Backend Development

CS 485/698: AI-Assisted SE

Today’s Agenda

- Team meeting (~15 minutes)
  - Standup meeting: check in with your team, discuss anything blocking you
  - Michael and I will be coming by to check in with your teams
- Brief discussion of backend development and what’s hard about it
- In-class activity: backend for “AI Tinder”

Backend Development

- The ***backend*** of a webservice is the code that runs on your (company's) servers:
  - business logic, database, etc.

Backend Development

- The ***backend*** of a webservice is the code that runs on your (company's) servers:
  - business logic, database, etc.

"***business logic***" = stuff your application is actually supposed to do. What's the business logic for our Tinder clone?

Backend Development

- The ***backend*** of a webservice is the code that runs on your (company's) servers:
  - business logic, database, etc.
- Typical architecture: backend provides a set of APIs (e.g., REST endpoints), frontend converts user intent into API calls

Backend Development

- The ***backend*** of a webservice is the code that runs on your (company's) servers:
  - business logic, database, etc.
- Typical architecture: backend provides a set of APIs (e.g., REST endpoints), frontend converts user intent into API calls
- Different concerns than frontend dev - need to worry about scalability, security…
  - Frontend concerns are about design, usability, accessibility…

Backend Development: LLM Concerns

- Can LLMs architect a backend system?
  - Especially a backend system that scales to millions of users?
- Can LLMs design an API?
  - Especially one that can be reused by many different services?
- Can LLMs design a database schema?
- Can LLMs manage authentication safely?
  - Or manage user data and PII safely?
- Can LLMs write business logic that actually implements the intentions of (non-technical) stakeholders?

Backend Development: High-level Advice

- Think in **layers**: API, algorithmic/business logic layer, data layer

Backend Development: High-level Advice

- Think in **layers**: API, algorithmic/business logic layer, data layer
- Think about scalability and performance using **worst case reasoning**
  - Your theory class talked about ***big-O analysis***; this is where to use it!

Backend Development: High-level Advice

- Think in **layers**: API, algorithmic/business logic layer, data layer
- Think about scalability and performance using **worst case reasoning**
  - Your theory class talked about ***big-O analysis***; this is where to use it!
- Never trust the client: always **validate** any data that comes from it
  - Because client code by definition runs on a computer you don't control, your ***threat model*** should assume it is malicious

Backend Development: High-level Advice

When generating backend code with an LLM, we'll want to prompt it to consider **scalability** and **security** explicitly.

- Think in **layers**: API, algorithmi
- Think about scalability and per
  reasoning
  - Your theory class talked about ***big-O analysis***; this is where to use it!
- Never trust the client: always **validate** any data that comes from it
  - Because client code by definition runs on a computer you don't control, your ***threat model*** should assume it is malicious

Backend Development: In-Class Activity

- Pair up with the person with the same letter (share one computer)
  - Early in the alphabet = front of room
- Pick one of your PRs from the snow day in-class from last week to serve as a base (doesn't matter which, you decide)
- Make (or generate) user stories for Like, Reject, and Super Like.
- Generate a dev spec for the user stories.
- Generate the backend code using the user stories and dev spec.
- Install and config the backend locally, convince yourself it works.
- Show the backend is working to one of us, make a PR.
  - If you finish, compare your backend to another finished group's

Aside: pair programming

Aside: pair programming

Definition: ***Pair programming*** refers to the practice whereby two programmers work together at one computer, collaborating on the same design, algorithm, code, or test.

Aside: pair programming

Definition: ***Pair programming*** refers to the practice whereby two programmers work together at one computer, collaborating on the same design, algorithm, code, or test.

The pair is made up of a ***driver***, who actively types at the computer or records a design; and a ***navigator***, who watches the work of the driver and attentively identifies problems, asks clarifying questions, and makes suggestions. Both are also continuous brainstorming partners.

Aside: pair programming: worth it?

Surveys of professional programmers:

[Cockburn and Williams. The Costs and Benefits of Pair Programming.]

Aside: pair programming: worth it?

Surveys of professional programmers:

- 90+% "**enjoyed** collaborative programming more than solo programming"
- 95% were "more **confident** in their solutions" when they pair programmed
- **Reduces defects** by 15% and **reduces code size** by 15%

[Cockburn and Williams. The Costs and Benefits of Pair Programming.]

Backend Development: In-Class Activity

- Pair up with the person with the same letter (share one computer)
  - Early in the alphabet = front of room
- Pick one of your PRs from the snow day in-class from last week to serve as a base (doesn't matter which, you decide)
- Make (or generate) user stories for Like, Reject, and Super Like.
- Generate a dev spec for the user stories.
- Generate the backend code using the user stories and dev spec.
- Install and config the backend locally, convince yourself it works.
- Show the backend is working to one of us, make a PR.
  - If you finish, compare your backend to another finished group's

Discussion

- How did this compare to creating the frontend?
- What are the components of your LLM-generated backend?
- Are you confident about what the LLM generated?
- Were you able to run it on the first try?
- How do you test what you can’t see?!

Wrapup and Reminders

- If you haven't yet signed up for A4, do so right now