<u>WHAT IS GITHUB?</u>

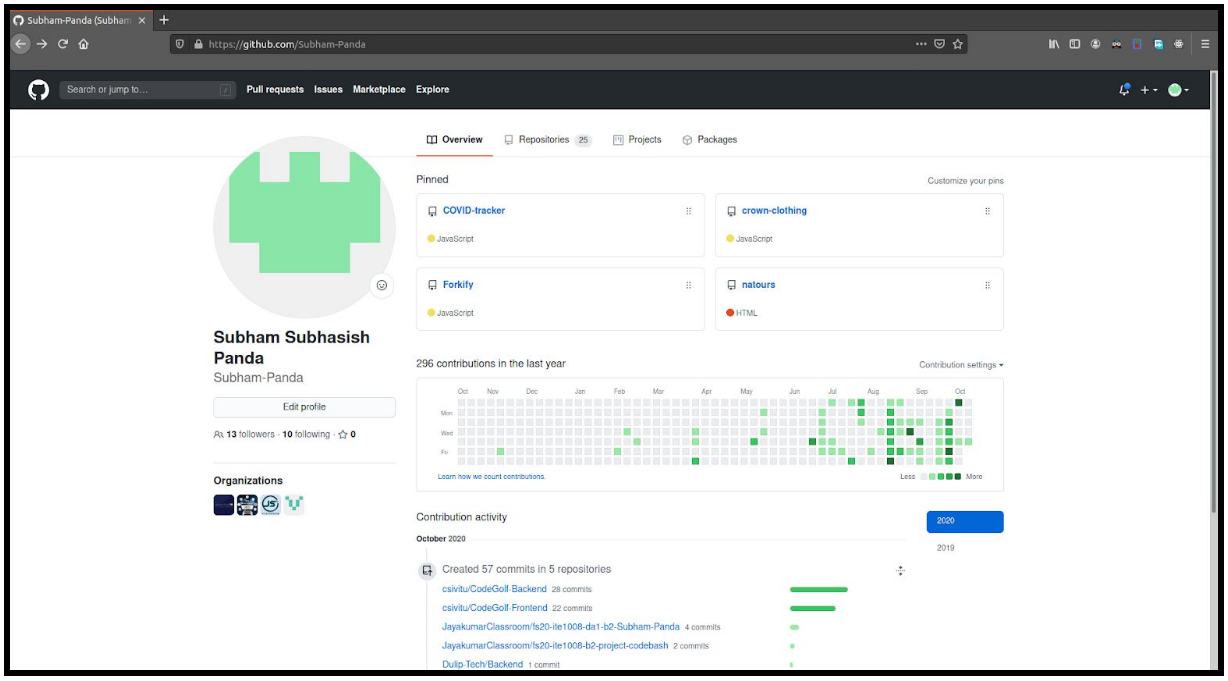
Github is a software that acs as a version-control system. A version-control system is a piece of software that lets the user keep track of various files and programs and the changes that are made to them over time. It also lets various users collaborate on the same code. It hosts a server program to hold your files and also lets users host static files.

To operate on github, first you need to have a github account by going to https://www.github.com

To store each project on github, we need to create what is called a repository for each project. A repository can be kind of thought of as a folder in which we store our project files.

Follow the following steps to create a repository:

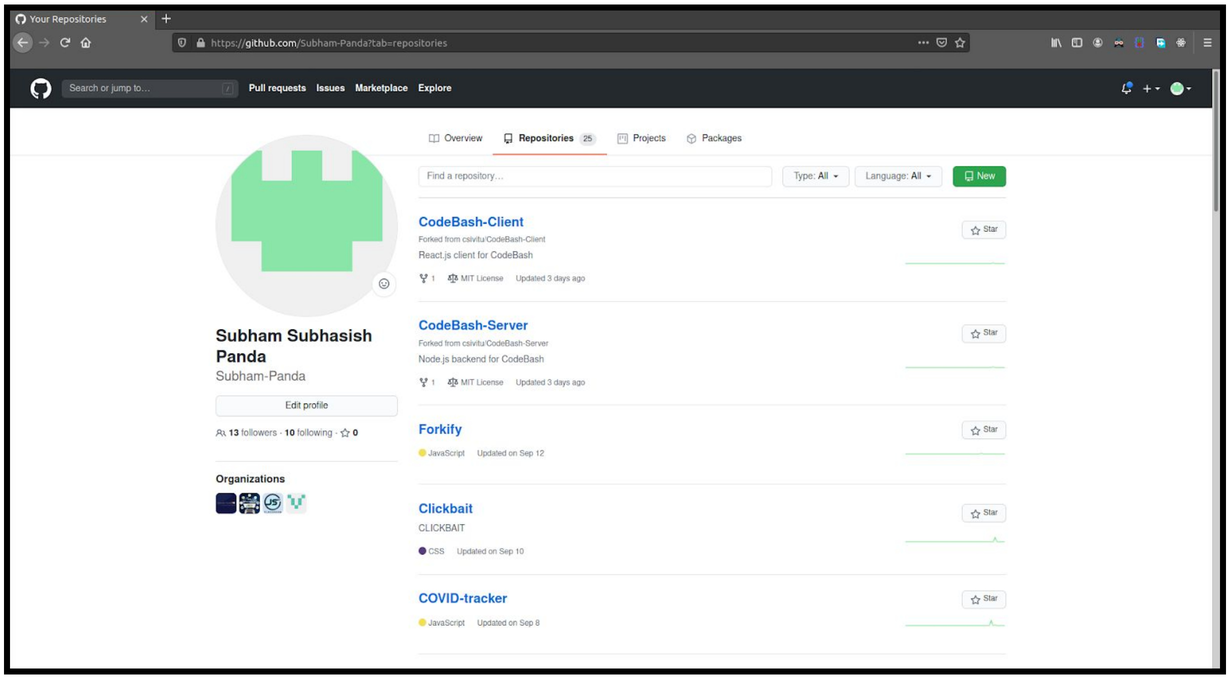
1. Go to https://www.github.com/<USERNAME>, here replace <USERNAME> with you github username. This would take you to a page as shown below



Created 57 commits in 5 repositories

csivitu/CodeGolf-Backend 28 commits
csivitu/CodeGolf-Frontend 22 commits
JayakumarClassroom/fs20-ite1008-da1-b2-Subham-Panda 4 commits
JayakumarClassroom/fs20-ite1008-b2-project-codebash 2 commits
Dulip-Tech/Backend 1 commit

2. Click on the Repositories tab. It would take you to a page with the list of your repositories



Subham Subhasish Panda
Subham-Panda

13 followers · 10 following · 0

CodeBash-Client
Forked from csivitu/CodeBash-Client
React.js client for CodeBash
1 MIT License Updated 3 days ago

CodeBash-Server
Forked from csivitu/CodeBash-Server
Node.js backend for CodeBash
1 MIT License Updated 3 days ago

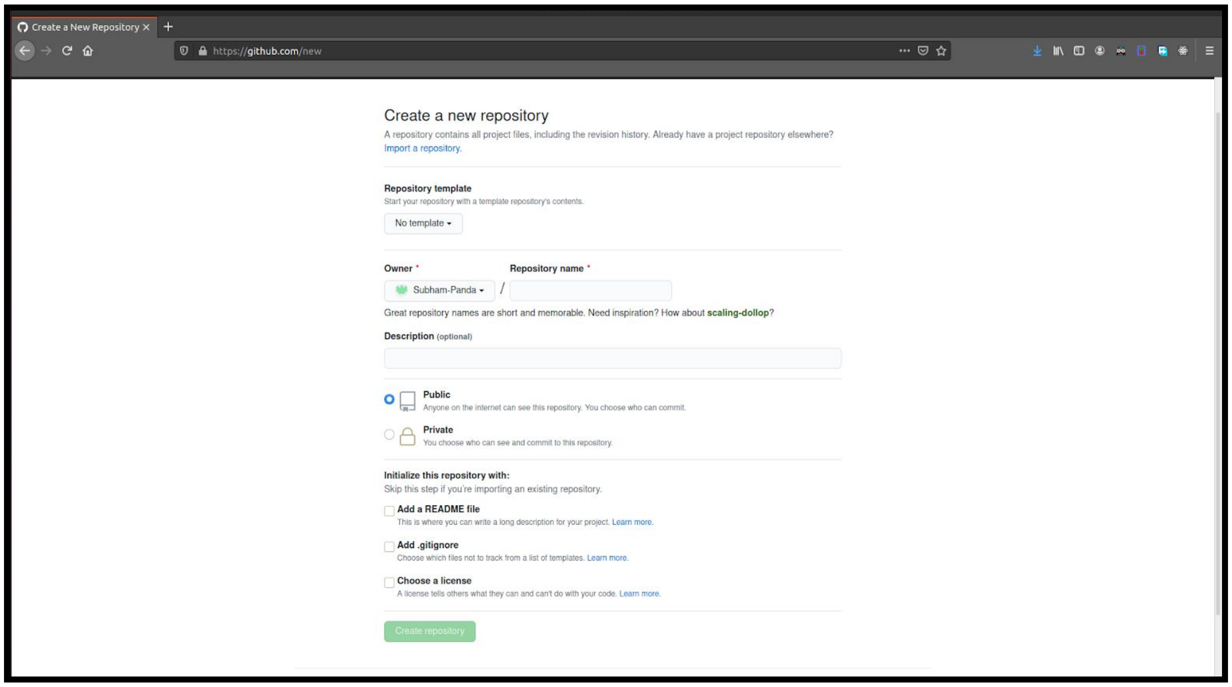
Forkify
JavaScript Updated on Sep 12

Clickbait
CLICKBAIT
CSS Updated on Sep 10

COVID-tracker
JavaScript Updated on Sep 8

3. Click on the green button New to create a new repository.

It takes you to the following page:



Create a new repository
A repository contains all project files, including the revision history. Already have a project repository elsewhere?
Import a repository.

Repository template
Start your repository with a template repository's contents.
No template

Owner * **Repository name** *
Subham-Panda /
Great repository names are short and memorable. Need inspiration? How about **scaling-dollop**?

Description (optional)

Public
Anyone on the internet can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

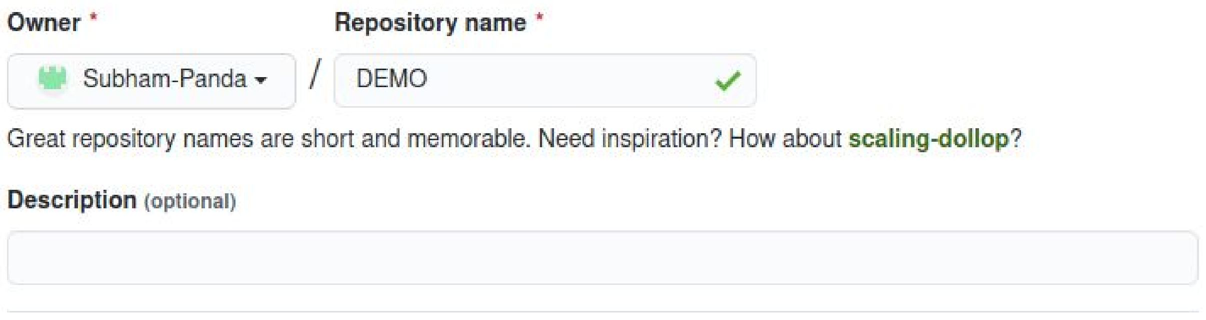
Add a README file
This is where you can write a long description for your project. Learn more.

Add .gitignore
Choose which files not to track from a list of templates. Learn more.

Choose a license
A license tells others what they can and can't do with your code. Learn more.

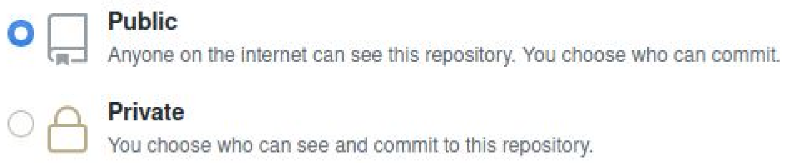
Create repository

4. Select a Repository name for you repo and an optional description for your repository



Great repository names are short and memorable. Need inspiration? How about scaling-dollop?

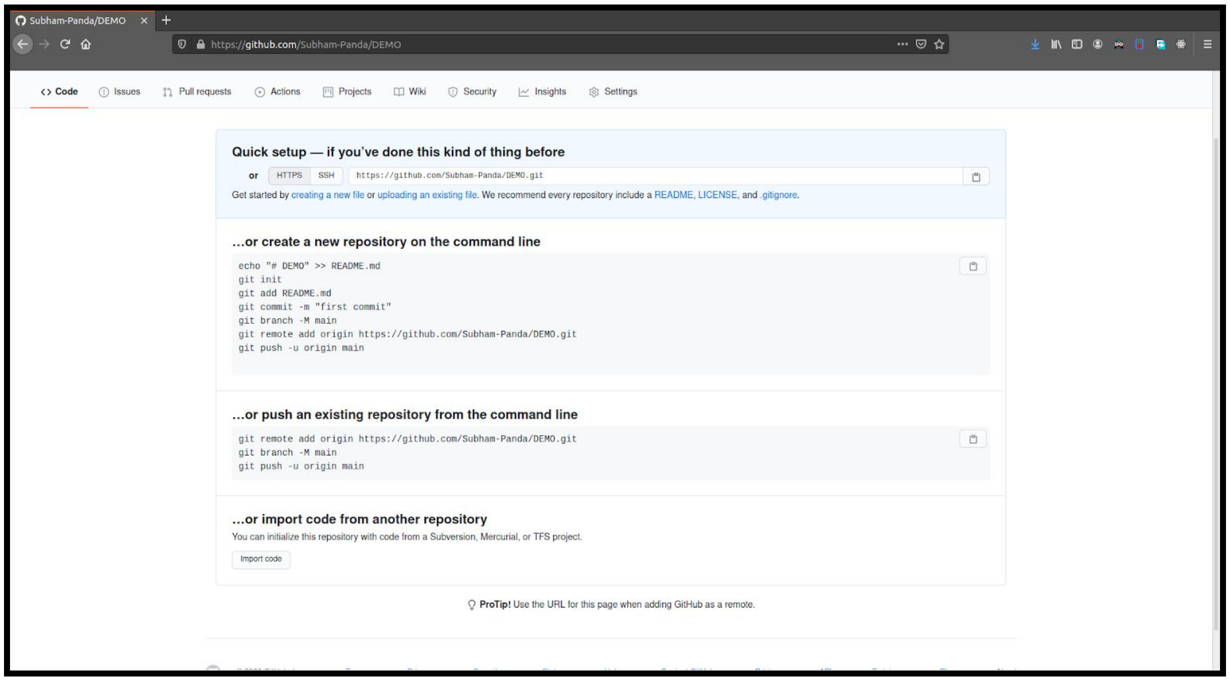
5. Select whether to make the repository Public or Private. A **Public Repository** is one which is visible to anyone visiting your profile whereas a **Private Repository** is one which is not visible to others visiting your github profile. Select one depending on your requirement.



[✓ Public] Anyone on the internet can see this repository. You choose who can commit.

Private: You choose who can see and commit to this repository.

6. Click on the Create Repository button at the bottom of the page. This will take you to the following page.



Quick setup — if you've done this kind of thing before

or HTTPS SSH `https://github.com/Subham-Panda/DEMO.git`

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# DEMO" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/Subham-Panda/DEMO.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/Subham-Panda/DEMO.git
git branch -M main
git push -u origin main
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

ProTip! Use the URL for this page when adding GitHub as a remote.

Now the repository has been created where you can store your files and your code for your project.

The following steps written below shows what are various stages in which github works and how to perform those along with an explanation:

1. In your command line go to the folder whose files you want to push to the repository.

2. Execute the command **git init** inside that folder.

The command git init creates a .git folder inside your current working directory. This git folder is responsible for tracking your files locally on your system and all the changes that happen to them over time.

3. Execute the command **git add .** to get all the files to the staging area or **git add <filenames separated by spaces>** to get only particular files to the staging area.

Staging area is a hypothetical place where once placed the files are started to be tracked from their previous versions for changes. Once tracked, now we can commit those files with a message.

4. To commit the changes with a message execute the command **git commit -m "message"** where message is replaced by an appropriate message.

Commit takes the files from the staging area and commits the. A commit is kind of a checkpoint you create for your files, so that you can come back to that particular checkpoint of your file and reverse changes if required anytime.

5. Execute **git branch -M main**, to first create a branch of your repository, this command creates the main branch for your repository.

We will talk more about branches later in the report.

6. Once done with git add and git commit, the status of your files are now saved in version control locally. To push them to your online repo, first we need to link you online

repo to your local git. To do that we execute the command **git remote add origin <github repo url>**, which in this case is **git remote add origin https://github.com/Subham-Panda/DEMO.git**

The above command sets the remote git to the online github repository.

7. Now to upload your files to the online repository, execute **git push -u origin main.**

Now you can refresh your repository page to view that your files have been uploaded.

MORE ABOUT BRANCHES IN GITHUB:

In github we can make different branches for a repository you are working on. What happens when you create a new branch? Well, doing so creates a new pointer for you to move around. Let's say you want to create a new branch called testing. You do this with the **git branch** command: **git branch testing**

Git keeps a special pointer called HEAD. In git, this is a pointer to the local branch you are currently working on. git branch command just creates a new branch and does not switch to that branch.

You can easily see this by running a simple git log command that shows you where the branch pointers are pointing. This option is called --decorate.
git log --oneline --decorate

To switch between branches, use the command git checkout, so in this case to switch from main to testing branch, use the command **git checkout testing**.

That command did two things. It moved the HEAD pointer back to point to the master branch, and it reverted the files in your working directory back to the snapshot that master points to. This also means the changes you make from this point forward will diverge from an older version of the project. It essentially rewinds the work you've done in your testing branch so you can go in a different direction.

Usually different branches are created to work on different modules on a project, so that different developers can work separately on their part of the projects. Once done we can merge the content of all branches by using the command **git merge.**

There also exists a single command to create a new branch and switch to it at the same time. The command is **git checkout -b testing**. This command both creates a branch testing and switches to it.

MORE GIT COMMANDS:

- **git status -** This commands gives the status of which files are untracked, which files are tracked but not committed and also which branch are you currently residing on
- **git log** - This gives a log of all the commits done with the timestamp, commit id and the commit messages made over time.

DIFFERENT WAYS TO ACCESS GITHUB:

There are three major ways to access github :

1. Git CLI (Git Command Line Interface)
2. Github Desktop
3. Github from Web

We have already discussed in detail how to access github from CLI, now let's have a look at the other ways to access Github.

Using Github Desktop:

1. While viewing your GitHub repo in the browser, click **Clone or download** and select **Open in Desktop**.

2. Open GitHub Desktop client and go to **File > Clone Repository**.

3. In the confirmation dialog, select **Open GitHub Desktop.app**. A window will launch with a “Clone a Repository”box about where to clone the repository. If desired, you can change the local path.

4. Click the **URL** tab, and then paste in the clone URL. In the **Local Path** field, select where you want the repo cloned.

5. Click **Clone**

6. Once changes are made in the folder, go back to GitHub Desktop. You’ll see the new file you added in the list of uncommitted changes on the left.

In the list of changed files, the green + means you've added a new file. A yellow circle means you've modified an existing file.

7. In the lower-left corner of the GitHub Desktop client, type a commit message, and then click **Commit to master**.

8. Click **Push origin** at the top.

To create different branches, follow the following steps:

1. Go to **Branch > New Branch** and create a new branch. Call it "testing" branch, and click **Create Branch**.

When you create the branch, you'll see the Current branch drop-down menu indicating that you're working in that branch.

2. Commit the changes using the options in the lower-left corner, and click **Commit to testing.**

3. Click **Publish branch** (on the top of the GitHub Desktop window) to make the local branch also available on origin (GitHub).
To merge branches, follow the following steps:

1. In the GitHub Desktop client, switch to the branch you want to merge the testing branch into. From the branch selector, select the **master** branch.

2. Go to **Branch > Merge into Current Branch**.

3. In the merge window, select the **testing** branch, and then click **Merge testing into master**.

4. Then click **Push origin** to push the changes to origin.

Using Github Web:

1. Go to the repository where you want to operate.

2. Click on **Add file** if and wherever you want to create a new file. Once created, put a commit message as required and click on **Commit Changes**.

3. If you want to change a file, select on the file and then select on the edit icon. Make changes as required, put a commit message at the bottom field given, and then click on the **Commit Changes** button.

To create different branches, follow the following steps:

1. Select the dropdown at the top right of the screen.

2. Write the name of the branch to be created in the given field and then select **Create branch**.

3. To switch between branches, select the dropdown and select the branch required.

To merge branches, follow the following steps:

1. Select the branch.

2. Go to the Pull Requests tab.

3. Create a Pull Request to merge the required branch to the required branch.

4. Go to the master branch, and go to the Pull Requests tab, to view the pending Pull requests and then select the **Merge** button.

<u>PROS OF GITHUB:</u>

1. Github has great support for Markdown. Markdown allows you to use a simple text editor to write formatted documents to let the user write documents like README for the repo.

2. Github has some of the best documentation. All topics related to git are well padded and written down in the github docs.

3. Github has a feature called Gists, which lets you convert one or several files into a working git repository.

4. Github has Github Pages, which lets you host static websites by simple assigning HTML pages onto another, separate repository.

5. Github lets developers collaborate on a project and work seamlessly, with new users requiring no major steup.

6. Since the repositories are hosted online, it provides a nice way to store code and version history and is accessible from anywhere around the globe.

7. Github also provides the features of Private Repositories to keep projects private and not visible to others.

8. Github provides features like Github Student Pack and Github Teacher Pack which lets emerging developers a lot of premium services for free.

CONS OF GITHUB:

1. Github offers private repositories but they are not perfect for many. You are putting all of this in the hands of Github as well as anyone who has login credentials, which like any other platform had security branches before.

2. Some of GitHub features, as well as features on other online repositories, are locked behind a SaaS paywall. If you have a large team, this can add up fast. Those who already have a dedicated IT team and their own internal servers are often better off using their own internal git for cost reasons

3. 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.

4. It sometimes becomes difficult to push unresolved merge conflicts.

5. You have to be careful with merge operations; a bad merge can be painful to reverse.

FEATURES REQUIRED TO BE ADDED TO GITHUB:

1. A feature to view commits by username.

2. Conflict management could be improved.

3. Implementation of two factor authentication to improve security.

4. Syntax highlighting for various languages could be improved.

5. Cross repo issue tracking can be improved.

6. The process for accepting invitations is kind of odd and doesn't always provide direct instructions to end-users. It can be improved

COMPARISON BETWEEN VERSION CONTROL SYSTEMS GIT , CVS, SVN and MERCURIAL:

Version Control is a very common feature of Software Configuration Management tools. There are plenty of options concerning which version control framework is the best. Every version control system has common properties like digital storage, centralized version history, code version management, isolated code branches etc.

Most common version control systems are GIT, CVS, SVN and Mercurial:

- **Git** was created by Linus Torvalds in 2005.It is written in C, Shell, Perl, TCL(Tool Command Language), and Python.
- **CVS**, also known as the Concurrent Versioning System, is a free client-server revision control system in the field of software. It was developed in the UNIX operating system environment and is available in both Free Software Foundation and commercial versions.
- **SVN** is a free and open source software under Apache license. It is written in C language. SVN was created as an alternative to CVS(Concurrent Versions System) with the aim to fix some bugs and make it more reliable to use.
- **Mercurial** was introduced close to Git and is also a distributed peer-to-peer system. Mercurial is a distributed revision-control tool which is written majorly in Python and some small sections in C Language.

VERSION CONTROL SYSTEM	**Features**	**Pros**	**Cons**
GIT	1. Open source version control system 2. Provides strong support for non-linear development. 3. Distributed repository model. 4. Compatible with existing systems and protocols like HTTP, FTP, ssh. 5. Capable of efficiently handling small to large sized projects. 6. Cryptographic authentication of history. 7. Toolkit-based design. 8. Periodic explicit object packing. 9. Garbage accumulates until collected.	1. Super-fast and efficient performance. 2. Cross-platform 3. Code changes can be very easily and clearly tracked. 4. Easily maintainable and robust. 5. Offers an amazing command line utility known as git bash. 6. Also offers GIT GUI where you can very quickly re-scan, state change, sign off, commit & push the code quickly with just a few clicks.	1. Complex and bigger history log become difficult to understand. 2. Does not support keyword expansion and timestamp preservation.
CVS	1. Client-server repository model. 2. Multiple developers might work on the same project parallelly. 3. CVS client will keep the working copy of the file up-to-date and requires manual intervention only when an edit conflict occurs 4. Keeps a historical snapshot of the project. 5. Anonymous read access. 6. ‘Update’ command to keep	1. Excellent cross-platform support. 2. Robust and fully-featured command-line client permits powerful scripting 3. Helpful support from vast CVS community 4. allows good web browsing of the source code repository	1. No integrity checking for source code repository. 2. Does not support atomic check-outs and commits. 3. Poor support for distributed source control. 4. Does not support signed revisions and merge tracking.

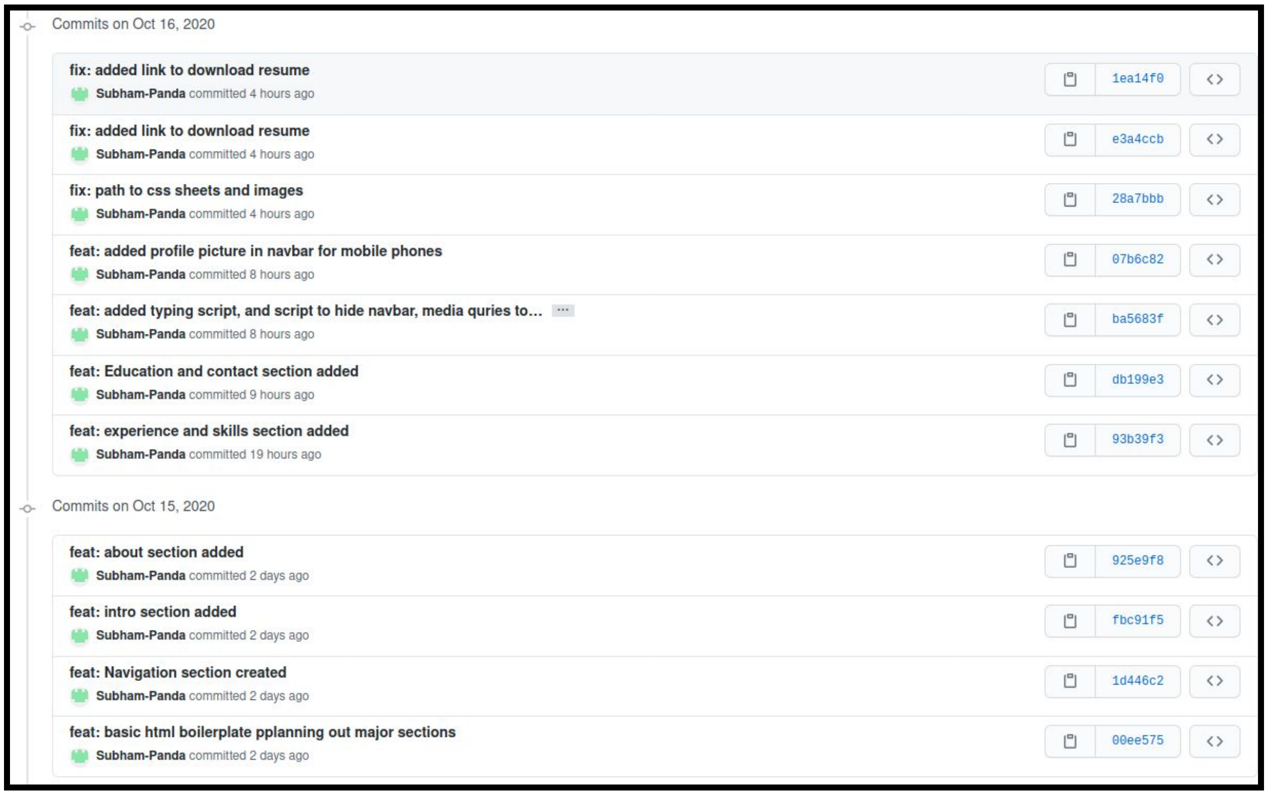
	local copies up to date. 7. Can uphold different branches of a project. 8. Excludes symbolic links to avoid a security risk. 9. Uses delta compression technique for efficient storage.	5. It's a very old, well known & understood tool. 6. Suits the collaborative nature of the open-source world splendidly.	
SVN	1. Client-server repository model. However, SVK permits SVN to have distributed branches. 2. Directories are versioned. 3. Copying, deleting, moving and renaming operations are also versioned. 4. Supports atomic commits. 5. Versioned symbolic links. 6. Free-form versioned metadata. 7. Space efficient binary diff storage. 8. Branching is not dependent upon the file size and this is a cheap operation. 9. Other features-merge tracking, full MIME support, path-based authorization, file locking, standalone server operation.	1. Has a benefit of good GUI tools like TortoiseSVN. 2. Supports empty directories. 3. Have better windows support as compared to Git. 4. Easy to set up and administer. 5. Integrates well with Windows, leading IDE and Agile tools.	1. Still contains bugs relating to renaming files and directories. 2. Insufficient repository management commands. 3. Slower comparative speed. 4. Does not support signed revisions.
Mercurial	1. High performance and scalability. 2. Advanced branching and merging capabilities.	1. Fast and powerful 2. Easy to learn 3. Lightweight and	1. Partial checkouts are not allowed. 2. Quite problematic when used with

	3. Fully distributed collaborative development. 4. Decentralized 5. Handles both plain text and binary files robustly. 6. Possesses an integrated web interface.	portable. 4. Conceptually simple	additional extensions.

PORTFOLIO WEBSITE

Visit the website at https://subham-panda.github.io/OSPportfolio/

Version History on GITHUB:



Commits on Oct 16, 2020

fix: added link to download resume
Subham-Panda committed 4 hours ago — 1ea14f0

fix: added link to download resume
Subham-Panda committed 4 hours ago — e3a4ccb

fix: path to css sheets and images
Subham-Panda committed 4 hours ago — 28a7bbb

feat: added profile picture in navbar for mobile phones
Subham-Panda committed 8 hours ago — 07b6c82

feat: added typing script, and script to hide navbar, media quries to... …
Subham-Panda committed 8 hours ago — ba5683f

feat: Education and contact section added
Subham-Panda committed 9 hours ago — db199e3

feat: experience and skills section added
Subham-Panda committed 19 hours ago — 93b39f3

Commits on Oct 15, 2020

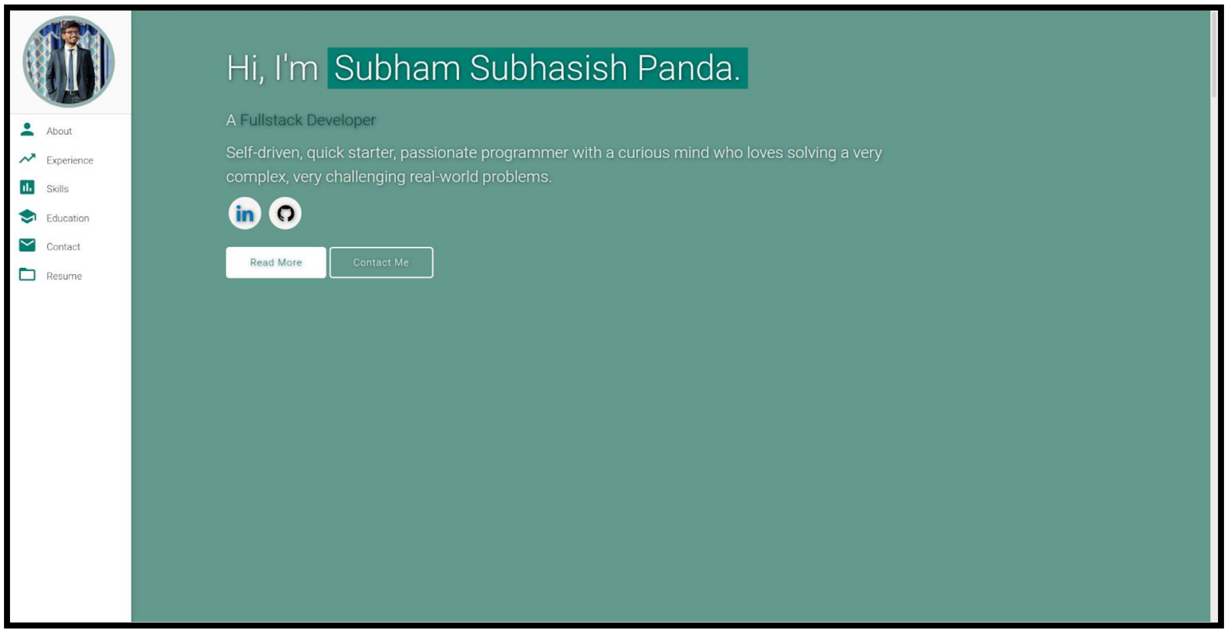
feat: about section added
Subham-Panda committed 2 days ago — 925e9f8

feat: intro section added
Subham-Panda committed 2 days ago — fbc91f5

feat: Navigation section created
Subham-Panda committed 2 days ago — 1d446c2

feat: basic html boilerplate pplanning out major sections
Subham-Panda committed 2 days ago — 00ee575

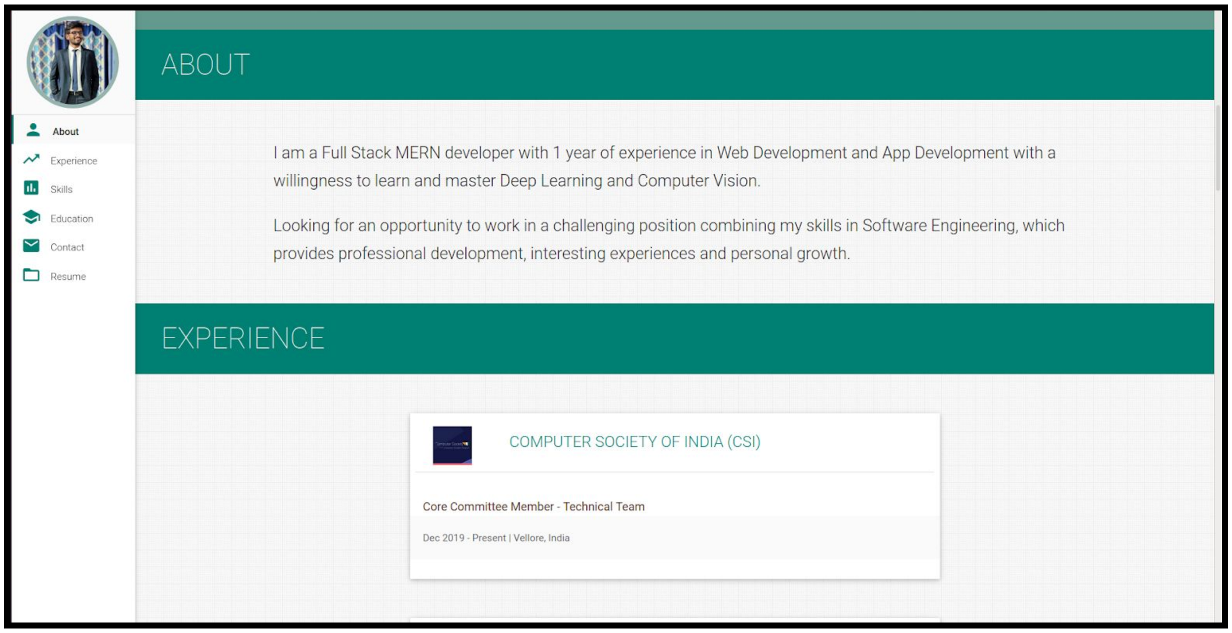
Desktop View:



Hi, I'm Subham Subhasish Panda.

A Fullstack Developer

Self-driven, quick starter, passionate programmer with a curious mind who loves solving a very complex, very challenging real-world problems.



ABOUT

I am a Full Stack MERN developer with 1 year of experience in Web Development and App Development with a willingness to learn and master Deep Learning and Computer Vision.

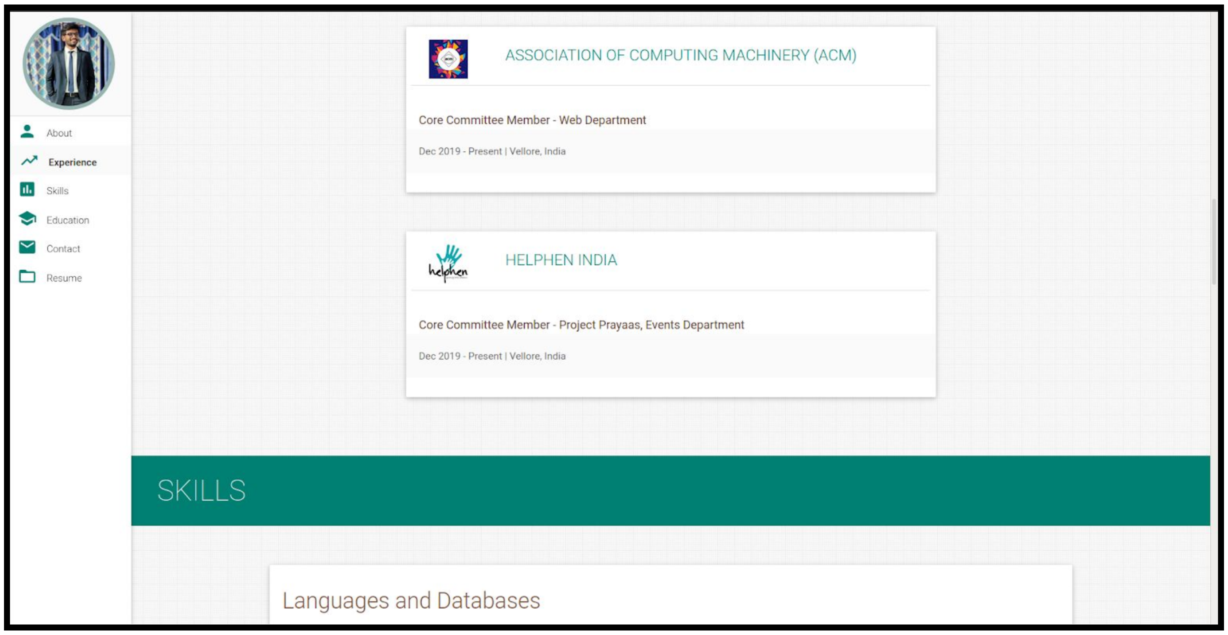
Looking for an opportunity to work in a challenging position combining my skills in Software Engineering, which provides professional development, interesting experiences and personal growth.

EXPERIENCE

COMPUTER SOCIETY OF INDIA (CSI)

Core Committee Member - Technical Team

Dec 2019 - Present | Vellore, India



ASSOCIATION OF COMPUTING MACHINERY (ACM)

Core Committee Member - Web Department

Dec 2019 - Present | Vellore, India

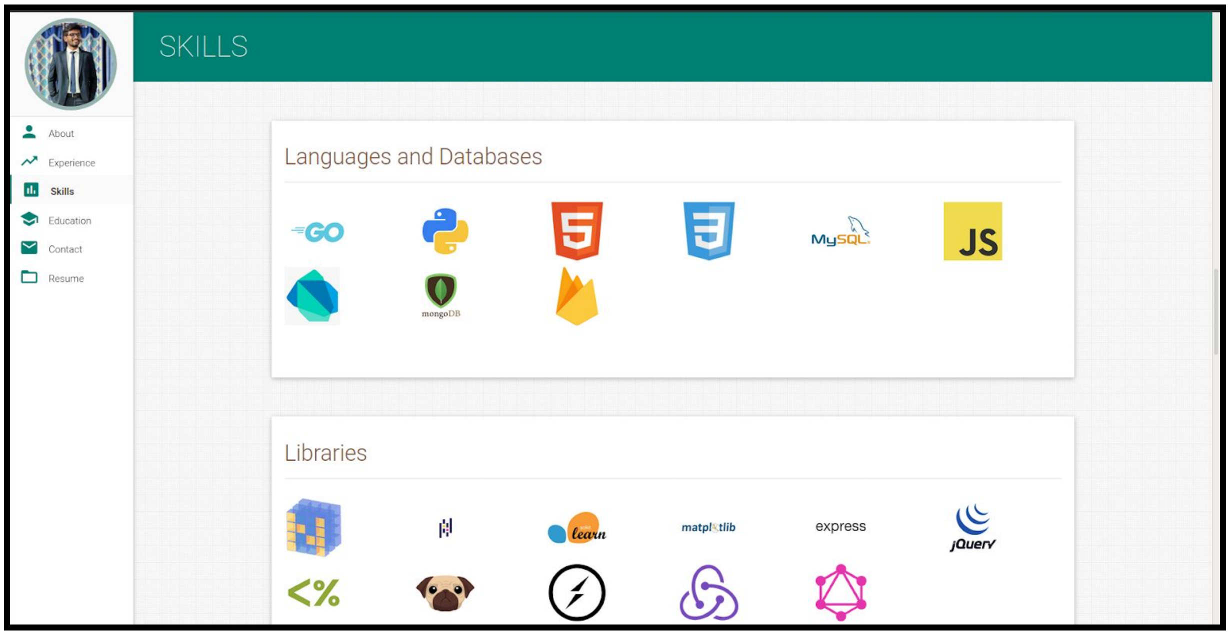
HELPHEN INDIA

Core Committee Member - Project Prayaas, Events Department

Dec 2019 - Present | Vellore, India

SKILLS

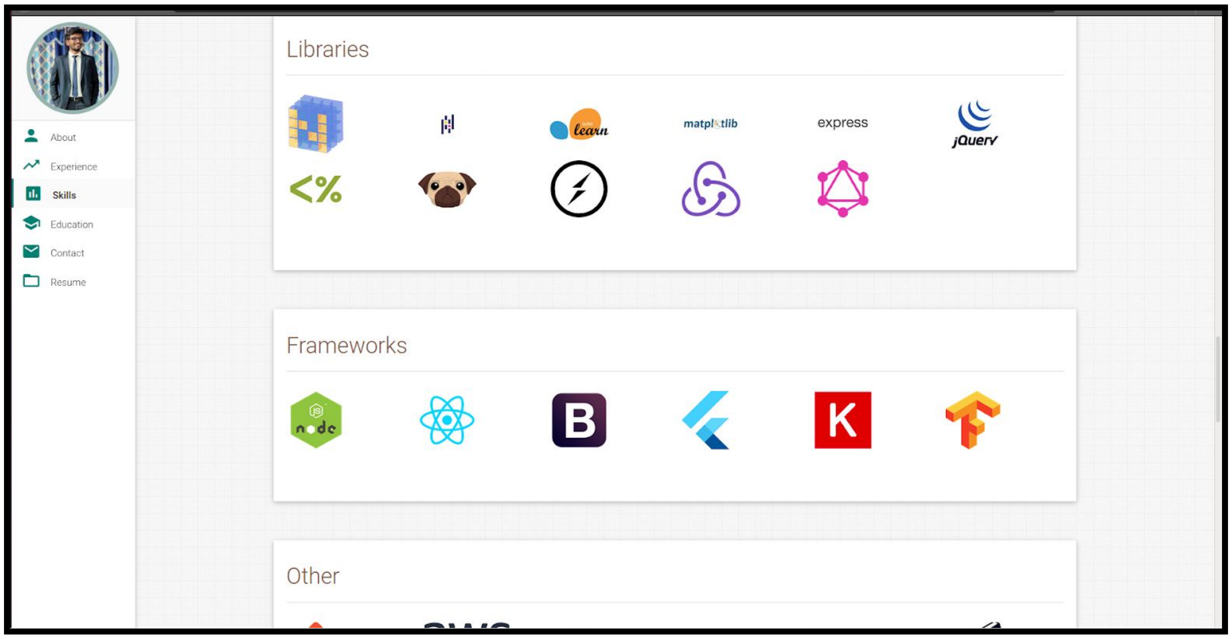
Languages and Databases



SKILLS

Languages and Databases

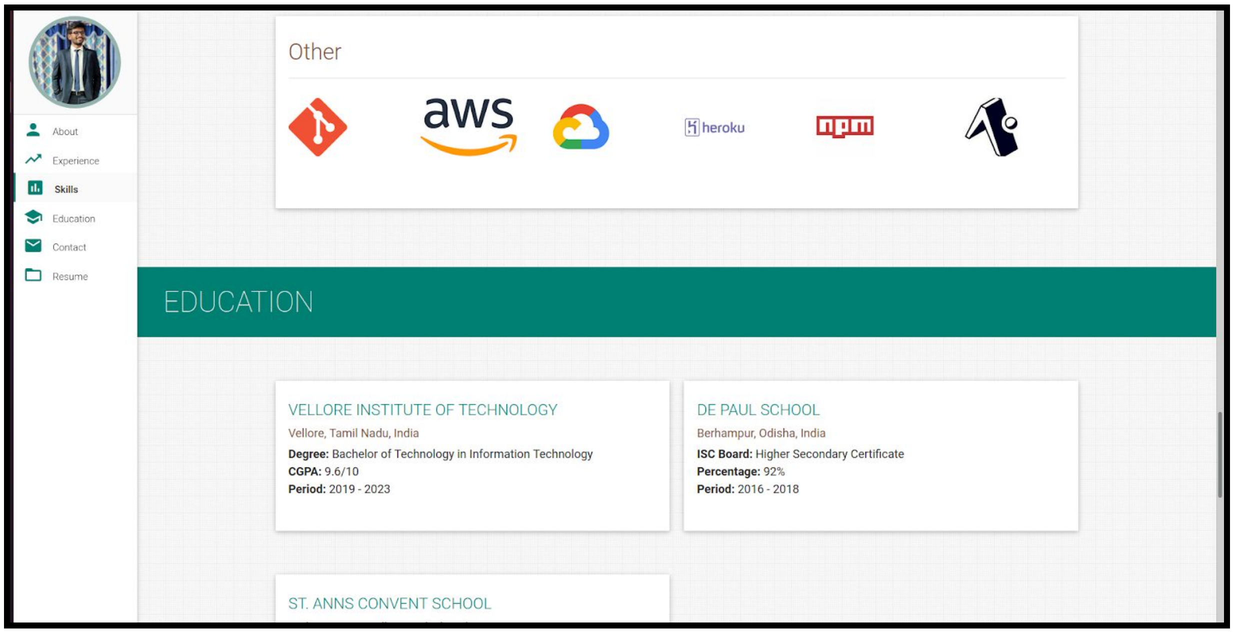
Libraries



Libraries

Frameworks

Other



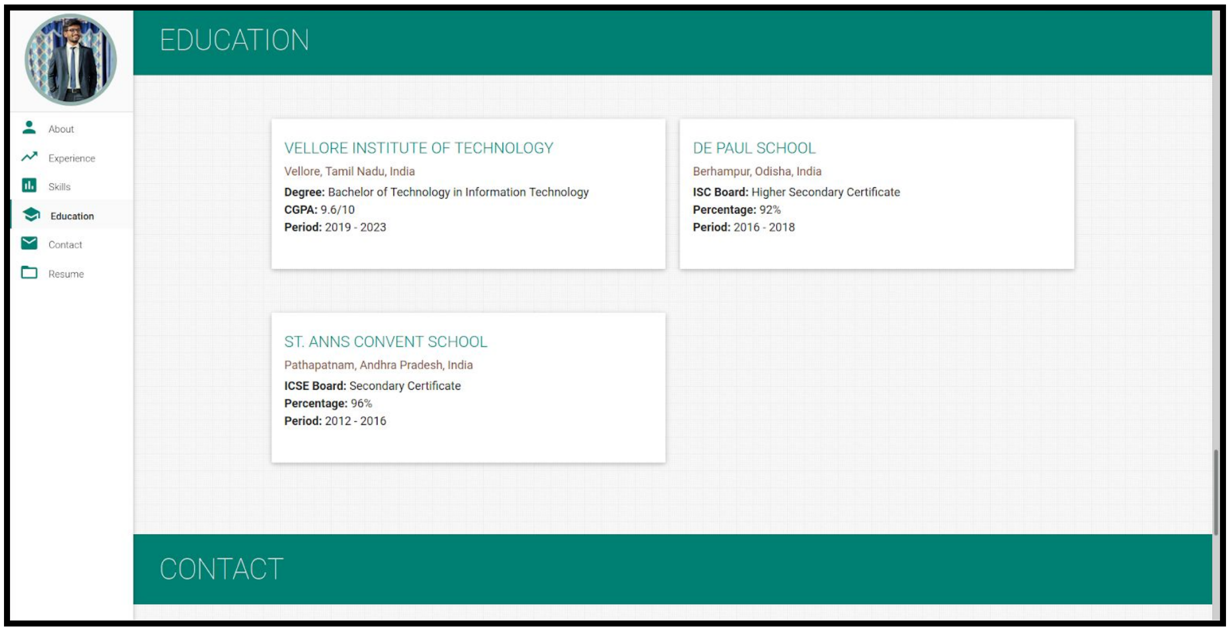
Other

EDUCATION

VELLORE INSTITUTE OF TECHNOLOGY
Vellore, Tamil Nadu, India
Degree: Bachelor of Technology in Information Technology
CGPA: 9.6/10
Period: 2019 - 2023

DE PAUL SCHOOL
Berhampur, Odisha, India
ISC Board: Higher Secondary Certificate
Percentage: 92%
Period: 2016 - 2018

ST. ANNS CONVENT SCHOOL



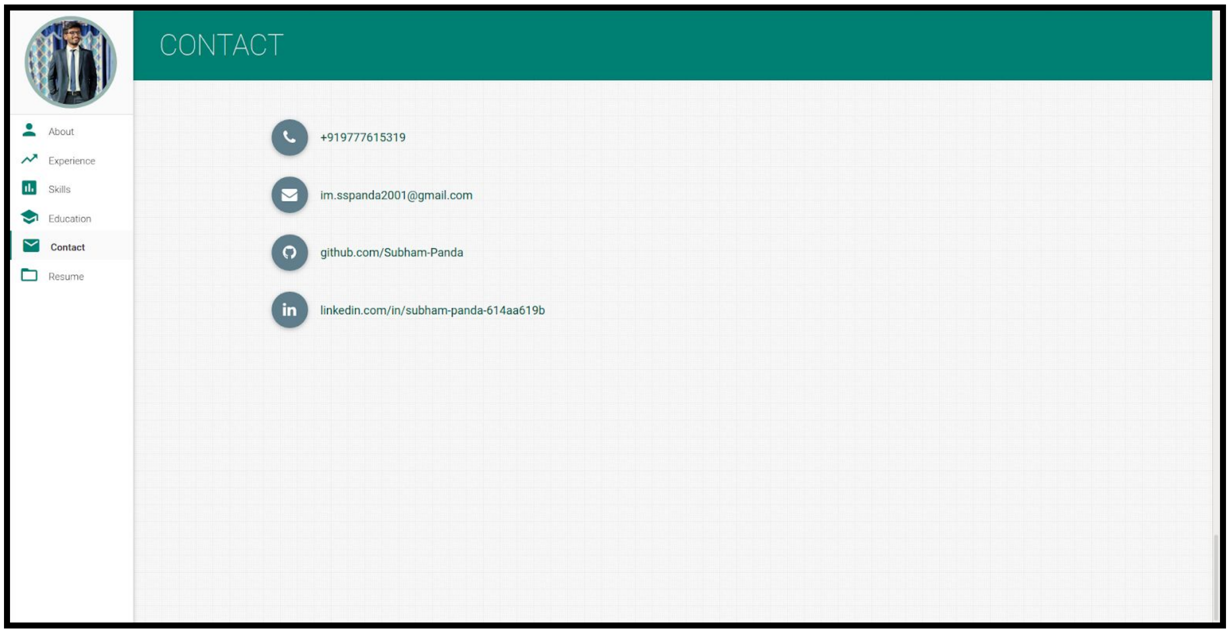
EDUCATION

VELLORE INSTITUTE OF TECHNOLOGY
Vellore, Tamil Nadu, India
Degree: Bachelor of Technology in Information Technology
CGPA: 9.6/10
Period: 2019 - 2023

DE PAUL SCHOOL
Berhampur, Odisha, India
ISC Board: Higher Secondary Certificate
Percentage: 92%
Period: 2016 - 2018

ST. ANNS CONVENT SCHOOL
Pathapatnam, Andhra Pradesh, India
ICSE Board: Secondary Certificate
Percentage: 96%
Period: 2012 - 2016

CONTACT



CONTACT

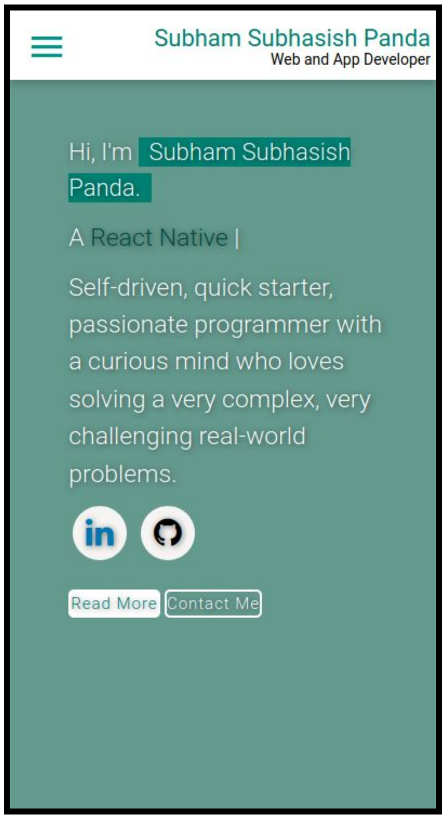
+919777615319

im.sspanda2001@gmail.com

github.com/Subham-Panda

linkedin.com/in/subham-panda-614aa619b

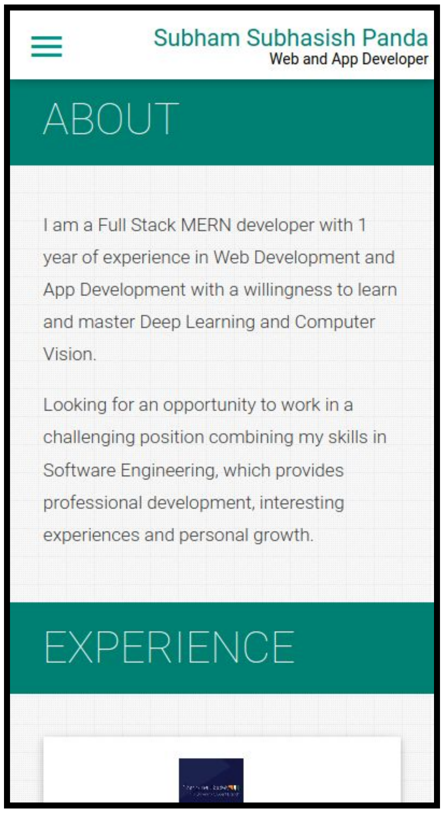
Mobile View:


Subham Subhasish Panda
Web and App Developer

Hi, I'm Subham Subhasish Panda.

A React Native |

Self-driven, quick starter, passionate programmer with a curious mind who loves solving a very complex, very challenging real-world problems.

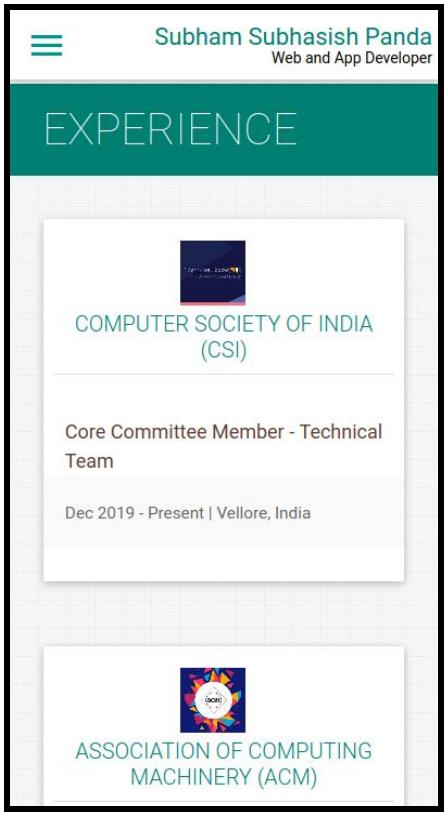

Subham Subhasish Panda
Web and App Developer

ABOUT

I am a Full Stack MERN developer with 1 year of experience in Web Development and App Development with a willingness to learn and master Deep Learning and Computer Vision.

Looking for an opportunity to work in a challenging position combining my skills in Software Engineering, which provides professional development, interesting experiences and personal growth.

EXPERIENCE


Subham Subhasish Panda
Web and App Developer

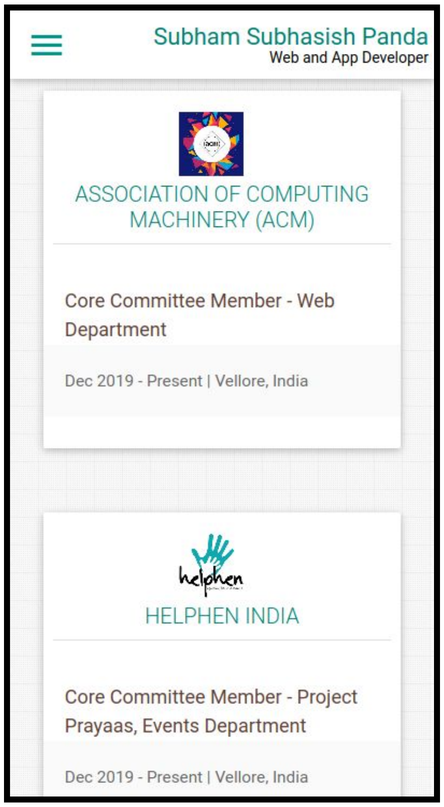
EXPERIENCE

COMPUTER SOCIETY OF INDIA (CSI)

Core Committee Member - Technical Team

Dec 2019 - Present | Vellore, India

ASSOCIATION OF COMPUTING MACHINERY (ACM)


Subham Subhasish Panda
Web and App Developer

ASSOCIATION OF COMPUTING MACHINERY (ACM)

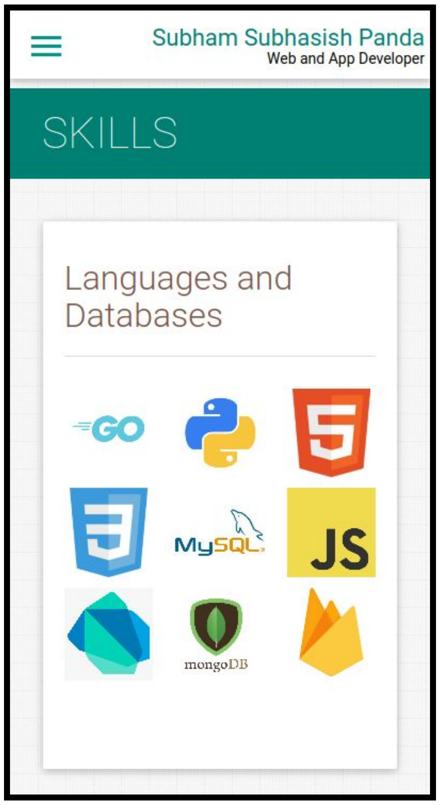
Core Committee Member - Web Department

Dec 2019 - Present | Vellore, India

HELPHEN INDIA

Core Committee Member - Project Prayaas, Events Department

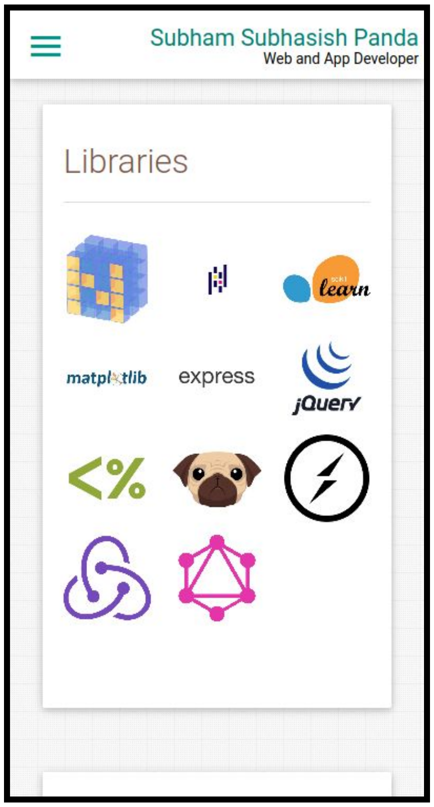
Dec 2019 - Present | Vellore, India



Subham Subhasish Panda
Web and App Developer

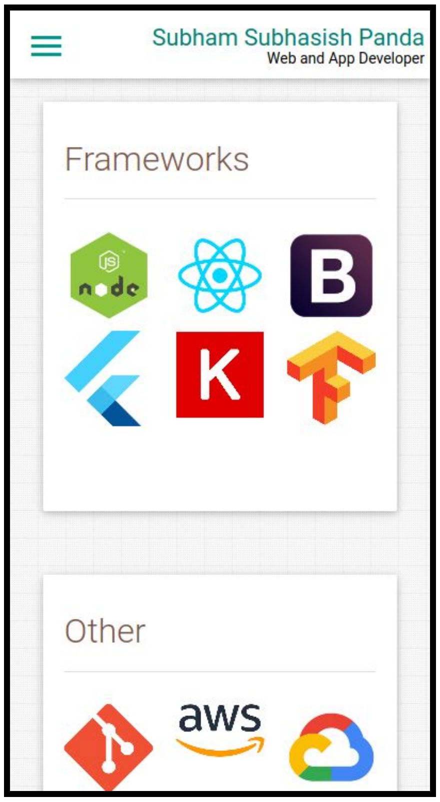
SKILLS

Languages and Databases



Subham Subhasish Panda
Web and App Developer

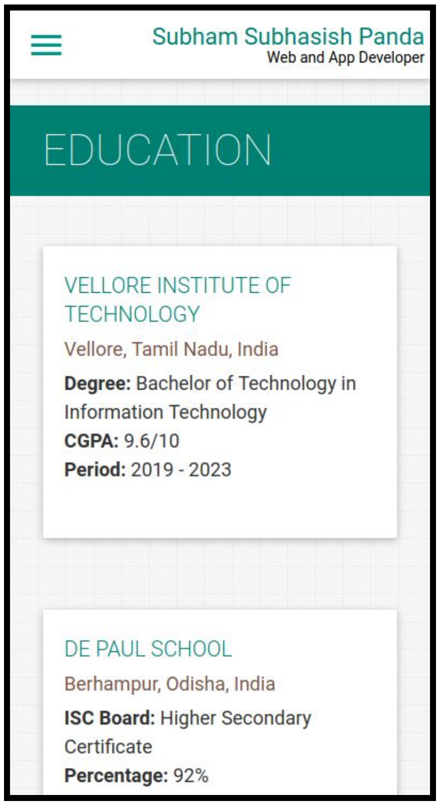
Libraries



Subham Subhasish Panda
Web and App Developer

Frameworks

Other



Subham Subhasish Panda
Web and App Developer

EDUCATION

VELLORE INSTITUTE OF TECHNOLOGY

Vellore, Tamil Nadu, India

Degree: Bachelor of Technology in Information Technology

CGPA: 9.6/10

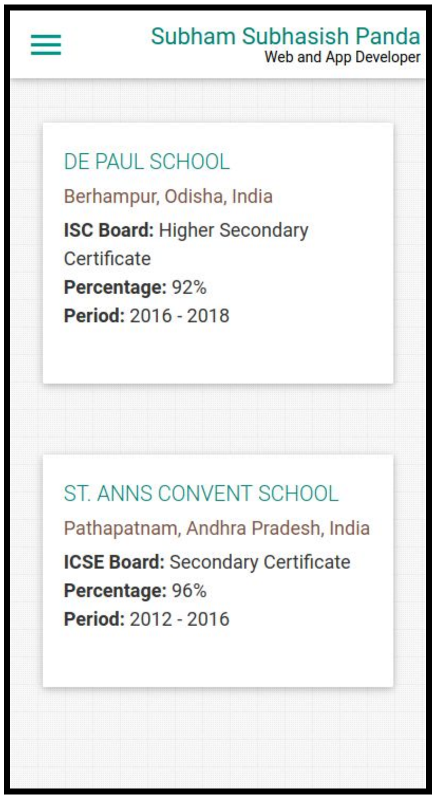
Period: 2019 - 2023

DE PAUL SCHOOL

Berhampur, Odisha, India

ISC Board: Higher Secondary Certificate

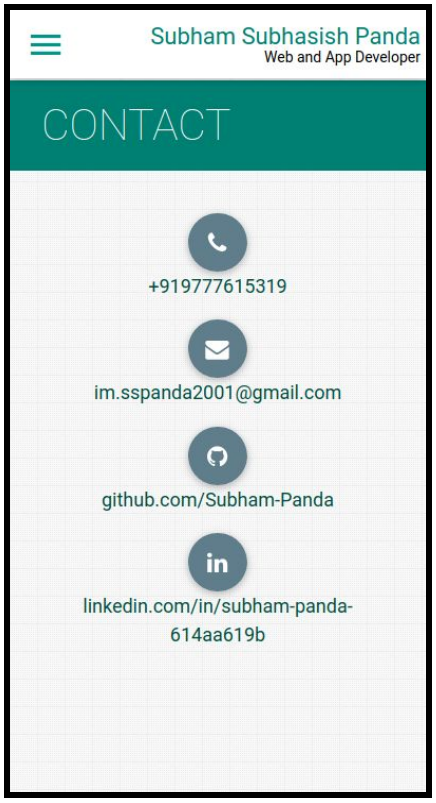
Percentage: 92%



Subham Subhasish Panda
Web and App Developer

DE PAUL SCHOOL
Berhampur, Odisha, India
ISC Board: Higher Secondary Certificate
Percentage: 92%
Period: 2016 - 2018

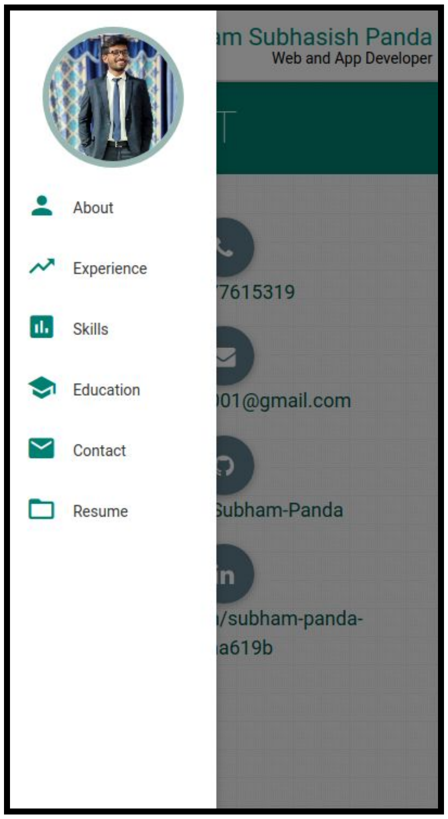
ST. ANNS CONVENT SCHOOL
Pathapatnam, Andhra Pradesh, India
ICSE Board: Secondary Certificate
Percentage: 96%
Period: 2012 - 2016



Subham Subhasish Panda
Web and App Developer

CONTACT

+919777615319
im.sspanda2001@gmail.com
github.com/Subham-Panda
linkedin.com/in/subham-panda-614aa619b



am Subhasish Panda
Web and App Developer
About
Experience
Skills
Education
Contact
Resume
7615319
01@gmail.com
Subham-Panda
/subham-panda-
a619b