you manage this workflow.

When you create a branch in your project, you're creating an environment where you can try out new ideas. Changes you make on a branch don't affect the main branch, so you're free to experiment and commit changes, safe in the knowledge that your branch won't be merged until it's ready to be reviewed by someone you're collaborating with.

Step 3 Make and commit changes

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 other contributors can understand what you've done and why.

Once your branch has been created, it's time to start making changes. Whenever you add, edit, or delete a file, you're making a commit, and adding them to your branch. This process of adding commits keeps track of your progress as you work on a feature branch.

Commits also create a transparent history of your work that others can follow to understand what you've done and why. Each commit has an associated commit message, which is a description explaining why a particular change was made. Furthermore, each commit is considered a separate unit of change. This lets you roll back changes if a bug is found, or if you decide to head in a different direction.

Step 4 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 green and red.

Pull Requests initiate discussion about your commits. Because they're tightly integrated with the underlying Git repository, anyone can see exactly what changes would be merged if they accept your request.

You can open a Pull Request at any point during the development process: when you have little or no code but want to share some screenshots or general ideas, when you're stuck and need help or advice, or when you're ready for someone to review your work. By using GitHub's @mention system in your Pull Request message, you can ask for feedback from specific people or teams, whether they're down the hall or ten time zones away.

Step 5 Merge your pull request

Once a Pull Request has been opened, the person or team reviewing your changes may have questions or comments. Perhaps the coding style doesn't match project guidelines, the change is missing unit tests, or maybe everything looks great and props are in order. Pull Requests are designed to encourage and capture this type of conversation.

You can also continue to push to your branch in light of discussion and feedback about your commits. If someone comments that you forgot to do something or if there is a bug in the code, you can fix it in your branch and push up the change. GitHub will show your new commits and any additional feedback you may receive in the unified Pull Request view.

Different access to access github

- ### Github Desktop

  GitHub Desktop is an open source tool that enables you to be more productive. GitHub Desktop encourages you and your team to collaborate using best practices with Git and GitHub.

  Just a few of the many things you can do with GitHub Desktop are:

  - Add changes to your commit interactively
  - Quickly add co-authors to your commit
  - Checkout branches with pull requests and view CI statuses
  - Compare changed images

- ### Github CLI

  GitHub CLI is an open source tool for using GitHub from your computer's command line. When you're working from the command line, you can use the GitHub CLI to save time and avoid switching context.

  You can use the following GitHub features with the GitHub CLI.

  - View, create, clone, and fork repositories
  - Create, close, and list issues and pull requests
  - Review, diff, and merge pull requests
  - Create, edit, list, and view gists

- ### Github for Mobile

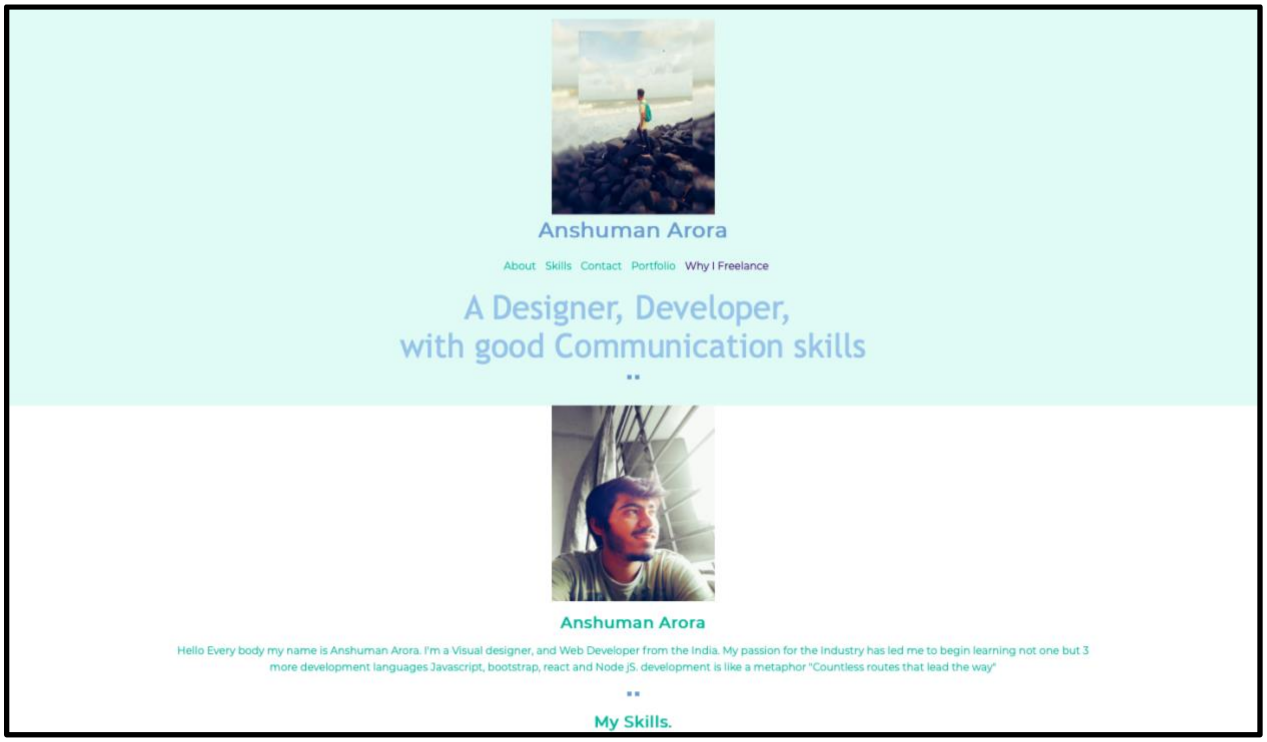
  GitHub for mobile is currently available for GitHub.com users as an Android and iOS app.

GitHub for mobile gives you a way to do high-impact work on GitHub quickly and from anywhere. GitHub for mobile is a safe and secure way to access your GitHub data through a trusted, first-party client application.

With GitHub for mobile you can:

- Manage, triage, and clear notifications
- Read, review, and collaborate on issues and pull requests
- Search for, browse, and interact with users, repositories, and organizations
- Receive a push notification when someone mentions your username

Host your Personal Portfolio in GitHub and provide the screenshot of the project and version history.



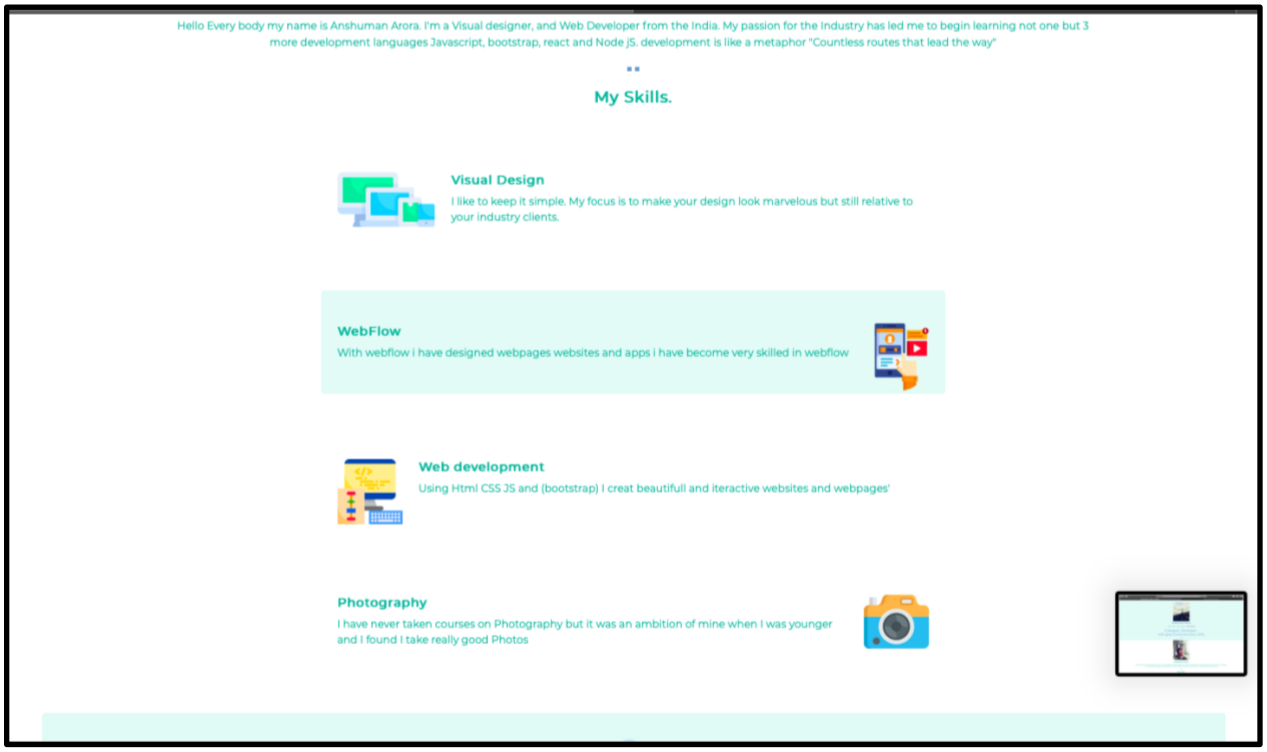
Anshuman Arora

A Designer, Developer,
with good Communication skills

Anshuman Arora

Hello Every body my name is Anshuman Arora. I'm a Visual designer, and Web Developer from the India. My passion for the Industry has led me to begin learning not one but 3 more development languages Javascript, bootstrap, react and Node jS. development is like a metaphor "Countless routes that lead the way"

My Skills.



Hello Every body my name is Anshuman Arora. I'm a Visual designer, and Web Developer from the India. My passion for the Industry has led me to begin learning not one but 3 more development languages Javascript, bootstrap, react and Node jS. development is like a metaphor "Countless routes that lead the way"

My Skills.

Visual Design

I like to keep it simple. My focus is to make your design look marvelous but still relative to your industry clients.

WebFlow

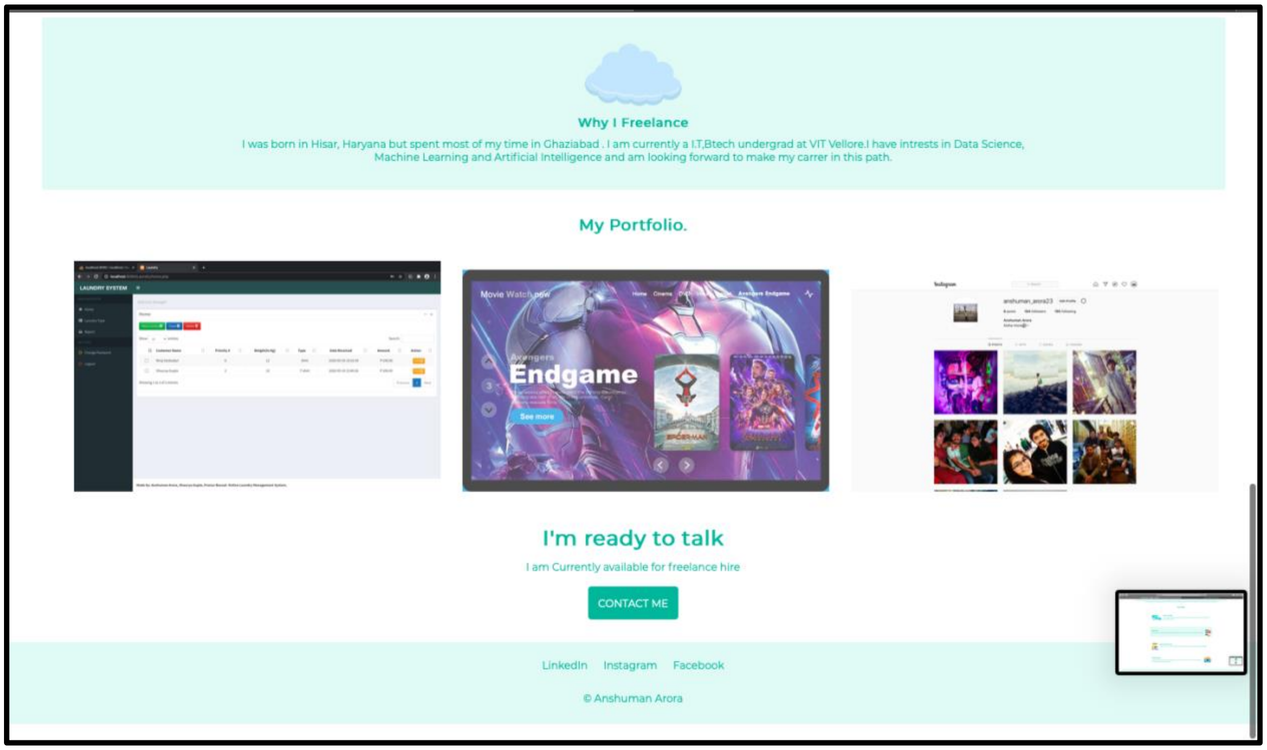
With webflow i have designed webpages websites and apps i have become very skilled in webflow

Web development

Using Html CSS JS and (bootstrap) I creat beautifull and iteractive websites and webpages'

Photography

I have never taken courses on Photography but it was an ambition of mine when I was younger and I found I take really good Photos



Why I Freelance

I was born in Hisar, Haryana but spent most of my time in Ghaziabad . I am currently a I.T,Btech undergrad at VIT Vellore.I have intrests in Data Science, Machine Learning and Artificial Intelligence and am looking forward to make my carrer in this path.

My Portfolio.

I'm ready to talk

I am Currently available for freelance hire

CONTACT ME

LinkedIn Instagram Facebook

© Anshuman Arora

Github Link: - https://github.com/anshuman-arora99/OSP-DA-1

Hosted Website: - https://anshuman-arora99.github.io/OSP-DA-1/

Write down the pros and cons of GitHub.

<u>Pros:</u> -

- Version Control: GitHub, being built over Git, makes it fast and easy to develop projects in versions/branches and easily rollback to previous versions when necessary.
- Pull Requests/Review: GitHub has a powerful UI for creating pull requests, with useful tools like inline commenting and more recently "suggested changes". Pull request history is always maintained and easy to search.
- Collaboration/Auditing: It's easy for multiple team members to work on the same project and merge changes (often) seamlessly. All contributions are tracked so it's easy to identify contributors.
- Industry Standard: GitHub is used by virtually all major open source projects so it's very easy to find and contribute to projects of interest if you're well versed with GitHub.

<u>Cons: -</u>

- 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.
- It should be a bit harder to push unresolved merge conflicts, we've had these slip through once in a while.
- You have to be careful with merge operations; a bad merge can be painful to reverse.

List down the features that need to be added in GitHub.

- Ability to choose a licence type for each project, with auto-generated licence.
- Ability to tag my own repositories and tags others too for my own usage
- A real dashboard, that aggregates activity by tags and projects
- A project activity indicator based on recents commits, open issues, PR opened/closed ratio...

- A way to distinguish the most active fork in a project.
- Ability to star issues
- A way to prioritize issues and sort by priority
- When listing issues by assignee, please add a milestone indicator
- Prevent page refresh while editing !!
- Prevent co-editing an issue comment (or version/history it)

Compare the minimum of three version control applications

1. Apache Subversion (SVN)

SVN was created as an alternative to CVS that would fix some bugs in the CVS system while maintaining high compatibility with it.To prevent corruption in the database from being corrupted, SVN employs a concept called atomic operations. Either all of the changes made to the source are applied or none are applied, meaning that no partial changes will break the original source.

Many developers have switched to SVN as it is a newer technology that takes the best features of CVS and improves upon them.SVN is designed to allow for it, lending itself better to large, forked projects with many directions.

Criticism of SVN includes slower comparative speed and the lack of distributed revision control. Distributed revision control uses a peer-to-peer model rather than using a centralized server to store code updates. While a peer-to-peer model would work better for world-wide, open source projects, it may not be ideal in other situations. The downside to a dedicated server approach is that when the server is down, no clients are able to access the code.

Pros:

- Newer system based on CVS

- Includes atomic operations
- Cheaper branch operations
- Wide variety of plug-ins for IDEs
- Does not use peer-to-peer model

Cons:

- Still contains bugs relating to renaming files and directories
- Insufficient repository management commands
- Slower comparative speed

2. GIT

It is primarily developed for Linux and has the highest speeds on there.It will also run on other Unix-like systems, and native ports of Git are available for Windows as msysgit.As there is no centralized server, Git does not lend itself to single developer projects or small teams as the code may not necessarily be available when using a non-repository computer. Workarounds exist for this problem, and some see Git's improved speed as a decent tradeoff for the hassle.

Git also comes equipped with a wide variety of tools to help users navigate the history system. Each instance of the source contains the entire history tree, which can be useful when developing without an internet connection.

Pros:

- Great for those who hate CVS/SVN
- Dramatic increase in operation speed

- Cheap branch operations
- Full history tree available offline
- Distributed, peer-to-peer model

Cons:

- Learning curve for those used to SVN
- Not optimal for single developers
- Limited Windows support compared to Linux

3. Mercurial

It was originally made to compete with Git for Linux kernel development, and as Git was selected, Mercurial has seen less success in that area. However, that is not to say that it is not used as many major developments use it, including OpenOffice.org.

It's different from other revision control systems in that Mercurial is primarily implemented in Python as opposed to C, but there are some instances where C is used.

Due to its distributed nature and its creation in Python, the Python language developers are considering a switch to Mercurial as it would allow non-core developers to have easier access to creating new trees and reverting changes.

Users have noted that Mercurial shares some features with SVN as well as being a distributed system, and because of the similarities, the learning curve for those already familiar with SVN will be less steep. The documentation for Mercurial also is more complete and will facilitate learning the differences faster.

Some of the major drawbacks to Mercurial include that it doesn’t allow for two parents to be merged and unlike Git, it uses an extension system rather than being scriptable. That may be ideal for some programmers, but many find the power of Git to be a feature they don’t want to trade off.

Pros:

- Easier to learn than Git
- Better documentation
- Distributed model

Cons:

- No merging of two parents
- Extension-based rather than scriptability
- Less out of the box power