



For



Open source

Your first PR *in 7 days.*

Pick a repo, find an issue, open a pull request. Day by day.

Made with engineers who **work at**
Microsoft and Atlassian and **teach at**
NYC.





Before you follow a plan, you should know who made it.

Made with engineers who work at Microsoft and Atlassian *and teach at NYC.*



Swapnil

Forward Deployed Engineer, Microsoft

Works on Microsoft Teams. Teaches Full Stack Engineering at NYC.



Akshay

Senior Software Engineer, Atlassian

Worked on UPI payments at Jio Platforms. Teaches DSA at NYC.

They speak for themselves, not for their employers.



Our students are already *sending pull requests.*

Open source and a hackathon are part of the course, and engineers review the work.



WEEK 1

Git from the first week

FROM WEEK 3

Code goes in through pull requests

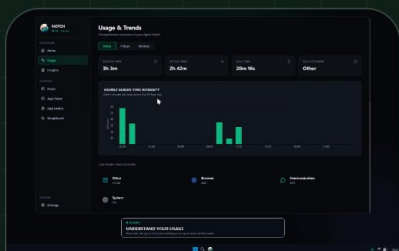
WEEKS 22-23

Open source and a hackathon



What our students *have* built.

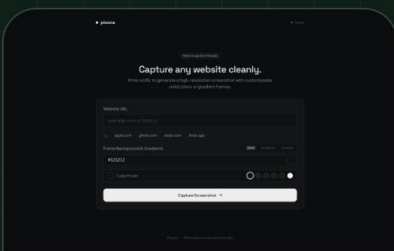
Projects from the NYC community, credited to the people who built them.



Notch

A Windows focus app, built in first year. Akshit.

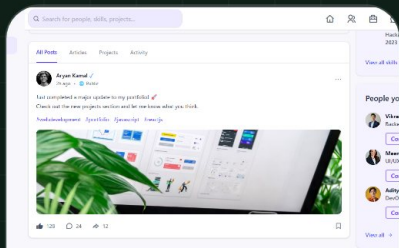
[Open ↗](#)



Pixora

A screenshot studio for any website. Abinash.

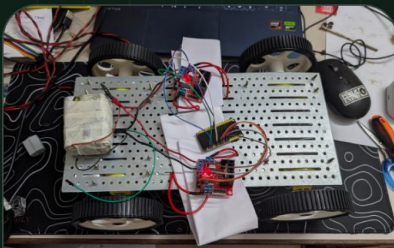
[Open ↗](#)



DevConnect

A network for developers to share projects. Aryan.

[See the code ↗](#)



AgroBot

A farming rover prototype. Akash and team.



His own Vercel, in 10th class

A deployment platform built by a school student.

[Watch the video ↗](#)



Now, let's start *the plan*.

Eight short sections. Read them in order.

01 Why a merged pull request matters

02 Your 7-day plan

03 Pick a repo

04 Make the change

05 Open the pull request

06 The Git commands you need

07 After day 7

08 6 mistakes to avoid

What a merged pull request *shows a recruiter.*



YOUR PULL REQUEST · MERGED

Fix broken install link in README



YOUR CODE

-

+

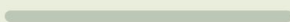
+



YOUR COMMIT HISTORY



HOW YOU TOOK FEEDBACK



Done.

Recruiters can read all three. It is the whole interview, in public.

Why it is worth a week



It is public proof.

Anyone can open the PR and see what you did and how you talked to strangers.



You learn a real codebase.

You learn to read code you did not write. That is 80% of every job.



It leads to GSoC and referrals.

Maintainers remember contributors. Every GSoC selection starts with a PR.

Your 7-day *plan*.

Each day has one job. Do not skip to day 5.

1

Set up

Install Git. Make a GitHub account and an SSH key. Commit once to your own repo.

2

Pick a repo

One you use, with 1k–20k stars and activity in the last month.

3

Find an issue

Look for good first issue, help wanted or docs. Comment: "I would like to take this."

4

Run it locally

Clone, install and run the tests. Read CONTRIBUTING.md twice.

5

Make the change

The smallest possible fix. One file if you can. Tests pass.

6

Open the PR

A clear title, what and why, a screenshot. Link the issue.

7

Respond to review

Fix what they ask within a day. Say thanks. Do not argue in your first PR.

8+

Do it again

A second PR in the same repo. Now you are a contributor, not a visitor.

What it takes

1

merged PR is enough to start. Not ten.

7

days, 1–2 hours a day

0

permission needed. Just open the PR.

Days 2–3: pick a repo *you can finish.*

1

Check the last commit

Within 30 days

If there is nothing newer, the maintainer is gone.

2

Check how fast PRs merge

Days, not months

Open the Pull requests tab and look at the merged ones.

3

Check the tone

3 closed PRs

Read 3 closed PRs. If the reviewers are kind, go.



Day 3: claim an issue

good first issue

help wanted

docs

Comment on it: "I would like to take this."

Where first PRs get merged



Docs and examples repos of big projects, like React, Node and Python libraries.



CLI tools and VS Code extensions.



Indian open-source orgs: FOSS United, Frappe, Chatwoot.



Anything with a "good first issue" label and replies within a week.

Where beginners drown: avoid for PR #1



Linux kernel, LLVM, Chromium.



Repos with no CONTRIBUTING.md.



Issues where 5 people already said "I'll take this".



Rewrites and "refactor everything" issues.

Days 4–5: make the *smallest real change*.

1 Run it locally Day 4

Fork the repo, clone your fork, install it and run the tests. Read CONTRIBUTING.md twice.

2 Make a branch Day 5

```
git checkout -b fix/thing
```

3 Make the change

The smallest possible fix, in one file if you can. Then run the tests.

4 Commit and push

Write a commit message that says why. Push to your fork, ready to open the PR against main.



Checkpoint, day 5

Change made, tests green, branch pushed.

Good first changes: small, useful, mergeable



Fix a typo or a broken link in docs



Add a missing example to a README



Fix a failing test on your OS



Add a clear error message where there was none



Translate one page

Day 6: write a PR *a stranger will merge.*

1

Title: say what changed

Under 60 characters



Fix broken install link in README



update

2

Body: three short lines

What changed

Why

How to test

Add a screenshot if anything is visual. Fill in the PR template.

3

Link the issue

Closes #123

Maintainers triage by issue. No link, no priority.

4

One PR, one change

Do not fix three things at once. Three PRs are easier to review.

5

Reply to every review comment

Within 24 hours

Fix it, push, say "done".

**That is how you get merged and remembered.**

At NYC: you open a real PR, with engineers reviewing it before you open it.

The 15 Git commands *you need.*

START WORK

git clone
git checkout -b
git status

SAVE YOUR WORK

git add -p
git commit -m
git push -u origin

LOOK AROUND

git log --oneline
git diff
git blame

STAY IN SYNC

git fetch upstream
git rebase upstream/main

UNDO

git stash
git reset --soft HEAD~1

GITHUB CLI

gh pr create
gh pr checkout

Sync your fork before every PR

- 1 git fetch upstream
- 2 git checkout main && git rebase upstream/main
- 3 git push

Fix a mistake without panic

- 1 **Wrong commit message**
git commit --amend
- 2 **Wrong branch**
git stash, checkout, stash pop
- 3 **Broke everything**
git reset --hard upstream/main

What to do *after your first PR.*

DAY 8



You are here

Your first PR is done. Now build on it.

MONTH 1

1

3 PRs, one repo

Become a name they recognise. Ask for slightly bigger issues.

MONTH 2

2

Pin it

Pin the repo on GitHub. Put "contributor to X" on LinkedIn and your résumé.

MONTH 3

3

GSoC or a referral

Ask a maintainer about GSoC. Ask a contributor who works at a company for a referral.

One résumé line that works

OPEN SOURCE

Contributor, *project*: fixed X in Y, merged by maintainers ([link](#)).

IN THE INTERVIEW, THEY WILL ASK

What did reviewers push back on?

What did you change?

That answer is worth more than any certificate.



At NYC: open source and a hackathon are part of the course, with engineers reviewing the work.

6 mistakes that *get first PRs closed.*

✕ ~~1. Claiming an issue and vanishing~~

✓ Comment only when you will start today.

✕ ~~2. A huge first PR~~

✓ Nobody reviews 40 files from a stranger. Keep it under 50 lines.

✕ ~~3. Ignoring CONTRIBUTING.md~~

✓ Read it first. It tells you the branch name, the commit format and the test command.

✕ ~~4. Arguing with the review~~

✓ Ask why, then do it. Debate on PR #10, not #1.

✕ ~~5. Going silent after review~~

✓ A PR with unanswered comments gets closed in two weeks. Reply to every comment.

✕ ~~6. Collecting repos, finishing nothing~~

✓ One merged PR beats twenty forks. Finish one before you start the next.

We teach open source *in* live classes.

● ● ● nyc · course

Covers this plan



WE TEACH THIS

Software Engineering + AI

Open source is module 6 of Software Engineering + AI: you open a real PR with engineers reviewing it before you open it.

📺 Live classes

👤 Engineer reviews

🎯 For: **Open-source contributor** → **SDE intern**

📞 Talk to an advisor

See the course → notyourcollege.com



Guide by the NYC team, 2026. Based on public contribution guides of major open-source projects. NYC does not guarantee employment or selection.