

Syllabus: CMPSC 189A -- CS CAPSTONE (Quarter 1 of 2)

[Chandra Krintz](#)
ckrintz@ucsb.edu

Office hours: By Appointment
[Course Info/Details in Canvas](#)

Last update: Sep 5, 2024

Prerequisite: CS48 or 56 or 148 or 156 or 172;
Senior standing in computer science or
computer engineering

Course Description

The UCSB Computer Science (CS) Department offers a six month, two quarter course sequence called "Capstone" in which teams of 4-5 senior students leverage their undergraduate computer science experience to pursue practical and innovative solutions to important and challenging information technology problems. Throughout the series, the teams engage in all aspects of an engineering problem including design, prototype, testing, deployment, and public demonstration. The course is based on a learn-by-doing approach in which teams employ cutting-edge software technologies, tools, and engineering practices to construct significant software applications and systems.

Teams work with industry (or research) mentors to collaborate from September to March (Fall and Winter quarters at UCSB) to push the limits of computing technology and to produce novel software advances. The teams and mentors interact weekly (virtually or in person) and work together to complete a series of milestones. The culmination of the program is a full day, public event in March at which the teams present their accomplishments, demonstrate their software products, and compete for prizes.

CS189A is the first class in the 2-class sequence taught in Fall quarter. During this class, students form teams, select projects, learn about modern software engineering practices, and implement vision statements, specifications, designs, prototype implementations, and testing infrastructure for their projects. The teams do so with advice/feedback from their mentors and the instructional staff as part of four 2-week agile development sprints. At the end of the class, the teams demonstrate their prototypes to the class and CS189B instructor and develop plans for extending and improving the project as part of the second class (Winter quarter).

About The Instructor

Chandra Krintz is a Professor of Computer Science at UCSB. She joined UCSB in 2001 after receiving her M.S. and Ph.D. degrees in Computer Science from UCSD. Chandra leads a number of different research projects that advance the state-of-the-art in programming and distributed systems in ways that improve performance and energy consumption, and that ease development and deployment of software. Her recent work focuses on the intersection of IoT, edge and cloud computing, and AI / data analytics, with applications in farming, smart cities, and conservation science (cf [SmartFarm](#) and [WTB](#)). Prior to this, she designed and developed the AppScale cloud platform and pursued research in resource-constrained, mobile and embedded systems. In her spare time, Chandra enjoys cycling, golf, and exploring back roads. <https://sites.cs.ucsb.edu/~ckrintz/racelab.html>

Learning Outcomes

Learning goals:

1. Describe what modern software engineering is and how it works today
2. Recognise the key properties of agile software development and how it came to be applied and practiced
3. Be able to specify and design the features of a project iteratively
4. Use software engineering tools to track progress and facilitate continuous development and improvement
5. Design and test a novel and innovative project using interconnected prototype components
6. Use tools to integrate, test, debug, and make robust project code
7. Demonstrate a working prototype at the end of each sprint and at the end of the class

The instructional staff will measure learning goals using class participation, evaluation of software artifacts (e.g. vision statement, PRD, and code base), code contributions, and class projects. The course will help students

- Achieve the aforementioned learning goals at a high level
- See themselves as contributors to the next generation of computing applications and systems and understand how they can impact society and the world around us using these technologies
- Build confidence in their software development and systems building skills necessary to excel in industry and computing research
- Gain an understanding of how computing research is performed in computer systems and develop the necessary foundation to participate in computing research in our department
- Learn how to work effectively with others with a range of different backgrounds, interests, and expertise

How to succeed in this course

The easiest and best way to get the most out of this class is to attend class and section and to put significant time into the project. Class and section provides the primary opportunity to meet in person as a team to make significant progress on the projects. Project participation impacts your grade and your relationship with your team and the instructor. Moreover, it impacts your relationship with the industry mentors (which can have long term ramifications).

Technologies and Software

- This course uses [Canvas](#) for course details, communication, schedule, and artifact turn-in, and [YouTube](#) (for posting recordings, if any).
 - All links and invites to these resources will be posted in Canvas
- Bring your laptop to class so that you can work with your team to complete the assignments and to work on your projects together. If you do not have a laptop, contact the instructor ASAP.

Grading criteria and policies

- If you are unable to complete an assignment or exam due to health reasons, the best course of action is to withdraw from the class. Please contact the instructor if you need to do so.
- All due dates are posted on the course schedule tab on the Canvas *Schedule* Page.
- Grade distribution:
 - project setup, documentation, and overview sentence (5%)
 - Project Vision (5%)
 - Product Requirements Documents (project specification) version 1 (10%)
 - Product Requirements Documents (project specification) version 2 (10%)
 - code/documentation contributions to github repo by all team members at the end of each Sprint (4%),
 - in-person (full team) presentation and demo at end of class (10min max, in class on last day of class) clarity, organization, and timing (10%),
 - The presentations should include: problem/objective, your contribution/solution, implementation, demonstration, and findings.
 - final project turn-in (50%: code quality, documentation (ease of setup/testing by instructors), recorded (10-15 min) presentation and demo, slides).
 - Class participation (+ peer evaluation if used); includes required readings (6%)
 - Attendance is required in both class and section each week – missing a class or section counts against your class participation score; you can miss one class or section in the quarter without being penalized
- Group size: 5 (other with permission from instructor)
- This class focuses on the development of a large-scale team project. Here are the team and [project expectations](#)
- Grading scale:
 - 90-100% of available points = A; 87-89% = A-; 83-86% = B+; 80-82% = B; 77-79% = B-; 73-76% = C+; 70-72% = C; 67-69% = C-; 63-66% = D+; 60-62% = D; 0-59% = F

Grading Policy

If you have any questions regarding your grade on the assignments, you need to inform the instructor **within one week of the grade announcement**. Complaints or concerns raised after one week will not be honored. You are encouraged to contact the instructor throughout the quarter if you have problems in the course and need some extra help.

Course topics

Lecture notes will be posted on the Canvas *Schedule* page for the class date at least 2 hours prior to the start of the class.

- Introduction to Software Engineering
- Team formation and management
- Software development processes
- Project management

- Software specification
- Software design, system modeling, and SW architectures
- Prototyping and implementation
- Open source software
- Testing and verification
- Tools and technologies

Exams

There are no exams in this course.

Course assignments

Group project:

- Expected outcomes include learning
 - (i) how to work well with others (contribute equally, learn from and teach each other, innovate together, present clearly and coherently to the class your accomplishments and findings),
 - (ii) how to construct, design, implement, and present a project related to IoT systems on your own (build your ability to think/innovate independently and develop your problem discovery/solving skills, communication and presentation ability, and programming and data analysis skills).
- Artifacts (due dates given on Canvas Schedule Page)
 - Project setup and documentation (e.g. team formation, sprint planning, retrospectives, and burndown charts)
 - Vision statement
 - Product Requirements Documents (project specification) version 1
 - Product Requirements Documents (project specification) version 2
 - Github code/documentation: weekly commits by each team member for each Sprint
 - Project presentation and demo (in class)
 - Project turn in: code repo, slides, recording of presentation, plan for CS189B

Readings

- Required book: ["Scrum/XP From the Trenches"](#) by H. Kniberg. (Free with registration).
- Required reading:
 - Reading from required book (above): Intro, sprint planning, scrum chapters
 - Read ["Scrum/XP From the Trenches"](#) by H. Kniberg. (Free with registration). Intro, sprint planning, scrum.
- Recommended book: [Clean Code](#) by R. C. Martin
- Recommended book: [Test Driven Development by Example](#) by K. Beck
- Recommended readings
 - [The Design of Everyday Things](#) by Donald Norman
 - [The five keys to a successful Google team](#)
 - [What Google Learned From Its Quest to Build the Perfect Team](#)

Prerequisites

UCSB CS48 or 56 or 148 or 156 or 172

By Topic

- Mastery of application development on computer systems
- Familiarity with algorithms including basic and advanced data structures, algorithmic techniques, dynamic programming, networking
- Familiarity with software engineering basics
- Exposure to operating systems and software engineering
- Programming and debugging experience with Python and C/C++ programming systems

Course and University Policies

- Attendance is required in both class and section each week – missing a class or section counts against your class participation score; you can miss one class or section in the quarter without being penalized.
- [Code of conduct](#) (University and departmental)
- Statement on [academic integrity](#). Any and all cheating or plagiarism in this class will result in a grade of zero on the assignment, and potentially an F in the course and reporting of the academic misconduct to the department and University. Cheating includes including code of others in your submissions of individual work without attribution (including cut/paste from code support sources like stack exchange, chatGPT, assistive AI technologies).
- [Sexual harassment and discrimination](#) will not be tolerated. If you experience or observe such behavior, please contact [Chandra](#) or other campus staff/administration for help/assistance.
- If you have disability that limits your ability to fully participate in this class or access its resources, please contact [Chandra](#) and [the UCSB Disabled Students Program](#)

Student Resources

- [Wellbeing @ UCSB](#)
- [Counseling & Psychological Services](#)
- [Events - The GradPost at UC Santa Barbara](#)

Calendar

- [CS189A Calendar](#)
- [Schedule of Lecture Topics](#) (in Canvas under Pages)