Final Project

For the final project, you prototype a product of your own that aligns with the College's vision of STEM for a Better World. You work on your own, with a small group of classmates (your pod) who give you feedback along the way.

AI plays two parts in your project. It helps you **build** the product, the way Claude Code built Lab 7. And it's part of what the product **does**: your app calls a large language model (LLM) because the people using it need something only a model can do.

Your grade has two layers. The **requirements** are the floor: a working product that meets each one. The **objectives** are where most of the grade is: how you designed it, how you built it with AI, and what you think it all means.

Requirements

Each one is checked on a simple scale: met, partly met, or missing.

#	Requirement	What counts	How we check
1	**It's for someone**	A real problem, for a real person or group other than you, whom you've talked to	Your process log: who, and what they told you
2	**A frontend**	A web page that a person uses to do the main task. Lab 6's standard holds: semantic, accessible, honest about the network, usable at phone width	You demo it on a phone
3	**A database**	Data that your app writes and reads back, and that survives a redeploy	You show us data that was there before your last deploy
4	**Publicly deployed**	A live `https://` web address (URL) that works from a phone on cell data, with secrets in environment variables and never in git	We open it on our own phones
5	**It calls an LLM**	From your server, on a user's action, doing something the product needs: take the model out and the product loses its point. When the model fails, the page says so	You show us the call in your code, and what happens when it fails
6	**It uses a new API**	A third-party web application programming interface (API) that you haven't used in this course, called from your server, with its key in the environment. The LLM doesn't count	You show us the call, and what happens when it fails
7	**Built with AI**	An AI coding agent writes code, in a repo that shows how: a `CLAUDE.md` from the start, and work merged through pull requests (PRs) that you reviewed	Your repo's history

A few notes:

- **Stack.** The course stack (FastAPI, SQLModel, Postgres, a plain HTML, CSS and JavaScript page, Docker on Railway) is the default, and it's the one we can help you debug. You're also welcome to explore something else, as long as you can explain every layer of it without AI.
- **Model calls cost money.** Your Glassroom key covers them. We also have Google Cloud credits, for Google's models or other Google Cloud services: ask us if you'd like some. Either way, call the model only when a person asks for something, and never on page load.
- **A new API** can be a data source (transit, weather, maps, a museum's collection, a public health dataset) or a service (payments, email, speech). Pick one with a free tier, and check its rate limits before you build around it.

Objectives

1. Design process

Design thinking has five stages: **Empathize, Define, Ideate, Prototype, Test**. They aren't a line you walk once. They're a loop, and a good project goes round it more than once: a test teaches you something, and you go back and change the design.

Show your work at each stage:

- **Empathize.** Talk to at least three people who have the problem. What do they do now, and what's hard about it?
- **Define.** Write a point-of-view statement: "*[user]* needs *[need]* because *[insight]*." Rewrite it when you learn something that changes it.
- **Ideate.** Generate many ideas before you pick one. Keep them all, and say how you narrowed them down.
- **Prototype.** Build more than one prototype, each a little more real than the last. Start with one that isn't code: a paper sketch, a wireframe, or a clickable mockup.
- **Test.** Put each prototype in front of people who aren't you. Write down what they did, not just what they said, and what you changed because of it.

Your process log

Record all of it in a **process log**, `PROCESS.md` at the root of your repo. Add an entry each time you work on the project: what you did, what you learned, and what you changed because of it. Photos of sketches and screenshots of prototypes go in a `process/` folder, linked from the entry. We grade the log on whether we can see the loop: tests that changed the design.

Your git history already records *what* changed. The log is where the *why* goes, and Claude can do the bookkeeping while you do the thinking. We've written a `/reflect` skill for Claude Code that does it. From your repo root:

```
curl -fLO https://course.aiprototypes.org/resources/files/eng190bo-fa26-lecture-15-reflect.zip
python -m zipfile -e eng190bo-fa26-lecture-15-reflect.zip .
rm eng190bo-fa26-lecture-15-reflect.zip
git add .claude/skills/reflect && git commit -m "Add the reflect skill" && git push
```

Then, whenever you've done something worth recording:

1. Type `/reflect` in Claude Code, and tell it what you did, who you talked to, what you learned, and what you changed, in your own words. It's quicker to say than to type: in a codespace, use your computer's dictation (press `Fn` twice on a Mac, or Windows key + `H` on Windows). Claude Code's own `/voice` needs a microphone on the machine running it, so it works only if you run Claude Code on your own computer.
2. It reads every commit, merged PR, `CLAUDE.md` change and new image since your last entry, and asks about anything you left out, such as a test with nothing that changed because of it.
3. It drafts an entry from your words, with your commits and PRs as the evidence, and shows it to you. Edit it until it's right.
4. It adds the entry to `PROCESS.md` on a branch, and opens a PR. Read it, and merge it.

The skill writes down what you said. It never invents an interview, a test or a lesson, so an entry is only as good as what you tell it. Because we wrote it, it doesn't count as your harness customization, but changing it to do something new can.

2. AI workflow

Answer these questions, with evidence from your repo:

- **Augmenting your work.** Where did AI make you faster or better? Where did you take the keyboard back, and why?

- **Version control.** Does your git history tell the story of the project? Small commits, a branch per slice, and PRs you actually read.
- **Managing context.** How did you keep the agent focused? A `CLAUDE.md` that grew from mistakes, one task per session, plan mode, subagents.
- **Customizing the harness.** At least one skill, hook, subagent or Model Context Protocol (MCP) server that you built for this project. What does it do, and what went wrong before you had it?
- **Tools.** Which tools and models did you use, for what, and what would you change next time?

You should be able to explain every part of your code without AI. At a design review, we'll pick a piece and ask you to walk us through it.

3. Personal reflection on the societal impact of AI

- **Two systems maps.** A *personal* one: how AI has changed the way you learn and work, including your skills, your habits, and what you depend on. A *societal* one: your product in the world, including who benefits, who bears the costs, and what happens to energy use, jobs, privacy, fairness and misuse.
- **A cost-benefit analysis of your product's use of AI.** What one use of your app actually costs, measured in tokens and dollars, and an estimate of the energy. What it gives the people who use it, what could go wrong, and who bears each cost and each benefit. Compare it with a version that doesn't use an LLM. Is the model worth it here?
- **A short reflection** (one to two pages) that ties the two maps and the analysis to your own choices as an engineer.

What you hand in

Due dates are on Canvas.

Deliverable	What it is
Proposal	One to two pages: the problem and who has it, a first point-of-view statement, what the LLM does, the new API you plan to use, how you'll reach the people you'll talk to, and your biggest risk
Prototype presentation	Your problem, what you learned from the people you talked to, and your prototypes so far
Design review	A working build in front of your pod and an instructor: what's working, what testing changed, a code walk-through, and what's next
Demo Day	Your product, live, to the class
Final submission	Your live URL, your repo and the exact commit, `PROCESS.md`, your AI workflow write-up, both systems maps, your cost-benefit analysis, and your reflection

How it's graded

Part	Share of the project grade
Requirements	20%
Design process	25%
AI workflow	20%
Societal impact	15%
Presentations (prototype presentation, design review, Demo Day) and feedback to your pod	20%

Every part has a rubric in the same shape: a row for each thing we look at, and a column for each level. The requirements are scored on three levels, and everything else on four. Each row is scored on its own, and for the design process, your log is the evidence for every row.

Requirements

	Met	Partly met	Missing
It's for someone	A real person or group other than you, whom you've talked to, and whose problem the product addresses	A real group, but you haven't talked to anyone in it, or the product has drifted from their problem	The only user is you, or a hypothetical one
A frontend	A page a person uses for the main task, meeting Lab 6's standard on a phone	The page works, but misses part of Lab 6's standard: unusable at phone width, or silent when the network fails	No page, or the main task needs `curl` or `/docs`
A database	Data your app writes and reads back, and that survives a redeploy	A database your app only reads, or one that resets on redeploy	No database
Publicly deployed	A live `https://` URL that works from a phone on cell data, with secrets only in the environment	Deployed, but often down or broken in production, or a secret in your git history	It runs only in your codespace
It calls an LLM	From your server, on a user's action, doing a job the product needs, with an honest message when it fails	The call works, but the product doesn't need it, or a failure breaks the page	No model call
It uses a new API	A third-party API you haven't used in this course, called from your server, with its key in the environment and its failures handled	It works, but its key is exposed in the browser, or a failure breaks the page	No new API
Built with AI	A `CLAUDE.md` from the start, and work merged through PRs you reviewed	Built with an agent, but `CLAUDE.md` came late or PRs were merged unread	No evidence of an agent in the repo

Design process

	Exemplary	Proficient	Developing	Beginning
Empathize	Three or more conversations, including someone with the problem at its worst; you watched what people do, not just what they say	Three conversations with people who have the problem, written up	Fewer than three, or only with friends who don't have the problem	No conversations
Define	A point-of-view statement grounded in something you heard, rewritten as you learned more, with the reason for each rewrite	A specific statement whose insight comes from your conversations	A statement whose user is vague ("students") or whose need is a feature	No statement
Ideate	Many different ideas, kept, with a clear account of how and why you narrowed them	Several real alternatives, and why you chose one	One idea, with a few variations	Only the idea you started with
Prototype	Several prototypes, each answering a question the last one raised, starting with one that isn't code	At least two, at different levels of fidelity, starting with one that isn't code	One prototype, or only code	No prototype before the product
Test	Each prototype tested with people who aren't you, recording what they did, and what you changed because of it	At least two rounds of testing with other people, and at least one change that came from them	Testing only with classmates or yourself, or tests that changed nothing	No testing
The loop	Several loops: a test sends you back to an earlier stage, and the product is different because of it	At least one full loop that changed the design	Stages done once, in order	A product with little evidence of how it got there
The log	Regular entries in your own voice, each with evidence (commits, PRs, photos), from the first idea to Demo Day	Entries for each stage, with evidence	Sparse, written after the fact, or with no evidence	Missing or nearly empty

AI workflow

	Exemplary	Proficient	Developing	Beginning
Augmenting your work	Specific moments where AI made you faster or better, and where you took the keyboard back, and why	Specific examples of both	General claims ("it saved time")	No account
Version control	A history that tells the story: small commits, a branch and a PR per slice, and review comments that caught something	Branches and reviewed PRs, with mostly small commits	PRs with no review, or a few huge commits	Everything on `main` in a handful of commits
Managing context	A `CLAUDE.md` that grew from specific mistakes, and deliberate habits (one task per session, plan mode, subagents) you can explain	A `CLAUDE.md` that grew, and habits you can name	A `CLAUDE.md` written once and never changed	No `CLAUDE.md`
Customizing the harness	A skill, hook, subagent or MCP server that fixed a real, recurring problem, with the before and the after	One customization built for this project, which you use	One copied in, or never used	None
Tools	Which tools and models you used, for what, why, and what you'd change	Which tools and models, and what each was for	A list with no reasons	No account
Explaining the code, without AI	Explains any part we pick, and why it's built that way	Explains most of it, and finds the rest quickly	Explains some, leaning on what the agent said	Can't explain the code you shipped

Societal impact

	Exemplary	Proficient	Developing	Beginning
Personal systems map	Specific to you: real changes in your skills, habits and dependence on AI, and how they affect each other	Your skills, habits and dependence, with some links between them	Generic: it could be anyone's	Missing

	Exemplary	Proficient	Developing	Beginning
Societal systems map	Your product in the world: named stakeholders, who benefits and who bears the costs, across energy, jobs, privacy, fairness and misuse, and how they connect	Named stakeholders, and most of those areas	AI's impacts in general, not your product's	Missing
Cost-benefit analysis	Measured cost per use (tokens and dollars), an energy estimate with its source, benefits and harms by who bears them, a comparison with a version without an LLM, and a conclusion	Measured cost, benefits and harms, and the comparison	Estimates without measurement, or no comparison	Missing
Reflection	Ties the maps and the analysis to a decision you made or changed	Engages with both maps and the analysis	A summary, not a reflection	Missing

Presentations and feedback

	Exemplary	Proficient	Developing	Beginning
Prototype presentation	A sharp problem, what people told you, and prototypes that show how it shaped them	The problem, what you learned from people, and your prototypes	Mostly the idea, with little from people	Not ready
Design review	A working build, what testing changed, a confident code walk-through, and a clear next step	A working build, your testing, and a walk-through	A build that mostly doesn't work, or no testing to show	Not ready
Demo Day	Your product live, the story from problem to product, and good answers to questions	A live demo and a clear story	A demo that partly works, or an unclear story	No working demo
Feedback to your pod	Specific, kind and useful every time, and acted on by your pod	Specific feedback every time	Generic, or only sometimes	None