

EECS 498 System Design of a Search Engine Winter 2026 Syllabus

Instructor	Nicole Hamilton nham@umich.edu nicole@nicolehamilton.com https://web.eecs.umich.edu/~nham/ C: 425-765-9574
Office hours	Tuesdays and Thursdays, Beyster 4820 3:00 pm to 4:00 pm
Staff	Christopher Davis chcdavis@umich.edu Rickey Dong rickeyd@umich.edu
Lectures	Tuesdays and Thursdays, Dow 1017 Section 005: 12:00 pm to 1:30 pm Section 012: 1:30 pm to 3:00 pm Lectures will be recorded but attendance is strongly advised as lectures and labs will also serve as your team meetings.
Labs	Fridays, IOE 1610 Section 051 (goes with 005): 3:30 pm to 5:30 pm Section 121 (goes with 012): 1:30 pm to 3:30 pm Labs will not be recorded.
EdStem	https://edstem.org/us/courses/89542/discussion
Autograder	https://autograder.io/web/course/358
Prerequisites	EECS 281
Credit hours	4
Classification	Depending on your major: CE: MDE or EECS elective CSE: MDE or Flex Tech elective CS-LSA: Capstone or Flex Tech elective EE and DS: Flex Tech elective

Overview

This is a course in how to tackle a large system programming project in C++. Students will work in teams of 6 to write a complete internet search engine from scratch.

My objective is to offer students an opportunity to work on a significant *relatable* design project (something you can explain to your family and to recruiters) early in your careers. I want the experience to be like working on the startup dev team for a new product (like the one I had on the startup team for what became the Bing engine.) I hope this class will help you get a job.

Fifty percent of your grade will be on the project, evenly split between your team's performance compared to others and your individual contribution to your team. Thirty percent of your grade will be on the exams. Twenty percent is homework.

Engines submitted in the past have ranged in size. The largest have been around 20 KLOC, but the average has been around 6 KLOC. (Yes, a little smaller than you might have guessed.) Sounds easy (maybe) but there's more ambiguity and you'll be working on a larger team than in most classes.

This will be the sixth time for this course after a four-year hiatus, so the content is finally pretty stable. Students reported being very satisfied last time in their W25 course evals at <https://web.eecs.umich.edu/~nham/CourseEvals/#EECS498>.

Learning objectives

Here is what I hope you will get from the class.

1. The experience of working on the startup dev team for a significant project in C++, building a complete distributed multithreaded internet search engine from scratch.
2. The knowledge, skills and ability to work with others to research, define, estimate, plan and carry out the design of such a large, complex software project, especially, one you don't already know how to do.
3. An understanding of how a large system is decomposed into separable parts with interfaces exposing a limited surface area and how they talk to each other.
4. An understanding of how a search engine works and how its parts talk to each other.
5. An understanding of and ability to use the low-level operating system facilities for the file system, process and thread creation, synchronization, IPC and sockets.
6. How to distribute a system across multiple machines to break up a problem that's too big for one machine.
7. An appreciation of software as art and of the software design process as a fundamentally creative activity where you make choices.
8. The ability to communicate your results by demonstrating your product, producing a final report and presenting your work to the class and invited guests in a professional manner.
9. Exposure to various other topics related to the practice of engineering including professional ethics, organization behavior, negotiations, and the social impact of technology.

System design

System design projects always seem to have some defining characteristics.

1. There's an important domain-specific part that asks you to learn something new about an interesting problem you've never seen before, in this case, how a search engine works.
2. There's a need to invent a solution, an architecture, breaking the problem down into lots of moving parts, with lots of data structures and algorithms.
3. It's usually "close to the metal" with lots of low-level OS calls everywhere and the need to define file and interchange formats, perform handshakes, share resources using locks, deal seriously with error recovery, etc.
4. They're usually team efforts because they're too big to do any other way.
5. You build the whole thing, often from scratch, and at the end, you get to see it work and it feels good.

Every new design project always has a new domain-specific part, which keeps a career in system design interesting, but the rest of the skills are transferrable and they're things most people learn by doing.

What I like about a search engine as a case study in system design is that it's a really compact fun project that hits on every bit of what, to me, system design is all about.

The project

You will work on a team of 6 to design and build a complete, working, distributed, cloud-based internet search engine. You will self-select your members and you will assign your own roles. Teams that like each other and form early have an advantage, so think about inviting friends to take the class with you. (But there's no advantage to trying to start coding before the class starts.) You only need EECS 280 for C++ and 281 for algorithms because I will cover the rest of what you need to know to build any part of the engine.

These are the basic pieces you will build.

1. HTML parser.
2. Crawler.
3. Index.
4. Constraint solver.
5. Query language.
6. Ranker.
7. Front end.

In lectures, we'll walk through the problem posed by each piece and how it might be solved. A lot of the pieces need to talk to the operating system, so we'll go through a lot of examples in C++ on Linux showing, e.g., how to memory-map a file, create a thread, or open a socket. (Students who've taken or are taking EECS 482 will notice a complementary

relationship: 482 looks at the OS/application interface from the OS side; here, we'll look at it from the application side to see how to build stuff.)

You will have to make many of your own decisions as a group about what your architecture should look like, what's required, what's nice to have, how each piece will work, how the pieces will talk to each other, who's doing what, and how you'll get it all done on time.

All your work will be in C++ and all of it must be yours. Please do not submit AI-generated code. I will provide a style sheet I hope you will use. I ask that you not use a lot of STL outside of test cases unless you write your own version or subset of it. I would like you to gain the experience of working at a low level. I'd like you to do your own problem-solving, inventing all your own data structures and templates because I think you will gain from it. (If the point was to be really efficient by using someone else's code, we could just type queries into Google and be done with it.)

Your grade on the project will depend on your team's performance and your contribution to your team. Your team's performance will be determined by ranking your engine's overall performance, features, index size, and result quality against the engines created by the other teams. I will also consider your team's ability to communicate your results through your presentation, demonstration and final report.

Your individual contribution to the project will be measured by amount of code you contribute relative to others on your team and to the rest of the class. Your ability to work with others will be measured by a confidential survey of your teammates.

Topics

Here is an outline of the topics we'll discuss in lecture. We'll finish the technical content – everything you need to know to build your engine – a few weeks before the end of the semester. That's deliberate, to give you a quiet period to finish the project.

This being intended as a capstone to your undergraduate careers, we'll use the last few lectures for topics I think you'll enjoy and may be helpful to your professional growth.

1. Introduction to the course, finding teammates.
2. Search engine basics.
3. Project planning, source tree layout, scaffolding and stubs, build rules, etc.
4. HTML, UTF-8, HTTP, robots and redirects.
5. Intro to the filesystem.
6. Mapped files and hashing.
7. Memory-mapped hash tables.
8. Processes and threads.
9. Locks and RAII.

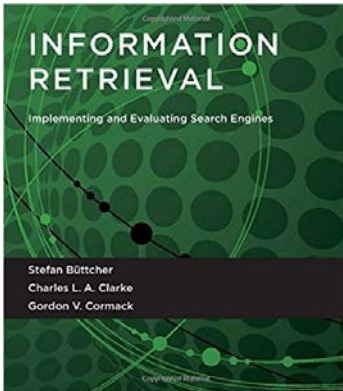
10. TCP-IP, DNS, and sockets.
11. SSL.
12. Threads, locks, and producer-consumer relationships.
13. The frontier, robots.txt, shingles.
14. Distributing an engine across multiple servers.
15. The index.
16. The constraint solver.
17. The query compiler.
18. Ranking.
19. LinuxTinyServer.
20. The front end, JSON.
21. Ethics.
22. Organizational behavior.
23. Negotiations.
24. Team presentations and demos.

Homework

You will have homework assignments nearly every week designed to help you with various parts of the project. You'll do a few of them by yourself but you'll submit the rest as a team, splitting up the work however you like. A dozen of them will be autograded coding assignments designed to help you get started with various parts of the project. But you will also have some writing assignments and some team-building exercises, e.g., to come up with a team name and photo.

Text

There will be only one recommended text. What I like about *Information Retrieval* is that it describes a lot of the pieces (except for ranking) in the same way we thought about it at Microsoft. An electronic version will be available. I recommend the first 6 chapters.



Information Retrieval: Implementing and Evaluating Search Engines
Reprint edition (February 12, 2016)
Stefan Büttcher, Charles L.A. Clarke, Gordon V. Cormack
The MIT Press
ISBN 978-0262528870

Coursepack

You'll also need a coursepack of the cases we'll be discussing near the end of the semester. You'll buy it online from Harvard as a set of PDFs after creating a free account.

Grading

Here's the weighting I expect to apply.

Group project and other activities	
Group performance	25%
Individual contribution	25%
Homework	20%
Midterm	15%
Optional final exam	15%

The midterm and final will be in-person. The final is optional; the midterm is not. If you choose to skip the final or if your score on the final is lower than your score on the midterm, I will use your midterm score for the final. (I'm trying to free up your time to finish the project.)

I grade on the curve, scaling the results by deciding what I think was an A and what I think was a C and then linear interpolating in between. I expect but do not promise that most students will fall between 3.0 and 4.0. The median will likely be around 3.4.

Late policy

Submissions are expected to be on time but I care more about students' success completing assignments and learning the material than strict deadlines. If you're stuck, need help or extra time, talk to me.

Late withdrawals

I can't stop you but I ask that you not withdraw once the teams are formed and underway. It's just not fair to your teammates, who invariably end up struggling. Talk to me before you do this.

Course revisions

Every time I teach this course, I learn how to make it better. I may make reasonable revisions to content, assignments, exams or grading as may be appropriate.

All the work must be your own

You may certainly compare notes with other students and other teams and of course I understand that you may do research using Google. But absolutely everything you turn in to me must be your own work. Please do not submit AI-generated code or written material. On your exams, you will be required to copy the Honor Pledge in your own handwriting and sign your name to it.

I have neither given nor received unauthorized aid on this examination, nor have I concealed any violations of the Honor Code.

I report everything.

Disability

Access and Accommodations: Your experience in this class is important to me and to the University of Michigan, where it is our policy and practice to create inclusive and accessible learning environments consistent with federal and state law.

If you experience barriers based on a temporary or permanent disability (including, but not limited to mental health, attention-related, learning, vision, hearing, physical or health impacts), please seek a meeting with the Services for Students with Disabilities (SSD) office at 734-763-3000 or <https://ssd.umich.edu> to help us determine appropriate academic accommodations. SSD typically recommends accommodations through a Verified Individualized Services and Accommodations (VISA) form. Any information you provide is private and confidential and will be treated as such.

If you have already established accommodations with SSD, please tell me what they are so I can be sure to provide them.