Software Engineering

CSE-416-03 · Fall 2026

Staff

- Lecturer: Dr. Shuai Mu
  - Please address me as Shuai (start your email as “Hi Shuai, …”)
  - shuai@cs.stonybrook.edu (the @stonybrook.edu address also works, with a delay)
- TA: to be announced

Office hours

- MW 1–2pm, NCS 351 — right before class
- Book a slot on GCal (at least a day ahead if you want a locked-in time)
- Class is MW 2:00–3:20pm in NCS 120
- After class I go to another course on West Campus, so “walk with me after class” will not work this semester

Piazza (for announcements, not debugging)

- https://piazza.com/stonybrook/fall2026/cse41603
- Join today. Course announcements go there.
  - If I post a broadcast, the assumption is you will read it within 24 hours.
- Do **not** treat Piazza or Stack Overflow as your first stop for coding help
  - That workflow is from a previous era
- Stuck on JavaScript, CSS, a stack error, a weird build? Ask an AI.

Class website

- http://mpaxos.com/teaching/se/26fa/
- Syllabus (also as PDF), schedule, office hours
- Updated as the semester goes

A little history of the class…

Era 1: J2EE + UML

- The old 416: Java, J2EE, and a lot of UML
  - Use-case diagrams, class diagrams, sequence diagrams — then code
- Lectures taught **the** process and **the** stack
- You designed on paper (or in a modeling tool) before you shipped
- A fine way to learn vocabulary. A slow way to ship a product.

Era 2: Python / Node / MERN

- Then the class became a web-app class
  - Django, Flask, Node, React, Mongo — MERN and friends
- Still a prescribed stack. Still a lot of lectures on how to type it.
- You built **a** web app, not **your** product
- Faster than J2EE. Still last decade's default internship stack.

This year is different

- Those two eras assumed you would type the stack by hand
- Last year: one shared project (an IPFS-like blockchain) in a fixed stack
- This year: **you** pick the product
  - Web or desktop; no required domain, architecture, or tech stack
  - Build it the 2026 way: agents, not hand-typed homework

What will we do in this class?

- Prepare you for a software-engineer job **in 2026**
  - Modern Internet companies (FAANG-style and startups)
  - Two overall abilities: **hack** (with AI) and **communicate** (teamwork)
  - Ship a product, not a pile of hand-typed homework
- We will do this by **PRACTICE**
  - A semester-long project with a team, of your own choosing
- Lifecycle of a project
  - Find what you want to do → design → implement → test → ship
  - Learn, discuss, and review at each step

What will we NOT do in this class?

- We will not teach you to type JavaScript / CSS / Python by hand
  - We are not going to pretend we can beat AI at that
- We will not debug your specific setup problems
  - e.g. “my Python 2 library does not work with Python 3 — can you help?”
  - Ask an agent. Paste the error. Iterate.
- There are no exams, quizzes, or a required textbook

Requirements

- Officially: C or higher in CSE 316; U4 standing; CSE major
- Informally: you should be almost ready to work as an engineer
- You can steer a coding agent
  - You do not need 12 weeks to “learn a language” before you can ship in it
- CSE 305 / 333 / 336 / 337 / 356 are **not** required
  - Need a database, a UI, a new stack? Have the model help you pick it up.
- If your instinct is still “I should write this CSS myself” — unlearn that

The project is the course

Class project

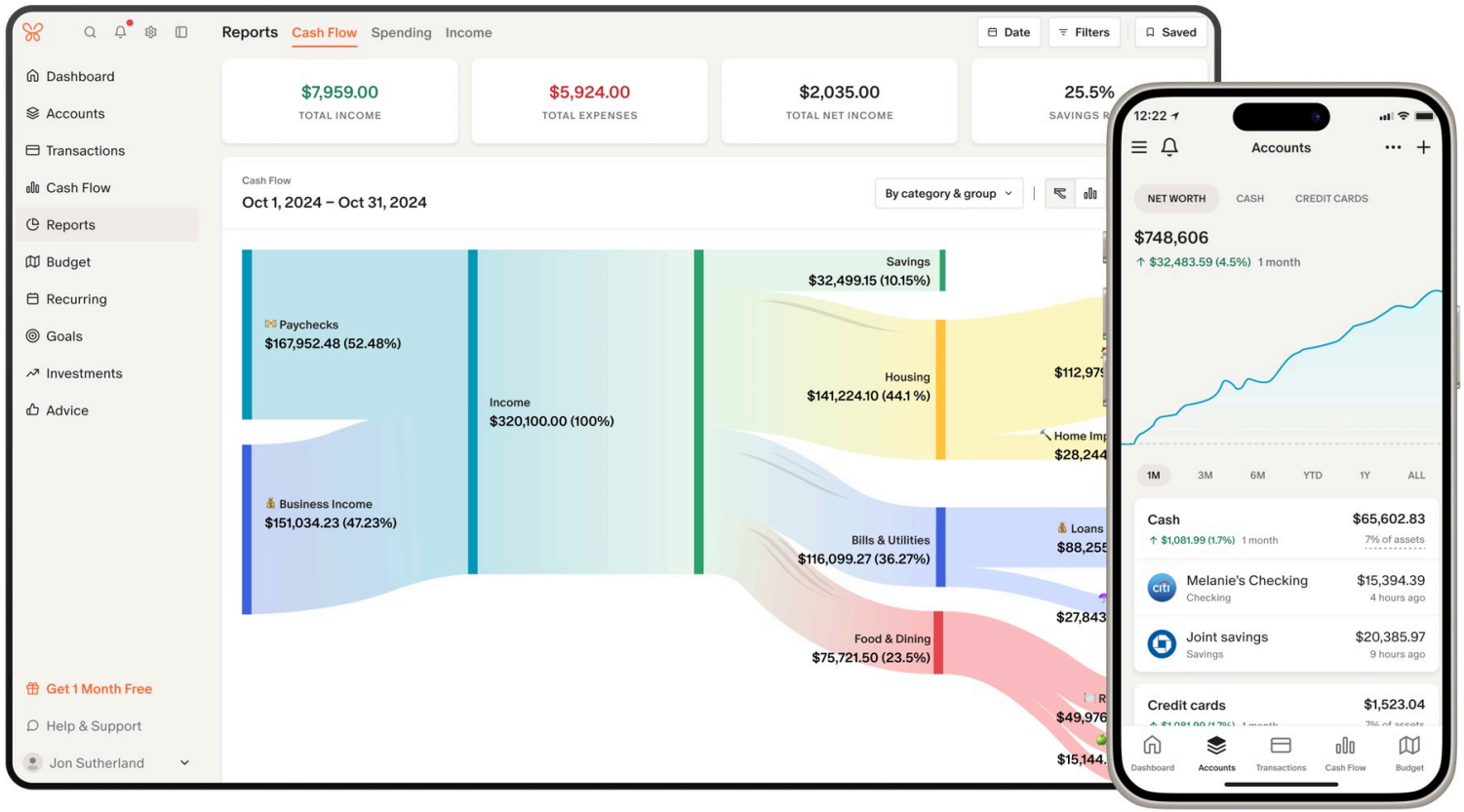
- Pick something you actually want to ship by December
  - Web, desktop, or mobile. We will **not** assign a topic, stack, or architecture.
- It has to be a **product** with real users — not a one-purpose tool
  - Accounts; login; something they actually do after they are in
  - Several screens, shared data, more than one kind of user or workflow
  - AI makes coding faster; it does not turn a thin page into a product
- You should be able to say who the users are and why they come back
- Do not want to invent a product? Two staff projects below. If unsure, ask **before** you lock it in.

You do not have to invent the idea

- Eric Schmidt (ex-CEO of Google) to Stanford students, 2024:
  - Tell your LLM: **“Make me a copy of TikTok … produce this program … release it.”**
- That is a fine way to pick **this** project
  - Choose a popular product people already use. Clone the **idea**.
- Build **your** version: your code, your name, your users
  - Do **not** take their logo, music, trademarks, or data

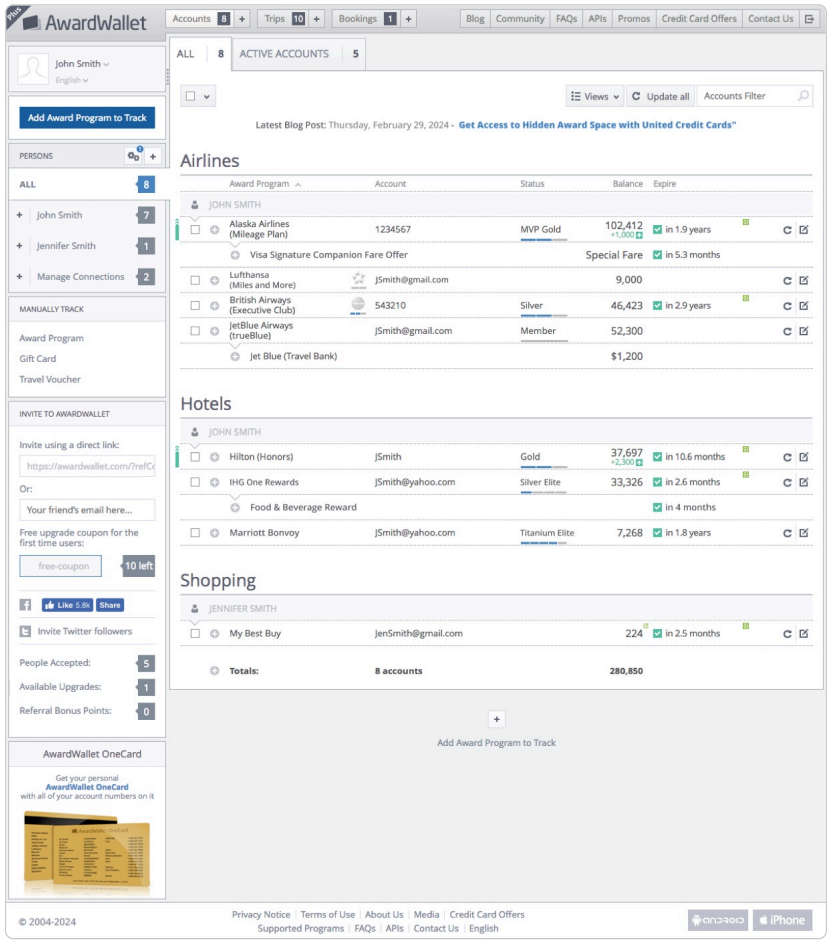
Example: Monarch

Money app — accounts, transactions, reports.



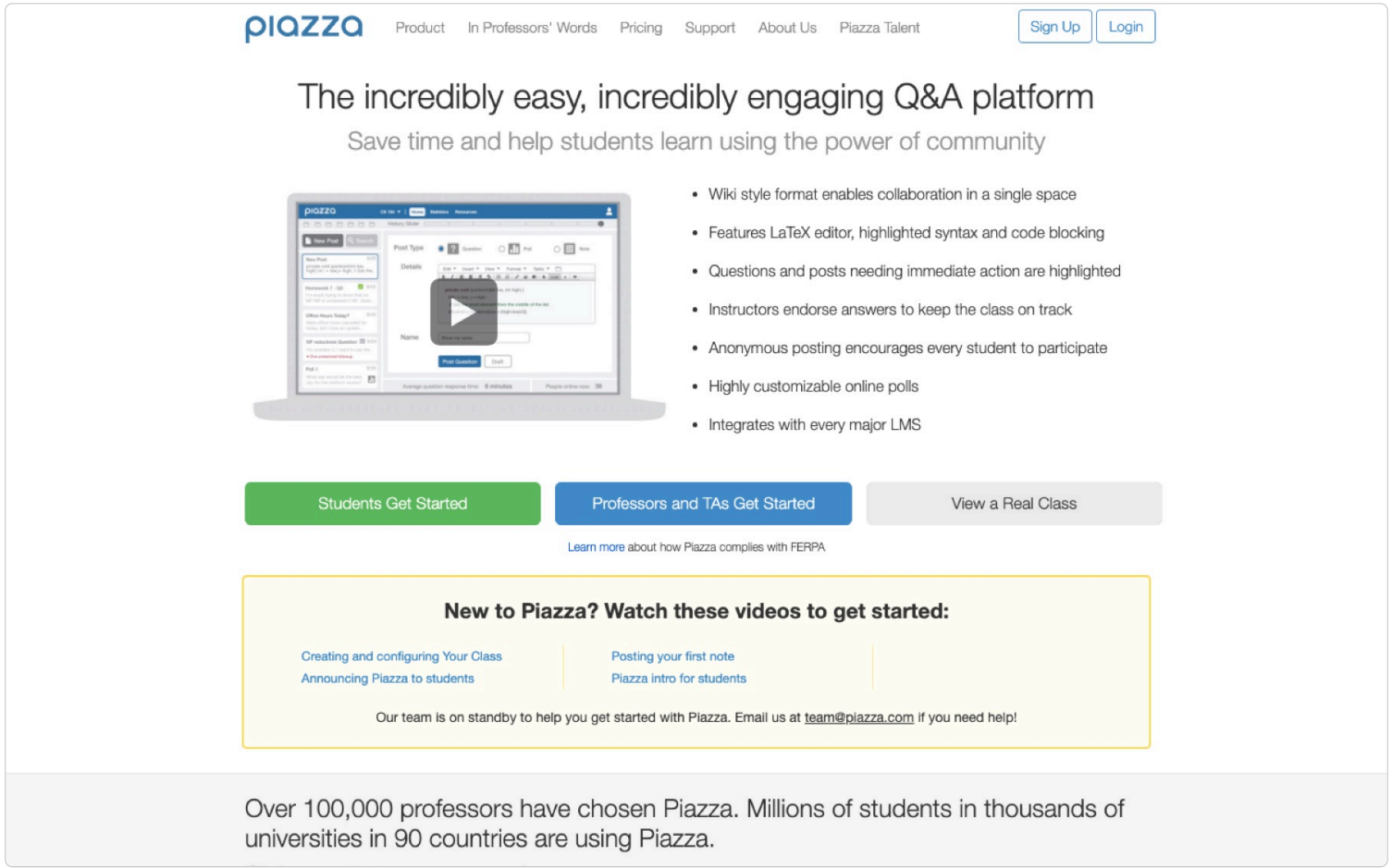
Example: AwardWallet

Loyalty / trips — many accounts, not a scraper page.



Example: Piazza

Q&A / discussion — accounts, posts, a real class.



Example: Stardew Valley

A video game.

What does not count

- A thin utility page, even if it technically has visitors
- A campus map of Stony Brook, a torrent search site, or any similar one-purpose tool
- **Not allowed** (morality): gambling, adult content, prediction markets (e.g. Polymarket-style betting)
- If you are unsure, ask

Backup: work on a staff project

- Some of you would rather **not** invent a product from scratch
- You can join a team that works with me on a systems project instead
- Two options. Both are **huge**. That is the point, if you like systems.
- Same milestones, same grading. You still have to ship and explain.

Chiral Network — A blockchain

- Last year's project, continued
- A **blockchain for resource sharing** — not just a token
  - Exchange resources (storage, compute, bandwidth, …)
  - Run hosted functions on the chain (think Internet Computer canisters)
- If you have ever wanted to know how that kind of network is actually built

Option B: a cloud database

- A cloud database in the spirit of **Supabase**
  - Auth, storage, a query API, the boring infrastructure everyone uses
- Postgres-shaped, cloud-hosted, meant to be a real backend for apps
- If you care about how a production database service is put together

Staff projects: the deal

- You work with me; we will not pretend this is a weekend app
- You will learn a lot **if** you are interested in systems
- Say so when you form teams (Aug 26 / 31). Limited seats.

a student life vs. an engineer one?

a course vs. a real project?

This class is closer to the right-hand side.

You will not be told exactly what to build. You **will** be asked to scope, design, ship, and explain the choices.

This class will be very different from your past classes…

- No textbook, no quizzes, no exam
- No one will tell you exactly what you must (or must not) do
- You will need to work with others
- You will need to **read and explain** code — including code a model wrote
- Use AI as much as you want. That is the default, not a loophole.
- Grades come from what the team **ships**, how you contributed, and one group presentation — not from a curve of exam scores

Find your teammate

- Form a team during the Aug 26 and Aug 31 sessions
  - It is **your** responsibility to join a team
- Aim for about 5–6 people
  - Ten presentation topics; one talk per team
- Can I work alone? Yes and no
  - This is a **team-project** class, not just a project class

A few suggestions on finding teammates

- Communication
- What type of person are you?
  - A hacker? A leader? An “HR”?
- Non-technical reasons matter too
  - Where do you work (on-campus? off-campus?)
  - When do you work (morning person or evening person?)
  - Why do you work (just pass, or maybe turn this into a startup?)

How will we spend time in class?

- A few lectures
  - Concepts that matter; nothing to memorize
  - We will not go into the specifics — read the docs when you use a tool
- Student presentations (one per team)
  - A **story**, not a textbook recap
- Milestone reviews
  - About 10 minutes: what you finished, why, how the team (including AI) worked
- Week 1: two free sessions to team up and brainstorm

The shape of the semester

Lectures
Aug 24 Overview (today)
Sep 2 SE tech overview
Sep 9 Modeling / spec-driven
Sep 7, Oct 12, Nov 25 no class

Presentations (10 topics)
Sep 21 → Nov 23

Milestones (in class)
Sep 14–16 M1 Proposal
Sep 28–30 M2 Design & setup
Oct 19–21 M3 MVP
Nov 2–4 M4 Feature-complete
Nov 16–18 M5 Testing & integration
Nov 30–Dec 2 M6 Final demo
Dec 7 Poster session

Class presentations

- One **30-minute** talk per team; everyone speaks
- Split one big story, or each person tells a smaller story on the same topic
- Tell a **story**: how a company, an incident, or a real project actually does this
  - What they do, what it is for, what happened
  - Interesting failures count
- Strongly discouraged (you would get a low grade):
  - “What is X”, taxonomies, concept dumps
  - Assume the class can look those up
- Send slides at least 72 hours before your slot
- Topics assigned after teams are set

How am I evaluated?

- **Course project — 80%**, in six milestones
  - M1–M5: 10% each
  - M6 Final product: 30%
- **Class presentation — 20%**
- No exams
- Milestone grades look at the quality of what the team delivers **and** each student's contribution
- Teams are graded on the day they present
- We will ask you to assess your teammates at milestones

What a milestone review looks like

- About 10 minutes
- Cover (1) what you finished, (2) the design choices and why, (3) how the team — including AI — worked in this phase
- Have the artifacts in the repo (proposal, design, tests, …)
- Bring a short demo or walkthrough

Override: ship something people actually use

- One way around the usual milestone grading
- If the product has **real** daily active users for **at least two weeks**:
  - 1000+ DAU → course grade **A**
  - 500+ DAU → course grade **A–**
- We will check the numbers
  - Classmates clicking a link, bots, and fake accounts do not count

We live in a revolutionary age of AI

Use AI as much as possible

- This is not “allowed.” This is how the work gets done.
- Autocomplete (Copilot) is the floor, not the ceiling
- Agentic tools (Cursor, Claude Code, Codex, …) are better
- Vibe-code if you want. Loop-code if you want. Spec → generate → run → paste the error back → repeat.
- Do **whatever you think is productive**

Do not write the code by hand

- Do not sit there typing JavaScript, CSS, HTML, or glue code like it is 2018
- We are not going to pretend a student (or the instructor) beats a frontier model at that
- Your job is to **direct** the work: what to build, whether it is right, what to try next
- If you are grinding out a stylesheet by hand, you are practicing the wrong skill

Piazza / Stack Overflow are the old world

- Searching for an error, posting a snippet, waiting for a stranger — that loop is slow now
- First: paste the error into an agent and let it iterate
- Piazza is for **course** news, not for “how do I center a div”
- Staff will not hand-debug your toolchain

What AI does not replace

- You still have to **understand** what shipped
  - If you cannot explain it to a teammate, you did not do the work
- You still decide **what** to build and **why**
  - The model implements; it does not pick the product
- You are still on the hook for correctness, security, and quality
- Disclose AI use the way you would cite any other source

TODO this week

- Join Piazza today
- Wednesday (Aug 26): a **3-minute** lightning talk on the ideas you want to do
- Aug 26 and Aug 31: form a team and brainstorm a project
- Lock in something you actually want to build
- Once the team exists, share a GitHub (or similar) repo with the staff
  - Details in class
- Presentation topics will be assigned after teams are set

Attendance

- No one will take attendance on ordinary lecture days
- Milestone days and your presentation day **are** the grade — be there, ready to demo
- Use AI as much as you want, on everything
- Look at any online materials that are legal (do not hack into them; follow the license)

Penalty

- You will receive an F if you
  - Copy another team’s project, presentation, or writeup
  - Claim someone else’s work as yours
  - Sabotage this class or others’ work
- Please do not negotiate for more scores or extra work
  - Each attempt is a 10% penalty on the overall grade, and it stacks
- I hope we do not need to apply this. I will.

General advice: hardware

- Dev machine: any modern laptop is fine
- You will need to **demo** a running system at milestones
  - Local is fine early; later milestones want something teammates (and we) can actually run or visit
- A cloud VM is useful if you ship a web service, not required on day one

General advice: choosing toolchains

- Pick a stack your **agents** are good at, and that the team can demo
- Language-first or framework-first — either is fine
  - Python then Django, or Rails then Ruby, or whatever ships
- Do not choose a stack because you want to “learn it the hard way” in this class

General advice: choosing toolchains (cont'd)

- Better to avoid
  - Toolchains that lock you to one platform (e.g. C#)
  - Tools with pretty demos but no real community
  - Tools that were once popular and are now unmaintained
- "Is the model good at this stack?" is a fair question now
- There is no required stack. Pick something the **team** can ship with AI, not something you plan to type by hand.