Milestones and presentations

CSE-416-03 · Sep 2, 2026

Today

- Teams should be forming. Next class (Sep 9): modeling / spec-driven.
- Then the rest of the semester is mostly **reviews** and **talks**.
- Today is the vocabulary for those two things:
  - What a milestone **is**, and what we will look at
  - What each presentation topic **means**, and how to talk about it
- Not a stack lecture. You pick the stack. The process is shared.

The grade, in one picture

Project 80%

M1	M2	M3	M4	M5	M6
10%	**10%**	**10%**	**10%**	**10%**	**30%**
Req	Design	MVP	Features	Test	Ship

+

Talk 20%

30 minutes

per team

- No exams. You are graded on the **day you present**.
- We will also ask you to assess your teammates.

Part 1 · Milestone reviews

A review is not a lecture

- Two class days per milestone (except the poster)
- Your team gets **about 10 minutes**
- Cover three things — always:
  1. What you **finished**
  2. The design choices, and **why**
  3. How the team worked — including AI
- Artifacts live in the repo **before** you walk in
- Bring a short demo or walkthrough, not a slide dump of the syllabus

Your review day (same all semester)

Monday teams

1. ChalkTalk
2. Virtual Tabletop
3. Productivity app
4. Meet People App
5. Seawolf Rides
6. Wolfie Companion
7. (全賭け)Zenkake

Wednesday teams

1. Harmonic
2. VITA Site Management System
3. Animal sanctuary video game
4. Personal Financing App
5. WorkIt
6. MiCasa
7. Transit

Seven teams × 10 min. That **is** the day. Be ready in this order.

The 10 minutes

1 · What

Show the thing. A running path, a doc, a test that exists. Not “we plan to…”.

2 · Why

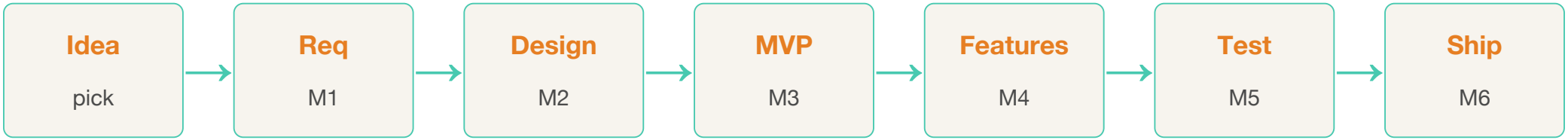
One or two decisions. Alternatives you dropped. Trade-offs.

3 · How

Who did what. How AI was used. Where a human still decided.

- If you only show a demo, we cannot tell whether you **designed** it.
- If you only show slides, we cannot tell whether it **runs**.

Lifecycle (this is the course)



- Each box is a **kind of work**.
- You can loop (tests find a bug → you change the design). The review still asks: **which box are you in?**
- AI speeds coding. It does not skip a box.

M1 — Proposal and requirements (10%)

- **Problem + users.** Who is this for, and what hurts today?
- **Why a semester.** Not a weekend with ChatGPT. Several workflows, shared data, accounts.
- **Scope.** What is in v1, what is explicitly out.
- **Who does what.** Names, not “we will all code.”
- Artifacts: written proposal, requirements, user stories (or equivalent), team roles, a **rough** architecture sketch

Requirements, in one picture

User	Need	Requirement
“Students who splits rent and groceries with roommates.”	See who owes what **without** a group chat of screenshots.	What: add a bill, split it, mark it paid. How: fast enough to do in a store line

- **What:** the job you can demo. **How:** that same job is actually usable — quick in a store line, and you do not lose the bill if you close the app.
- A user story: **As a ___, I want ___, so that ___.**
- M1 fails if the “requirement” is “build an app with AI.”

The format is flexible

- Flexible: Markdown, a wiki, user stories, a spec, a structured prompt, something you can use to coordinate with your team
- It has to be a **document** a person can read. Not a chat dump, not a 40-page IEEE template
- Enough detail to **move forward**: another human, or an agent, could start implementing from it
- If the next person still has to invent the product, the doc is not done

M2 — Design and setup (10%)

- Architecture: boxes and arrows for **your** system (clients, APIs, data, jobs)
- Stack: what you picked, and **one sentence why** (agents good at it is a valid why)
- A repo someone else can clone, with a CI **skeleton** (lint / test / build — even if thin)
- A **minimal prototype** that runs — not the product, a heartbeat
- Artifacts: A design doc that connects to the requirement, and the prototype

M3 — MVP (10%)

- MVP = **minimum viable product**: the thinnest path a real user could complete
- End to end: sign in (or equivalent) → do the **one** core thing → see the result persist
- Not “all screens sketched.” Not “the database exists.”
- Artifacts: working MVP, updated design doc

MVP is a slice, not a shrink

Whole product (later)

Accounts, 8 features, admin, email, polish, mobile, …

Do **not** try to demo all of this at M3.

MVP slice (M3)

One user type, one core workflow, data that saves, you can click it live.

Example: add an expense and see the roommate balance change.

- If the core loop does not run, it is not an MVP. It is a prototype (that was M2).

M4 — Feature-complete (10%)

- The **planned** feature set for the semester is in
- Still allowed: ugly UI, missing edge cases, “we will harden tests next”
- Not allowed: “the other half is in a branch”
- Talk about trade-offs: what you cut, what you refused to cut
- Artifacts: implementation of planned features, updated docs

M5 — Testing and integration (10%)

- Integration: the pieces actually talk (auth + data + UI, not three demos)
- Bugs you found and **fixed** — a log of that is evidence
- Something we can run or visit without your laptop magic
- Artifacts: tests and results, deployed or otherwise runnable, updated docs

M6 — Final product (30%) + poster

- M6 is most of the project grade. Treat it like a ship, not a checkpoint.
- Final demo + a short retrospective: what worked, what did not, how AI was used
- Artifacts: working product, cleaned repo and docs, final report (includes the retro)
- Dec 7: poster session, last day of class — show the finished thing

What we are actually grading

- Has the work been done?
- Do you own your work, or does AI do?
- Are there “real efforts” put into this project?
- Are the team working in their comfort zone, or are they thinking and working towards making this a successful project?
- Do the team work as an organized group?
- Have all team member contributed?

Part 2 · Class presentations

The talk is 20%

- One **30-minute** presentation per team. **Everyone speaks.**
- Send slides **72 hours** before so we can say “this is not what we are looking for, rewrite it”.
- You can sign up which topic you want to do today.
- You get a chance to do it again, but …

A story, not a textbook

- Pick a **company, incident, or real project**.
- What did they do, what was it for, what happened (good or bad)?.
- We do not want to hear 10 minutes of definitions!

One big story, or a few small ones

One story, split

Four or five people tell **one** incident. You take scenes: who, what they did, what broke, what they changed.

Several small stories

Each person tells a **different** incident on the **same topic**. Same bar: real, what happened — not a definition.

- Either way: 30 minutes, everyone talks, one topic.
- Do not split “I do the vocabulary, she does the example.”

The 10 topics

1. User experience (what users need)	Sep 21
2. Architecture and tech stack	Sep 23
3. Prompt engineering / AI-assisted	Oct 5
4. API design and documentation	Oct 7
5. CI/CD and DevOps	Oct 14
6. Testing	Oct 26
7. Debugging and observability	Oct 28
8. Security and secure coding	Nov 9
9. Maintenance, debt, refactoring	Nov 11
10. Responsible and ethical tech	Nov 23

- We have flexibility: if you want to discuss something else, ask me.

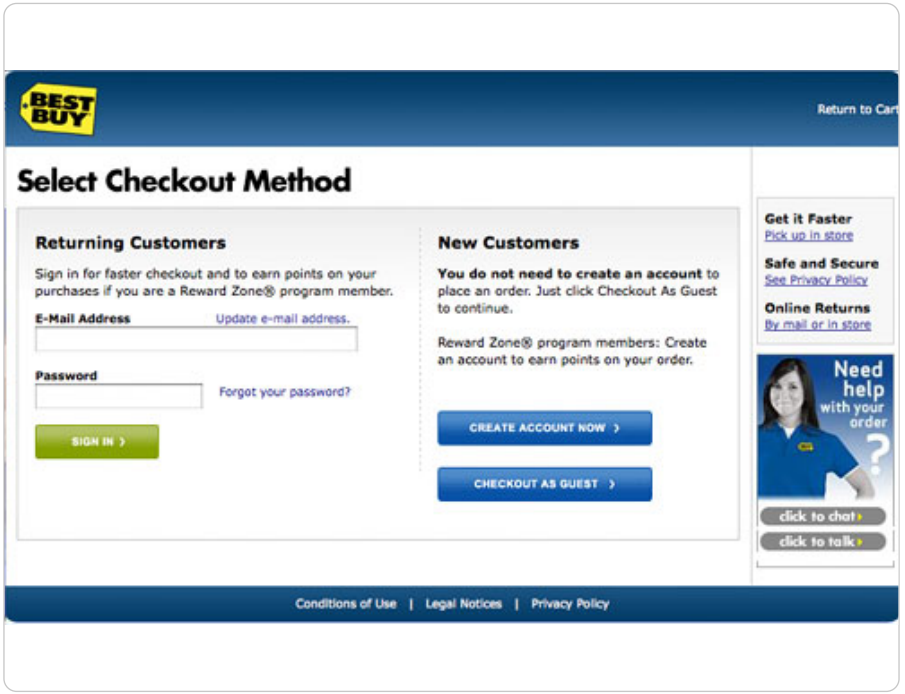
1 · User experience

BlackBerry keyboard vs iPhone glass

- BlackBerry bet professionals needed keys they could feel, email on the train, no looking down
- The screen was small because the keys were the product. RIM sold “a tool, not a toy”
- iPhone (2007) deleted the keyboard. The whole face is glass. Software can change without a new keypad
- Enterprises said they would never switch. Then they did, a second phone, then the only one
- The market chose a worse typer and a better phone
- There is also another story of iPhone vs “PDA”

The “$300 million button”

- Jared Spool (UIE). Large store site — rumored Best Buy; he never named it
- Checkout opened on **Register**. People thought they had to join a club
- Returning customers who forgot a password bounced rather than recover an account
- Button became **Continue**. About 45% more purchasers, $300 million that year



Best Buy — Checkout as Guest

2 · Architecture and tech stack

Uber’s driver-app rewrite

- By 2017 the driver app was hard to change: many teams, old MVC, every feature fighting the last one
- Drivers are the supply. If the app is slow or crashes, they log off and the marketplace dies
- Rewrite (Carbon, 2018): native iOS and Android, plus RIBs — routing follows **business** logic
- They looked at React Native and Flutter and skipped them (too heavy for Uber Lite)
- An architecture story needs **why this shape, why not that stack**, in a product that already had users

Bezos, 2002: talk through interfaces

- Bezos mandate (as later told by Yegge and others): teams talk only through service interfaces
- No reading another team's database. Do it, or you are fired
- Reaching into each other's stores was faster — and meant you could not scale a team without scaling a mess
- That shape later became AWS: if you already talk over a network API, you can sell that API
- Cost: almost every feature is now a distributed system. Architecture is the **shape you are stuck with**

3 · Prompt engineering / AI-assisted

Air Canada’s chatbot (2024)

- Jake Moffatt’s grandmother died. He needed to fly
- The site chatbot said: buy full price now, claim bereavement later (90 days)
- False. Real policy: ask **before** you fly. Air Canada refused the refund
- They argued the bot was a separate legal entity. A BC tribunal (Feb 2024) said no — about $800
- Shipping a chatbot is shipping a **policy**. If the policy is wrong, you still own it

Claude’s loop: a C compiler (2026)

- Nicholas Carlini (Anthropic) did not write a clever one-shot prompt
- He stuck Claude in a loop: finish a task, start the next. Humans own the loop and when to stop
- 16 agents in Docker, one git repo, lock files. A C compiler in Rust — 100k lines, two weeks, $20k
- When they all chased the same Linux bug, GCC was the **oracle** so they could split the work
- It built Linux, Redis, FFmpeg. The story is the loop and the check — not a magic prompt

4 · API design and documentation

Stripe: old versions stay up

- Each account is pinned to an API version dated like `2019-12-03`
- Last year's mobile app keeps talking that shape
- A renamed JSON field is a **new** version. Old clients keep working for years
- Expensive for Stripe: old shapes stay documented and tested, so Black Friday does not 500
- The opposite: rename `amount` → `amt` in place. An API is a promise about **time**

Twitter / X API lock-down (2023)

- For years the public API was open enough to build a paper, a bot, or a business
- The contract: here is how you fetch a tweet, here are the rate limits
- In 2023 X cut most of that access or priced it out. Tools died in days
- There was no cheap `/v2` you could rewrite against over a weekend
- A lock-down is still an API story: what you promised, then took away

5 · CI/CD and DevOps

Knight Capital, 1 Aug 2012

- Automated stock trader. New code went to a cluster. One box still had an old flag:

  `Power Peg`
- That flag woke eight-year-old **test** code. Unlimited buy and sell orders
- About 45 minutes. About $440 million. The firm had to be sold
- “We have CI” missed it: tests were not running against the bits that actually executed
- Deploy is which bits run in production — and whether a dead feature can still wake up

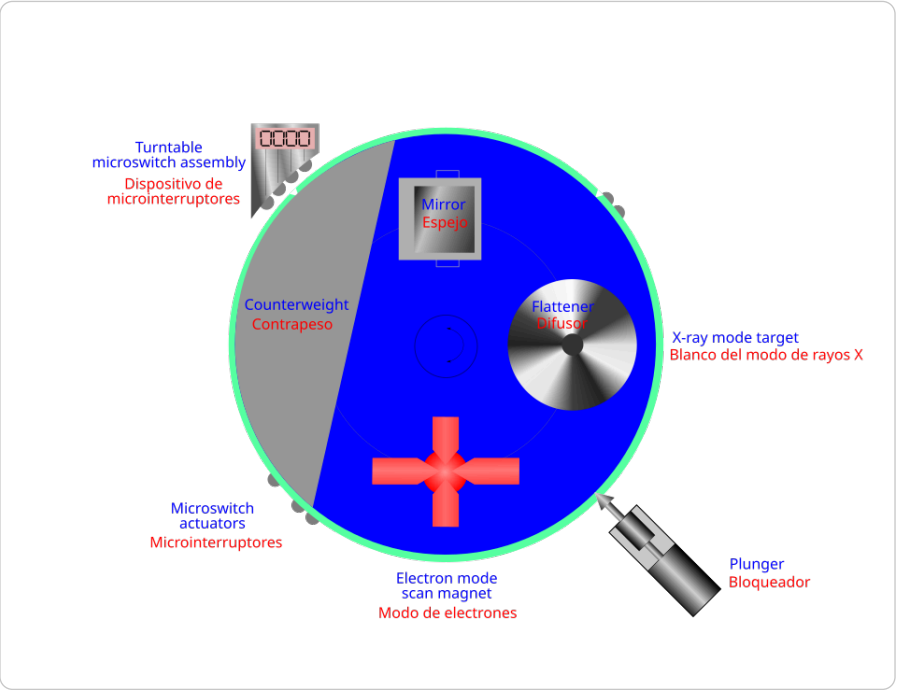
Etsy: ship on Friday

- Many teams ban Friday deploys after a weekend outage
- The ban is a symptom: deploys are large, rare, and scary
- Etsy kept shipping Friday: feature flags, one-click rollback, tiny diffs. Boring on purpose
- If you can undo in minutes, the calendar is not the risk
- A job title is not DevOps. The **loop** is how code reaches production

6 · Testing

Therac-25

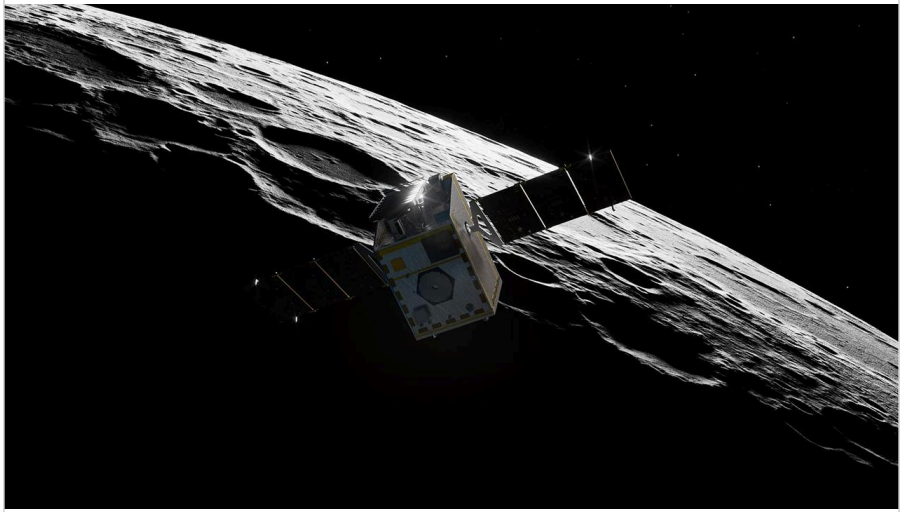
- 1980s radiation-therapy machine. Type too fast: a race, plus reused code, could fire an overdose
- Almost no independent hardware check. The software **was** the safety system
- Hospitals blamed operators for “typing wrong.” Several patients died
- A test that never races the UI, or never checks the **actual dose**, looks green
- “The beam is safe” has to be executable the way operators actually use the machine



Therac-25 turntable — mode vs hardware

Lunar Trailblazer (2025)

- NASA, Feb 2025: map water on the Moon. About $72 million. Signal and power gone in a day
- Solar arrays pointed 180° from the Sun — a sign / axis error between hardware and flight code
- An end-to-end pointing test before launch should have caught it
- Fault software then fought recovery. A reset put the wrong pointing back
- Many unit tests can still miss the one path the vehicle actually flies



NASA / Lockheed Martin — artist concept

7 · Debugging and observability

Facebook, 4 Oct 2021

- A backbone command meant to check capacity dropped **all** links between data centers
- The audit tool that should have blocked it had a bug
- DNS withdrew BGP routes: “we cannot see HQ.” facebook.com became unfindable
- Internal tools and some badge readers lived on that network. About six hours down
- Observability that lives on the network you just killed is not observability

AWS S3, 28 Feb 2017

- Playbook: take a **few** S3 boxes out of a billing cluster. Wrong range. A large set came out
- Index and placement both need a quorum. They went down. GET/PUT in us-east-1 stopped
- Half the internet stores files on S3, so sites failed in a pile
- AWS's status page **ran on S3**. The dashboard looked fine or would not load
- The graph you trust can live on the system that is on fire

8 · Security and secure coding

Equifax, 2017

- A known Apache Struts bug was public, with a patch. A consumer portal was not patched in time
- Names, SSNs, birth dates for about 147 million people
- Not a clever 0-day. A patch they did not apply
- The miss was **inventory**: what is on the public internet, and a process that actually installs the fix
- Secure coding is also the code you **run** after the world already knows it is broken

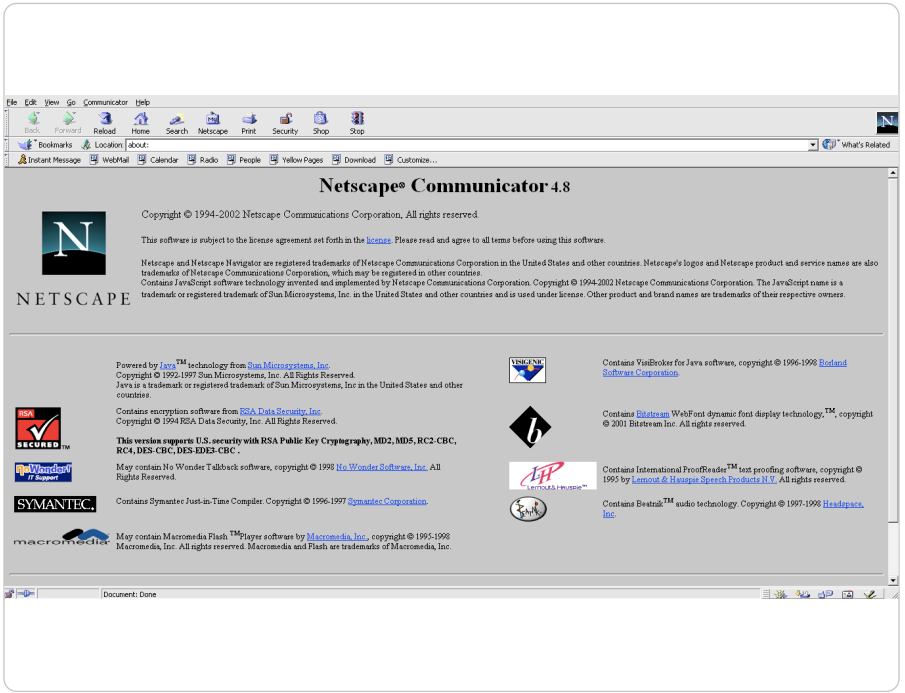
Uber, 2016: keys on GitHub

- AWS keys left in a GitHub repo. Attackers downloaded about 57 million riders and drivers
- Uber paid $100k to delete the copy and hid the breach for a year
- Later: fines, and executives charged over the cover-up
- The leak is a hook, a scanner, and not putting prod keys in git
- The year of silence is the other half: what you do in the next hour, and who you tell

9 · Maintenance, debt, refactoring

Netscape’s rewrite

- Navigator 4 was too messy to keep changing. They rewrote from scratch — what became Mozilla
- The rewrite took years. In the gap, IE shipped with Windows and took the market
- Users had something that worked. Netscape had a construction site
- Shape **and** behavior were both new. No “same product, cleaner insides”
- Refactor: change shape, keep behavior. Rewrite: you bet the company on a date you do not control



Netscape Communicator 4.8

EVE Online: Python 2 → 3

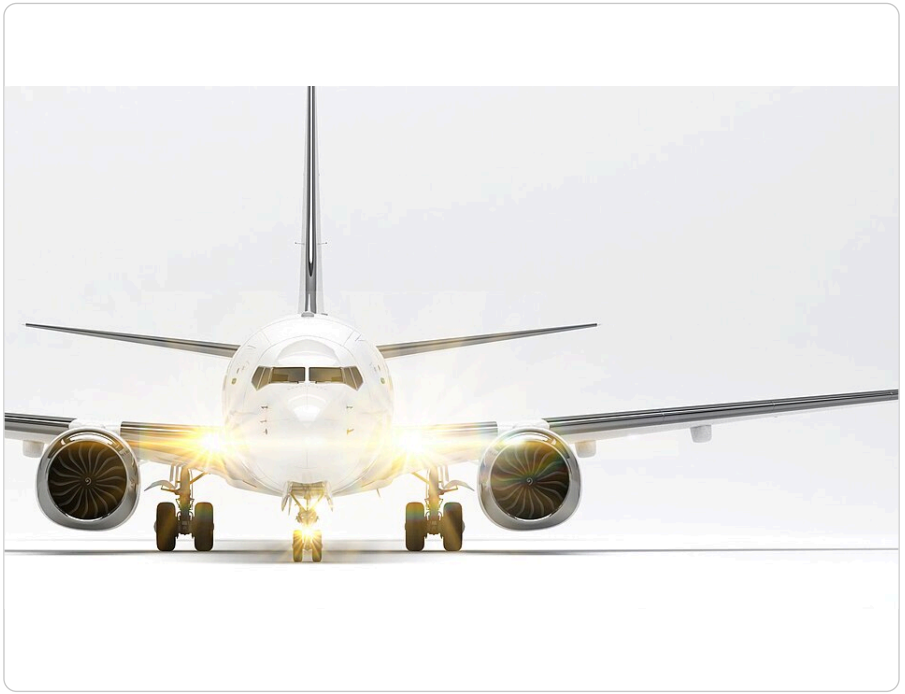
- EVE launched in 2003 on Stackless Python. Last bump: 2.7 in 2010. Sixteen years on the same version
- Python 2.7 died in 2020. The game kept flying: 2.4 million lines, about 20k files, Tranquility almost never down
- Aug 2026: Stage 1 — make it Python 3-ready **while it still runs 2.7** (Python-Future / 2to3)
- Parsing is the easy hill (about 3,300 blocking lines). About 20,000 lines compile on both but **behave** differently (`1 / 2` is 0 vs 0.5)
- EVE Frontier already runs Carbon on Python 3. Success: players notice nothing

EVE Online — New Eden

10 · Responsible and ethical tech

Boeing 737 MAX

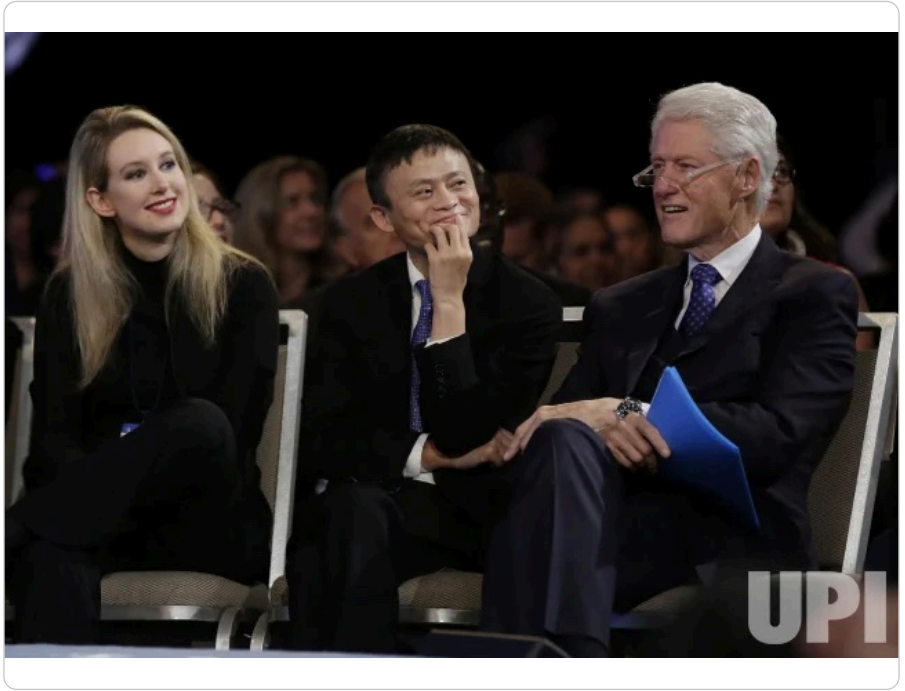
- Boeing need to upgrade their 737 “hardware” (engine) but run into some compatibility issues. The incompatibility would push the nose up.
- They could redesign the entire plane but it would be a new model, which means more paperwork with FAA
- So they implement a patch in the software (MCAS) quietly, MCAS pushes nose down.
- Pilots were not fully told it existed.
- Lion Air (2018) and Ethiopian (2019). 346 people died.



Boeing 737 MAX

Theranos

- Holmes claimed a finger-prick could run hundreds of tests on a tiny Edison box
- Investors, pharmacies, and patients were told the tech worked
- Most tests ran on ordinary Siemens machines. Many results were wrong
- The company collapsed. Holmes was convicted of fraud
- The story is the claim, the hidden pipeline, and who believed a demo



Elizabeth Holmes with Bill Clinton and Jack Ma

Do not do this (all topics)

- **UX:** Nielsen's 10 heuristics, color theory, a Figma tour — no users in the story
- **Arch/stack:** 15 GoF patterns, or a laundry list of frameworks, no product
- **AI:** "zero-shot vs few-shot vs CoT" tutorial
- **API:** REST vs GraphQL definitions + HTTP verb list
- **CI/CD:** Jenkins vs GitHub Actions feature matrix
- **Testing:** "there are 7 kinds of tests; a unit test is…"
- **Debug:** "what is a debugger? here are 8 kinds of logs"
- **Security:** OWASP Top 10 read aloud
- **Debt:** a smell taxonomy with no codebase
- **Ethics:** generic "ethics of AI" TED recap, no case

Logistics

- Topics assigned after teams are set (soon)
- **30 minutes.** Slides due 72 hours before
- Everyone speaks: one shared story (split the scenes) **or** several small stories (one each)
- Same integrity rules as the project: do not copy another team’s talk

Before you leave

This week and next

- Sep 7: no class (Labor Day)
- Sep 9: modeling, diagrams, spec-driven — useful for M1/M2
- Sep 14–16: **Milestone 1** in class

M1 is in two weeks

- If you cannot name users, a core loop, and who owns what — you are late
- Write the proposal in the repo, not in a chat thread
- A rough architecture sketch is enough; a stack lecture is not
- Use AI to draft, but make sure you read and iterate