Senior Design Lab 3

GitHub Projects • Discovery & Research • GenAI for Development

September 9, 2026

Required tonight: every team leaves with a working GitHub Project and a first two-week plan.

Tonight’s outcomes

- Create one GitHub Project for the team.
- Set up a simple Agile/Scrum-style workflow.
- Create a two-week plan with owners and due dates.
- Connect project discovery, writing, presentations, and coding to the same plan.
- Practice one small GenAI-assisted technical task.

Tonight's plan

6:10–8:30 PM

6:10–6:25	Project management: what we need
6:25–6:45	GitHub Projects: setup + example
6:45–7:30	Hands-on: create your team Project
7:30–7:45	Discovery & research → tasks
7:45–8:00	GenAI for development + career value
8:00–8:20	Hands-on: small technical experiment
8:20–8:30	Team self-check + instructor/TA spot-check

Where we are now

- Teams are formed and each team has a private repository.
- Project ideas are still being refined.
- The first draft project proposal is coming soon.
- From this point forward, the course includes technical work, writing, presentations, and project communication.

Today we add one place to plan and track all of that work: GitHub Projects.

Part 1 — GitHub Projects

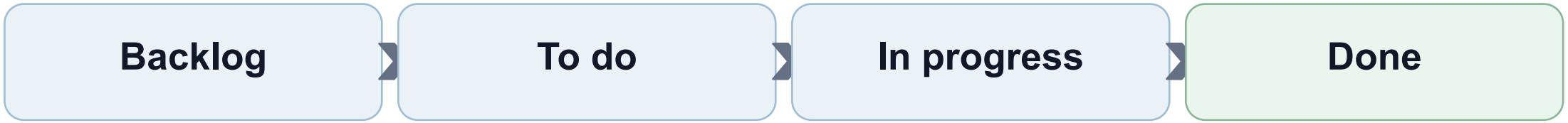
Lightweight Agile/Scrum for a year-long team project

Why use a project board?

- A team project fails when work is invisible or ownership is unclear.
- The board gives the team one shared view of what is next, who owns it, and what is finished.
- It also gives instructors and mentors a quick way to see progress.

The board is not paperwork. It should make the project easier to run.

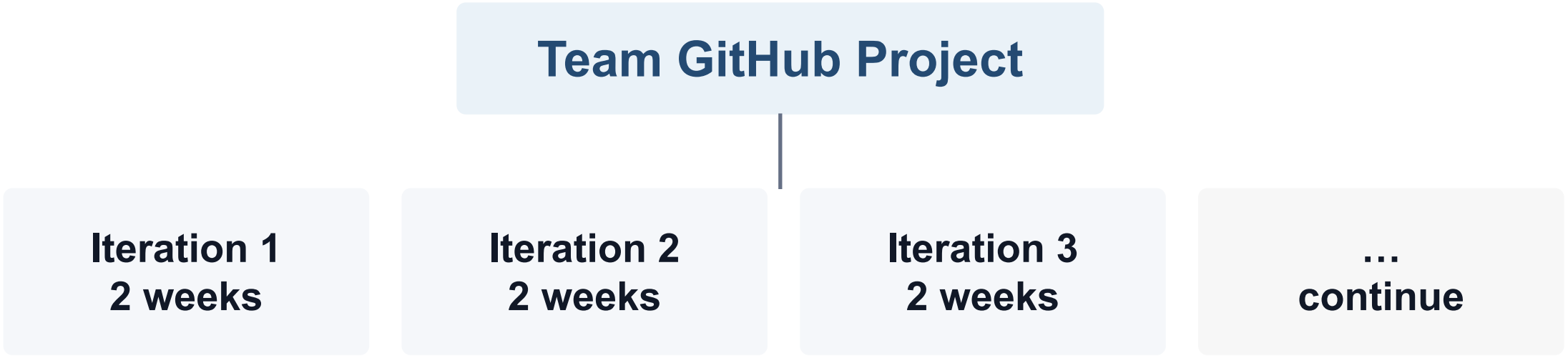
We will use a lightweight Scrum-style process



- Plan work in two-week iterations.
- Each task has an owner and a due date.
- Move tasks as the work changes.
- At the end of the iteration: finish it, move it, or drop it.

No story points, velocity calculations, or formal Scrum roles are required.

One Project for the whole year



- Use GitHub’s Iteration field for the two-week blocks.
- Keep the same Project as the project grows.
- Create views or filters later only if they help the team.

Minimum information for a task

Title	What is the work?
Owner	Who is responsible?
Due date	When should it be done?
Iteration	Which two-week block?
Status	Backlog / To do / In progress / Done

Optional later: priority, labels, estimates, automation.

Example: first two-week plan

These are examples. Your team should edit them.

Task	Owner	Due	Status
Compare two existing solutions	Student A	Sep 15	To do
Try one public API	Student B	Sep 16	To do
Sketch a simple UI flow	Student C	Sep 17	To do
Draft a problem statement	Student D	Sep 18	To do
Draft project proposal outline	Team	Sep 20	To do

Minimum tonight: at least 1 task per member + 1 team-level task.

Write tasks that can actually be checked

Too vague	Better
“Work on backend”	“Create a GET /health endpoint and verify HTTP 200”
“Research competitors”	“Compare 3 existing products and list 2 gaps”
“Do presentation”	“Draft 3-slide proposal and assign sections to team”

GitHub Projects: setup steps

- Open your team repository and go to Projects.
- Create or link one Project for the team.
- Use a Board view with: Backlog, To do, In progress, Done.
- Add an Iteration field and make the first iteration two weeks.
- Add a Due date field.
- Create your first tasks and assign owners.

If you do not see “New project” or cannot create fields, ask the TA/instructor. It may be a permission issue.

Hands-on — 45 minutes

- Create the team Project.
- Create the first two-week iteration.
- Add at least one test task for each team member.
- Add at least one team-level task.
- Set owner, due date, iteration, and status.
- Move one test task from To do → In progress → Done so everyone knows how.

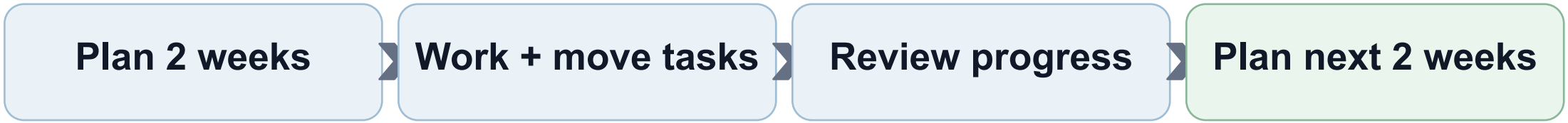
Do not worry about perfect task details tonight. The point is to learn the workflow.

Team check before we continue

- One Project exists and all team members can access it.
- The workflow columns are visible.
- A two-week iteration exists.
- Every member has at least one task.
- There is at least one team task.
- Tasks have owners and due dates.

If your team is done early: refine the tasks or help a neighboring team.

Our progress rhythm



- Every two weeks: update the board and discuss progress.
- End of September, October, and November: graded progress checkpoint.
- Formal writing and presentation deadlines still follow the course schedule.

Part 2 — Discovery & Research

What are you building, for whom, and why?

Discovery: answer four questions

1. What problem are we solving?
2. Who has this problem?
3. What solutions already exist?
4. What evidence would tell us our idea is useful?

September goal: be able to explain what you are building and why.

Turn questions into work

Question	Possible task
What problem?	Write a 3–5 sentence problem statement
Who are the users?	Interview 2 potential users or review user evidence
What exists?	Compare 3 existing products / papers / tools
Can we build it?	Try one API, dataset, model, or technical component

Discovery is not separate from project management. Put the work on the board.

The board is not only for coding

- Research and discovery tasks
- Coding and technical experiments
- Writing assignments and revisions
- Presentation preparation and practice
- Testing, documentation, and mentor follow-up

If it takes meaningful project time, it can be a task.

Part 3 — GenAI for Development

Use it to move faster — but understand what you ship

Why we are talking about GenAI

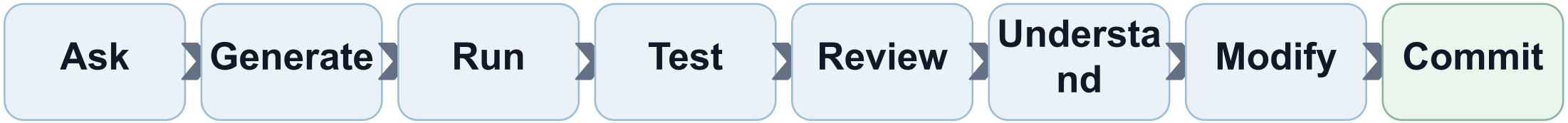
- AI-assisted development is becoming part of normal software work.
- LinkedIn reports U.S. job postings requiring AI-literacy skills grew 70% year over year.
- For job searches and graduate applications, this project should show both technical depth and your ability to use modern tools responsibly.

Source: LinkedIn Economic Graph, 2026 Labor Market Report

Course approach to GenAI

- We encourage you to use GenAI tools for development.
- Share useful prompts, workflows, and lessons with your teammates.
- Document the major tools your team uses.
- You are still responsible for correctness, security, testing, and understanding the code.
- You should be able to explain and modify anything you submit.

A practical workflow



Do not stop at “Generate.” The engineering work starts when you run and test it.

Good use: small, testable tasks with clear inputs and expected results.

Give the AI a small engineering task

Weak request

“Build our whole app.”

Better request

“In Python, call this public REST API, check the HTTP response, parse JSON, and print two fields. Keep the example under 15 lines.”

- Give context.
- State the result you want.
- Add constraints.
- Ask how to test the result.

Tech labs in 2026

- You may still need React, FastAPI, Node, PyTorch, mobile, IoT, databases, or another technology.
- The difference: use documentation + GenAI to get to a small working example faster.
- Do not spend the whole lab passively following a long tutorial.
- Build one small thing, run it, verify it, and understand it.

Choose experiments that are relevant to your team’s likely project.

Worked example — read a public REST API

- Goal: read public GitHub user data from Python.
- Endpoint: https://api.github.com/users/octocat
- This example does not require an API key for a small public request.
- General skill: HTTP GET → check response → parse JSON → use selected fields.

First: open the endpoint in a browser. You should see JSON.

Worked example — Python

```
import requests

url = "https://api.github.com/users/octocat"
r = requests.get(url, timeout=10)
r.raise_for_status()
data = r.json()

print(data["login"])
print(data["html_url"])
```

If needed: pip install requests

Worked example — expected result

Expected output

```
octocat
https://github.com/octocat
```

What each line did

- GET: request data from a URL.
- raise_for_status(): stop on an HTTP error.
- json(): convert the response into Python data.
- Select only the fields you need.

Verification: compare the program output with the JSON shown in your browser.

Hands-on GenAI tech exercise — 20 minutes

- Pick one small technical question that may matter to your project.
- Use official documentation and a GenAI tool together.
- Get a small example running.
- Test it with a known input or a known-good example.
- Ask the AI to explain anything you do not understand.
- Optional: commit useful code to the team repository.

If you are unsure what to try: repeat the GitHub API example, then change it to a public username of your choice.

Possible small experiments

Web	Render one React component with sample data
Backend	Create one API endpoint and test the response
Data/API	Call a public API and parse useful fields
ML	Load a small dataset and run one baseline model
LLM	Call an LLM API with one test prompt and inspect the result
IoT	Read one sensor value or run one hardware example

Small + working + understood is better than large + generated + untested.

Part 4 — Make the Project Useful for Your Career

Build something you can explain, show, and defend

By spring, your project should give you evidence

- A working demo or prototype
- A repository that shows sustained contribution
- A clear README and project documentation
- Design decisions you can explain
- Tests, evaluation results, or user evidence
- A clear description of your own contribution

The goal is not “I used ChatGPT.” The goal is “I built and validated this system.”

A useful résumé bullet has four parts

What you built → **Your contribution** → **Tools / methods** → **Result / evidence**

Example structure

Built [system] with a 4-person team; implemented [your component] using [tools]; validated it with [test / metric / users / demo result].

Use real numbers and outcomes when you have them. Do not invent metrics.

How to talk about the project

Job interview

- What problem did you solve?
- What did you personally build?
- What tradeoff or bug did you handle?
- How did you test it?

Graduate school

- What question or need motivated the work?
- What method did you use?
- What evidence did you collect?
- What would you investigate next?

Good project management now makes these answers easier later.

Optional resources

Use these later if useful.

GitHub Projects quickstart	https://docs.github.com/en/issues/planning-and-tracking-with-projects/learning-about-projects/quickstart-for-projects
GitHub Projects iteration fields	https://docs.github.com/en/issues/planning-and-tracking-with-projects/understanding-fields/about-iteration-fields
Learn to code with GitHub Copilot	https://docs.github.com/en/get-started/learning-to-code
Use your GitHub profile to enhance your résumé	https://docs.github.com/en/account-and-profile/tutorials/using-your-github-profile-to-enhance-your-resume
LinkedIn 2026 Labor Market Report	https://economicgraph.linkedin.com/research/labor-market-report-2026
Course website	https://gw-cs-sd-26-27.github.io/

Optional: GitHub Project extras

Skip this slide if time is short.

- Create a “Current iteration” view.
- Filter by assignee when you want to see one person’s work.
- Add Priority only if the team will actually use it.
- Use automation later to move or archive items automatically.
- Keep the board simple enough that the team will maintain it.

Before you leave tonight

- One team GitHub Project exists.
- The first two-week iteration is set up.
- Every member has at least one task.
- There is at least one team-level task.
- Tasks have owner, due date, iteration, and status.
- Everyone knows how to move a task across the board.
- You tried or started one small GenAI-assisted technical experiment.

Final 10 minutes: team self-check, then instructor/TA spot-check.