# CS 4350: Fundamentals of Software Engineering
# CS 5500: Foundations of Software Engineering

## Lesson 1.1 Course Introduction

Jon Bell, John Boyland, Mitch Wand

Khoury College of Computer Sciences

## Instructors

Mitch Wand

John Boyland

Jonathan Bell

## Teaching Assistants

Joseph Burns

Michael Davinroy

Yuting Gan

Eiki Kan

Satyajit Gokhale

Guneet Kaur

Ben Schultze

## What is software engineering?

- Software Engineering encompasses the tools and processes that we use to design, construct and maintain programs over time.

## Good code is necessary but not sufficient

- Developing software is a ***systems enterprise***
  - there are many stakeholders
  - how to determine the requirements?
  - how to design code for:
    - reuse, readability, scale? other factors?
  - how to organize the development process?
  - how to make sure you've built the thing right?
  - how to make sure you've built the right thing?

## SE includes Tools and Processes

- The answers to those questions will depend on things like:
  - the size of the team
  - the size of the product
  - the longevity of the product
- There's no one "right" way; there are always tradeoffs.
- But there are best practices, which we will expect you to follow.

## Learning Objectives for this course:

- By the end of this course you will--
  - Be able to define and describe the phases of the software engineering lifecycle.
  - Be able to explain the role of key processes and technologies in modern software development.
  - Be able to productively apply instances of major tools used in elementary SE tasks.
  - Design and implement a portfolio-worthy software engineering project in a small team environment that can be showcased to recruiters.

## Learning Objectives for this Lesson

- By the end of this lesson you should be able to:
  - Explain what SE is and why it’s important
  - List your weekly obligations as a student
  - List the requirements for completing the course
  - Explain how assignments will be graded

## Approach

- We will try to mirror the experience of a young software engineer joining a new team...
  - "onboarding" to a new codebase
  - start with small individual projects (apx. 3 homeworks)
  - participate in small team project to design & develop a major new feature (Team Project, with intermediate deliverables)

## Technology

- In our shop we use:
  - TypeScript as implementation language
  - React for web pages
  - Chakra-UI for design elements
  - Visual Studio Code
  - + git, etc...

## Course Mechanics

- Our goal is to provide a productive learning environment to both remote and on-the-ground students
- “Flipped-Classroom” Model:
  - Lecture videos will be posted at start of week:
    - watch videos before coming to class
    - Come prepared with questions!!
  - During scheduled class time: discussion, activities. If you come in person, bring laptop and headphones
  - You are expected to come to class, either in person or remote.

## Laboratories (Tutorials)

- There will be regular laboratory exercises to give you practice with the technologies we will use.
- Not graded, but highly recommended
- Typically, will consist of structured steps that will guide you through a typical task
- Typically asynchronous, but the TAs will have a dedicated office hour for handling your questions about the lab exercise.

## Pedagogy

- We are big on "learning objectives"
- A learning objective is something you should be able to do after completing the learning experience (lesson|homework|etc.).
- Every lesson and every homework will have explicit learning goals.
  - We will tell you these before and remind you of them afterwards

Like Slides 5-6!

## Course Requirements

- There will be three programming assignments and a final project. You will complete the assignments individually, and the project in a group of 3 or 4.
- The overall grading breakdown is:
  - 36% Programming Assignments
  - 29% Final Project
  - 10% Quizzes and in-class activities
  - 25% Final Exam

## Grading (1)

- We will be using a new grading system this semester, called "specification grading".
- In this system, we will give a rating on each element of each assignment.
- there will be exactly three possible ratings for each element:
    - below minimum expectations
    - meets minimum expectations
    - satisfactory

## Grading (2)

- This is called "specification grading" because we will be as precise as we can to specify what is necessary for you to achieve each rating.
- Here's an example from a hypothetical assignment about code review:

| Weight in Assignment | Criterion | Meets Minimum Expectations | Satisfactory |
|---|---|---|---|
| 4% | Naming | Identify at least 1 good name and 1 bad name in the code base.  Some examples may not be well-grounded in a design rationale | Identify at least 3 “good” names and 3 “bad” names. Each example is substantiated with 1 sentence justifying why this is a good or bad name. |

## Grading (3)

- Note: there is no partial credit. For each gradable element you will get **exactly** one of those 3 ratings.
- The factors that we might have considered for partial credit will be broken out into separate elements. The elements will be aligned, as best we can, with the stated learning objectives of the assignment.
- You will know the rating criteria before you do the assignment.
- At the end of each assignment, you will receive
  - your rating on each of the elements in the assignment
  - a 3-tuple, with the weighted total of "below minimums", "minimums" and "satisfactories" that you got.
  - So in the example, if you got a "satisfactory", that would count as 4% towards your satisfactories.

## Grading (4)

- So if you had an assignment with 4 parts, and you got the scores indicated in the matrix below, you would wind up with a score of (0.2, 0.1, 0.7)

| Item | Weight | Below Minimum | Minimum | Satisfactory |
|---|---|---|---|---|
| 1 | 0.2 | X | | |
| 2 | 0.3 | | | X |
| 3 | 0.4 | | | X |
| 4 | 0.1 | | X | |
| TOTAL | | 0.2 | 0.1 | 0.7 |

## Grading (5)

- Note: Do **NOT** ask us to average your 3-dimensional grade into a single value. The ratings are ordinal data, not linear data, and in statistics it's a big no-no to take averages over ordinal data.

Go look up "ordinal data" on the internet. It's an important but underappreciated concept.

## Grade Appeal Policy

- If you have concerns regarding the grading of your work, please let us know right away by opening a regrade request in GradeScope.
  - Do not post on Piazza or email your TA or instructor
    - GradeScope provides an interface that allows us to review all regrade requests in one place.
  - All regrade requests must be submitted within 7 days from your receipt of the graded work.
  - If your regrade request is closed and you feel that the response was not satisfactory, you may appeal to the instructor via email within 48 hours

## Late Policy

- Your work is late if it is not turned in by the deadline.
  - 10% will be deducted for late HW assignments turned in within 24 hours after the due date
  - HW assignments submitted more than 24 hours late will receive a zero.
  - If you're worried about being busy around the time of a HW submission, please plan ahead and get started early.

## Academic Integrity (1)

- Students must work individually on all homework assignments.
- We encourage you to have high-level discussions with other students in the class about the assignments, however, we require that when you turn in an assignment, it is only your work. That is, copying any part of another student's assignment is strictly prohibited.
- If you steal someone else's work, you fail the class.
- You are responsible for protecting your work. If someone uses your work, with or without your permission, you fail the class.

## Academic Integrity (2)

- You are free to reuse small snippets of example code found on the Internet (e.g. via StackOverflow) provided that it is attributed.
- If you are concerned that by reusing and attributing that copied code it may appear that you didn't complete the assignment yourself, then please raise a discussion with the instructor.
- If you are in doubt whether using others' work is allowed, you should assume that it is NOT allowed unless the instructors confirm otherwise.

## Communication

- Course web page (https://neu-se.github.io/CS4530-CS5500-Spring-2021)
- Canvas
- Piazza
  - for questions about assignments, etc.
- Slack (nusespring2021.slack.com)
  - for more general discussions
  - # ta-office-hours
  - knowledge-sharing (within the limits of the Academic Integrity Policy)
  - whatever else you might want (within the limits of the Code of Student Conduct)

## Review

- Now that you've studied this lesson, you should be able to:
  - Explain what SE is and why it’s important
  - List your weekly obligations as a student
  - List the requirements for completing the course
  - Explain how assignments will be graded

We told you that we were going to list the learning objectives at the beginning and end of each lesson, and here we are!

## OK, now let’s get down to work!

- In our next lesson, we'll learn about the basic principles of well-designed software