

# EECS 498 System Design of a Search Engine

Winter 2026

## Lab 9: Design Choices

GSI: Christopher Davis  
IA: Rickey Dong

Nicole Hamilton  
[https://web.eecs.umich.edu/~nham/  
nham@umich.edu](https://web.eecs.umich.edu/~nham/nham@umich.edu)

# agenda

1. Introduction / announcements
2. Design Questions

# agenda

1. Introduction / announcements
2. Design Questons

# introduction

Upcoming Assignments:

- **Bloom Filter** 3/22
- **robots.txt** 3/29

After these two, you'll have all you'll need for the Crawler!

- I'd recommend finishing them early

# crawler roadmap

- Test locally on laptop (end-to-end)
  - Run a crawl for a few thousand docs
  - Your index team can then use this to work concurrently on their index-builder code!
  - Iron out bugs
- Test distributed version locally with multiple processes
- Set up cloud instance(s)
  - Covered next lab
  - Do short test-runs
  - Start your main crawler run

# agenda

1. Introduction / announcements
2. Design Questions

# design questions

I will ask a series of system design questions.

Take a minute or so to discuss amongst your teammates / neighbors, and then share your idea with your labmates.

# crash-resilience

A team last year took ~5 days to crawl 100 million documents on a single machine.

Having to re-do your crawl will waste a lot of time and budget.

What strategies can you use to make your crawler crash-resilient? How do you recover after a crash?

What data are you saving?

# ranking

When answering queries, should we find all matches, and then rank them? Or should we find a match, and then immediately rank it?

Why or why not? What index design decisions would influence our choice here?

```
while (auto match = isr.getNext()) {  
    auto rank =  
    ranker.rank(match);  
    results.push(rank);  
}
```

```
auto matches = isr.getMatches();  
for (auto match: matches) {  
    auto rank =  
    ranker.rank(match);  
    results.push(rank);  
}
```

# crawler

What strategies, if any, are you using to detect and prevent crawler traps?

- E.g, redirects, infinite calendar widgets, generated URLs.

# frontier

How is your team managing the frontier?

- What data structures are you using?