



For **AI app**

Build your first AI app *this weekend.*

From a blank folder to a live app with an
AI feature, in one weekend.

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





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

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



Swapnil

Forward Deployed Engineer, Microsoft

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



Abhishek

Engineering Analyst, Google

Earlier a senior data scientist at Optum. Teaches Data Science at NYC.

They speak for themselves, not for their employers.



ABOUT NYC · 2 OF 3

Our students are already *building apps like this.*

On build days, with engineers from Microsoft and Google.



EVERY
FRIDAY

Build, debug or review

WEEKS 18-21

AI features on your own data

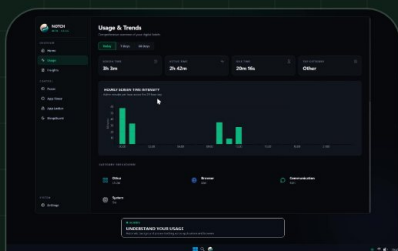
EVERY
PROJECT

An engineer reviews it first



What our students *have* built.

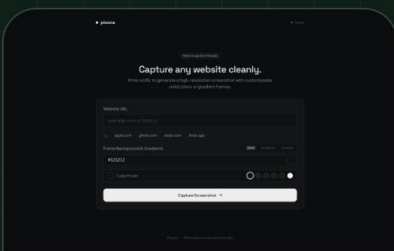
Projects from the NYC community, credited to the people who built them.



Notch

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

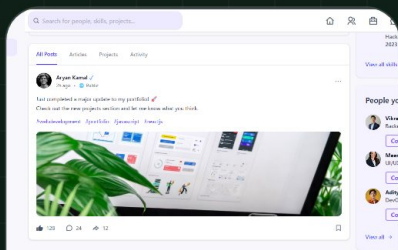
[Open ↗](#)



Pixora

A screenshot studio for any website. Abinash.

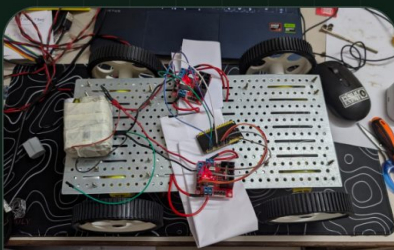
[Open ↗](#)



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 Your weekend plan
- 02 Saturday morning: call an AI model
- 03 Saturday afternoon: add your notes and a page
- 04 Sunday morning: make it honest
- 05 Sunday afternoon: put it online
- 06 9 interview questions about your app
- 07 7 mistakes to avoid
- 08 Weekend two: 3 upgrades

Your weekend *plan.*

A study-notes assistant: upload notes, ask a question, get an answer with the source line. Small and real, and you put it online.

SATURDAY

9am

Set up

Pick Node or Python.
Make one folder. Get an API key.

SATURDAY

11am

First call

Send a prompt and print the answer. That is an AI app.

SATURDAY

2pm

Add your data

Load a notes file. Put it in the prompt. Ask again.

SATURDAY

6pm

Make it a page

One input, one button and one answer box.

SUNDAY

10am

Make it honest

Show the source line. Say "not in notes" when it is not.

SUNDAY

3pm

Deploy and share

Put it online on Vercel or Railway. Send the link to three people.

By Sunday night

15

hours in total: 9 on Saturday, 6 on Sunday

1

feature, not a chatbot that does everything

3

people have tried your app

Saturday morning: *call an AI model.*

Hours 1–3, from 9am to noon.

1

Set up

9am

Make a folder. Run `npm init -y`, or make a Python venv (a virtual environment).

Get an API key from any model provider. Put it in `.env`, never in your code.

Install the provider's SDK (its code library). Read the 10-line quickstart.

2

Make your first call

11am

Send this prompt and print the reply:

Explain recursion in two lines.

Change the prompt. Print again. Then add a system message:

You are a study helper. Be brief.

Notice the temperature (how random the reply is), the max tokens (how long it can be) and the cost per call.



Checkpoint

Noon

A script that takes a question and prints an answer. Commit it.

Two rules before you start



Pick a small model

Use the smallest, cheapest model that works. You can upgrade later.



Set a spend cap

Set a spend limit in the provider dashboard before you run anything in a loop.

Saturday afternoon: *add your notes and a page.*

Hours 4–9. Steps 1–4 are the simplest RAG: find the right part of your notes, then send it to the AI with the question.

1

Load one notes file 2pm

One chapter of your notes, 2,000 words.

2

Split it into paragraphs

Give every paragraph a number.

3

Pick 3 paragraphs for each question

Pick the 3 with the most words that match the question. Yes, plain keyword match.

Embeddings, a search by meaning, can wait.

Maybe tomorrow.

1

2

3

4

5

6

7

8

9

10

4

Send only those paragraphs

Tell the AI:

Answer only from these paragraphs. Cite the number.

5

Make it a page 6pm

A box for the question, a button and an answer box.

Under the answer, show which paragraphs were used.

Add a loading state and an error state. That is the whole page.

Build it with AI. Then read every line it writes.



Checkpoint Saturday night

On your laptop, ask a question from your notes. You get an answer with paragraph numbers.

Sunday morning: *make it honest.*

Hours 10–12. This is what separates a demo from a project you can defend.

AN HONEST APP ANSWERS LIKE THIS

A question from your notes

The answer.

From paragraphs 3, 7

A question that is not in your notes

Not in notes.

1

Ask something not in the notes 10am

If it answers anyway, it is making things up (a hallucination). Fix the prompt:

If the answer is not in the paragraphs, say so.

2

Show the source every time

The answer, then "from paragraphs 3, 7".
Without a source, nobody can trust it.

3

Write 5 test questions with known answers

Run them after every change. This is your evaluation.

4

Handle the failures

An empty question, the API down, a question over the token limit (too long for the model).
Show a message, not a crash.

5

Log every call

Save the question, the paragraphs used, the tokens and the time. You will need this to explain it in an interview.

Sunday afternoon: *put it online and share it.*

Hours 13–15. Deploy it, share it and watch people use it.

1

Deploy on Vercel or Railway 3pm

Push to GitHub and connect it. Set the API key as an environment variable on the host, not in your code. It is live in 10 minutes.

2

Share it with 3 people

A classmateA seniorSomeone outside CS

Watch them use it. Do not explain.

3

Fix what they hit

The first 3 bugs real people find are your Sunday night.

By Sunday night

1

live link, by 6pm
Sunday

3

people tried it. Their
confusion is your
to-do list.

1

README: what it
does, how it works,
a screenshot, the
link

9 interview questions *about your app.*

If you can answer these out loud, the app is a project. If not, it is a demo.

THEY WILL ASK

Why this model?

What is in the prompt, exactly?

How do you pick the paragraphs?

What happens when the answer is not in the notes?

How do you know it works?

Your 5 test questions

What does one question cost?

What breaks at 1,000 users?

Where did AI write code you had to fix?

What would you add next? Embeddings?
Login?

7 mistakes to avoid *in* *your first AI app.*

✗ ~~1. Building a general chatbot~~

✓ Build one feature on your own data. It beats "ChatGPT but worse".

✗ ~~2. Putting the API key in your code~~

✓ Use environment variables only. A key in code is on GitHub in an hour and billed by Monday.

✗ ~~3. Starting with embeddings and vector databases~~

✓ Start with keyword match. Upgrade when you can explain why.

✗ ~~4. No tests~~

✓ Write 5 questions with known answers. Run them after every change.

✗ ~~5. Trusting the answer~~

✓ Make it cite. Make it say "not in notes". Then check it.

✗ ~~6. Not deploying~~

✓ Put it online. An app that runs only on your laptop is not a project. A link is.

✗ ~~7. Pasting AI code you cannot read~~

✓ Use AI to build. Then read every line. The interviewer will.

Weekend two: *3 upgrades to your app.*

1

Embeddings

Replace keyword match with a vector search, which finds paragraphs by meaning. Compare the answers on your 5 tests.

2

Upload

Let users upload their own notes. Add login, so each user has their own data.

3

Tools

Let it call one function, like "add to flashcards". Ask the user to confirm first.

Then put it on your résumé

RÉSUMÉ · PROJECTS

Built a notes assistant with cited answers; 5-question eval, 92% correct; deployed on Vercel; 40 users in week one.

An example line. Use your own numbers.



Keep a log: every decision, every bug, every fix.

That log is your interview.



At NYC: LLM APIs, retrieval and evaluations are taught live in class.

We teach all of this, *live*.

● ● ● nyc · course

Covers this plan



WE TEACH THIS

AI Engineering

AI Engineering: LLM applications, retrieval, agents and evaluation, live. You build a document assistant and a tool-using assistant, reviewed by engineers.



Live classes



Engineer reviews

For: **AI engineer intern**

Talk to an advisor

See the course → notyourcollege.com



Guide by the NYC team, 2026. Model providers and hosting platforms named are independent of NYC. NYC does not guarantee employment or selection.