BU

Boston University Global Programs
Study Abroad | London

CAS CS 411: Software Engineering
2025.FALL.SYLLABUS, London

Instructor Information

A. Name	Dr Mohammad Shah
B. Day and Time	Please see course chronology
C. Location	Brompton Room, 43 Harrington Gardens SW7 4JU
D. BU Telephone	020 7244 6255
E. Email	moshah@bu.edu
F. Office hours	By appointment

About the course

What is Software Engineering and what does a Software Engineer do? Software Engineering is the discipline of designing, building, and maintaining complex systems, built out of code, that solve real-world business problems. A Software Engineer architects and coordinates the development of entire software solutions, ensuring they are reliable, scalable, and maintainable.

While Software Engineers write code more than say, Structural Engineers mix concrete, Software Engineers are not programmers. They are Engineers, many, but not all, of whom spend varying amounts of their time writing code.

How do code modules, programs, and software relate to each other? In this course, we will explore how code modules, when combined, form programs. These programs, when working together, become the software that drives solutions to large-scale business challenges. Programs are built from smaller, self-contained modules, each with a specific role, and software integrates these programs into a cohesive system.

What will you learn in this class? We will focus on writing large-scale programs composed of "highly cohesive and loosely coupled" code modules. The success and reliability of these programs depend on the development processes used. While programming will be part of the course, our primary focus will be on how to deliver and deploy bespoke, scalable solutions to complex problems.

Entering this course, you have experience writing code, but you may not yet have had the opportunity to write actual programs. This course will guide you through that process. We will be working in Python and focusing on backend microservices, but only because that is my area of expertise. We will be discussing general principles that will still apply to your platform of choice. I will ensure that I flag anything that is specifically "Pythonic" and microservice backend-oriented to ensure that what we learn is as universal as possible.

We will begin by discussing the tools of software engineering[1] before covering topics you should be aware of about how modern software works. We will end by talking about how software engineering projects are managed in practice.

We will also

- Become comfortable working from the command line, which is a necessary skill for all software engineers.
- Get our hands dirty managing modern distributed cluster-based environments. This will give you some understanding of what is going on Behind The Scenes[2] that make the environments you rely on Just Work.™

Textbook (singular)

We will be using What Every Engineer Should Know about Software Engineering, Laplante and Kassab. We will be using the **2nd Edition**. It's very important that you have the **2023 version**.

I have also *recommended* a book: High Output Management, Andy Grove. The book is just that: recommended.

Also, you are free to use any books, online resources, blogs, articles etc.

[1] E.G. architectural and software design principles. Things like git and IDEs are *programming* tools.
[2] Have you thanked your friendly neighborhood DevOps / Infra team today?

HUB

This course carries a single HUB unit in Teamwork/Collaboration.

Teamwork Outcome 1

This course is about strategies and techniques, both theoretical and practical, for writing software in teams – in other words, the course is devoted to explicit training in this Hub unit in the context of computer programming. Hence is it not surprising that students will be able to identify the characteristics of a well-functioning team, theoretically (through readings and lecture) and practically (through periodic review sessions). During the course, peer reviews are conducted in which each student evaluates the contribution of each of the other member(s) of the team in terms of their contribution to the overall effort.

Teamwork Outcome 2

Again, this course is essentially a course in “teamwork in CS,” in which a portion of the overall grade is based on the teamwork component. Considerable theoretical and practical knowledge is communicated, and practiced, in the formation of teams, writing specifications, developing plans for execution and testing of deliverables, and preparation and group delivery of the final project. A variety of tools are used which facilitate group software projects. The peer review emphasizes group reflection on the roles and participation of each member of the team.

Delivery Tools

We will be using Blackboard for sharing resources and discussion. It will also be used for our communication.

Grades:

Research/Presentation	10%
In Class Quizzes	10%
HW	5%
Tests	2x 10% each
Final	15%
Project	30%
Peer Review	10%

We will not assign letter grades to individual assignments and exams. When assigning final letter grades, we will look at the total points you earned and decide on the cut-offs for A, B, C, etc. Those cutoffs will likely be lower than the usual US high school cut-offs (90, 80, 70, etc.), because this class is harder than a typical high school class. To help you determine where you stand in the course, we will post averages of every assignment. If you are at the average, you are roughly around a B.

Research/Presentation

You will be given or ask to research on topics and present in the class. The presentation will be either solo or in group.

In-class quizzes/test

In this course we will be doing the kind of brief, in-class multiple choice quizzes you are probably familiar with from other courses.

Tests / Final

The tests will be the same short answer format as the readings. A test needs to balance between going deep on a few topics and covering a wide range of topics superficially. The goal is to be broadly familiar with a wide range of topics so that you recognize them when they come up and know what to google, and therefore in this class we focus on the latter exclusively.

Home works / Labs

There is only one way to learn to program, and that is by writing lots of code.

There will be approximately 6 homeworks related to designing and deploying a modern (toy) web-app. Some homeworks will be design-orientated. You will do these alone and have approximately 1.5 weeks to complete them. Others will be about “getting your hands dirty” writing code and getting it to work. These will be done in (randomly assigned) pairs. You will have approximately 2.5 weeks to complete these.

Project

The project will be to essentially replicate the functionality of the homeworks in an application of your choice. You are encouraged to re-use the architecture / tech stack used in the homeworks, but you are not required to do so. The required components are:

- A decoupled frontend / backend

- (At least) one external API call
- Use of framework
- A database-integration containing (at least) user accounts with salted password hashes
- Unit & integration tests
- Docstrings, logging and exception-handling
- Containerization

Peer Review

We are not here to write code or to learn to write code. However, mastering the principles of Software Engineering will involve writing code, and will make you a better coder.

All coding homeworks in this class will be done in pairs, ideally using pair programming (see below). Don't worry if you are not the strongest coder on your team, just contribute to the best of your ability. If you *are* the strongest coder on your team, please try to let your partners contribute.

Pair programming: Pair programming is the practice of having two engineers working together on a single computer, one as the "driver" who is actively using the mouse and keyboard, and the other as the "navigator." You will quickly find out temperamentally which of you is which. This practice is very standard; it is widespread in industry. It is especially effective in pedagogical situations. You will be surprised to find that often neither of you will completely understand it, but that you will still be able to explain it to each other.

I've done this. I know it sounds painful, but it *works*. Really. If there are companies who pay two engineers to use a single computer, they must be seeing a greater than 2x gain in efficiency.

Peer Reviews: One of the objectives of this class is to mimic the experience of working in a professional setting, where collaboration with colleagues you didn't choose is essential.

Your performance as a team will be graded based on your "professionalism." In this section, you'll evaluate your partner on a scale of one to five across several areas and provide a brief statement about your collaboration.

In a real job, your team's performance is visible across the company, while your individual contributions are seen by your team, your boss, and your "skip-boss." This visibility is formalized through annual or bi-annual peer reviews. To reflect this, we have both a team grade and a peer review grade.

To advance in your career, you must be an effective team member contributing to collective success. Success can come in many forms, and you don't need to be the best coder to make meaningful contributions. However, contributing in some way is essential.

With that in mind, you'll assess your partner's communication and follow-through. For example, if your partner was actively engaged, but you ended up correcting their work because they needed guidance, that's a positive outcome. However, if you had to fix their work because they started and then didn't follow through, that is not.

Long story short, to succeed in your career, people have to like you.

ChatGPT

DO NOT LET AI WRITE CODE THAT YOU PUBLISH ON THE INTERNET UNLESS YOU FULLY UNDERSTAND INTERNET SECURITY.

If you use ChatGPT, you must also provide the prompts you used to generate the code. There is a line between

- "ChatGPT, implement these ten different things and talk to me about how to stitch them together"

and

- "ChatGPT, do my final project."

It remains to be seen how thin that line is.

If you do use ChatGPT, we are more than happy to look at any code but we will not look at your prompts (other than to provide guidance about whether your prompts are fine-grained enough to get credit).

I use ChatGPT all the time, but I also know how to tell when it's wrong and what to do about it. Use at your own risk. Also consider that **if all you learn how to do is use LLMs**, **you will not be able to pass a job interview**.
DO NOT LET AI WRITE CODE THAT YOU PUBLISH ON THE INTERNET UNLESS YOU FULLY UNDERSTAND INTERNET SECURITY.

On code reuse

You are expected to use as many resources as you can when doing your assignments. (Before you ask a question, did you google it? Did you check source code we used in class and in the various repos that I've shared? Did you look on Ed Discussion?) It is entirely appropriate to "copy-pasta" blocks of code, so long as you cite your sources. This is done all the time, as you can tell because it has a cute nickname.

You may not turn in someone else's program as your own work, even if it is properly cited. However, you can implement it yourself. If you find a solution you want to use that's more than a few lines of code, you can always translate it into English, and then turn the English back into code without looking at the original source.

Course Chronology

Session 1	Tuesday 9th September	9.30am-12.30pm
Session 2	Monday 15th September	9.30am-12.30pm
Session 3	Tuesday 16th September	9.30am-12.30pm
Session 4	Monday 22nd September	9.30am-12.30pm
Session 5	Tuesday 23rd September	9.30am-12.30pm
Session 6	Tuesday 30th September	9.30am-12.30pm
Session 7	Tuesday 7th October	9.30am-12.30pm
Session 8	Tuesday 14th October	9.30am-12.30pm
Session 9	Tuesday 21st October	9.30am-12.30pm
Session 10	Tuesday 28th October	9.30am-12.30pm
Session 11	Tuesday 4th November	9.30am-12.30pm
Session 12	Tuesday 11th November	9.30am-12.30pm
Session 13	Tuesday 18th November	9.30am-12.30pm

Grading Criteria

Grade	Honour Points	Usual %
A	4.0	93-100
A-	3.7	89-92
B+	3.3	85-88
B	3.0	81-84
B-	2.7	77-80
C+	2.3	73-76
C	2.0	69-72
C-	1.7	65-68
D	1.0	60-64
F	0.0	Unmarked

All work must be completed on time. We also do not allow **'Audits'** (AU), '**Withdrawals'** (W), or **'Pass/Fail'** (P) grades.

The grades reflect the quality of the work. Lecturers and students should use the following criteria for an understanding of what each grade means.

A This exceptional grade is assigned only to work that has persistently outstanding quality in both substance and presentation. The student must demonstrate a sustained capacity for independent thought and extensive study, producing rigorous and convincing analyses in well-ordered prose.

A- Awarded to work that is clearly focused and analytical, and based on wide reading. The student must cover all the principal points of a question and systematically develop a persuasive overall thesis, allowing for one or two venial omissions or inapt expressions.

B+, B, B- This range of grades indicates that the student has shown some evidence of original thought and intellectual initiative. The student has cited sources beyond the class materials, and shown a degree of originality in perception and/or approach to the subject. The work will show thoughtful management of material, and a good grasp of the issues. The differences between a

B+, a straight B and a B- may reflect poor presentation of the material, or mistakes in punctuation, spelling and grammar.

C+, C, C- Work in this grade range is satisfactory, but uninspiring. If the work is simply a recitation of the class materials or discussions, and shows no sign of genuine intellectual engagement with the issues, it cannot deserve a higher grade. Should an essay fail to provide a clear answer to the question as set, or argue a position coherently, the grade will fall within this range.

Quality of presentation can lift such work into the upper levels of this grade range. Work of this quality which is poorly presented, and riddled with errors in grammar, spelling and punctuation, will fall into the lower end of the range. To earn a C grade, the work must demonstrate that the student is familiar with the primary course material, be written well enough to be readily understood, be relevant to the assignment, and, of course, be the student's own work except where properly cited.

D A marginal pass can be given where some but not all the elements of the course have been completed satisfactorily.

F The failing grade indicates the work is seriously flawed in one or more ways:

- Obvious lack of familiarity with the material
- So poorly written as to defy understanding
- So brief and insubstantial that it fails to properly address the subject
- Material presented is not relevant to the assignment
- Demonstrates evidence of plagiarism (see following section in Academic Conduct Code)

Please refer to the Academic Handbook for detailed grading criteria and policies on plagiarism. This can be accessed via Blackboard Learn: http://learn.bu.edu

** Final Grades are subject to deductions by the Academic Affairs Office due to unauthorised absences.*

Artificial Intelligence Guidance

Boston University London faculty may incorporate the use of artificial intelligence (AI) tools in teaching where deemed appropriate for student learning.

However, students should not use generative AI and automated content tools to develop or create any piece of assessed work unless explicitly instructed and permitted by the presiding faculty.

AI tools can provide inaccurate and superficial responses and their inappropriate use undermines the integrity of the academic process.

As use of AI tools in assessed work is prohibited in BU London programmes unless explicitly permitted by the faculty, any suspected unauthorised use will be investigated and may be considered academic misconduct, in line with the University's Academic Conduct Code.

Attendance and Engagement

All students on BU London programmes are expected to familiarise themselves with the full Academic Engagement Policy, which can be found on the personal page or main student Blackboard page.

To receive course credit, students must fulfil all study contact points, including attendance at every class, seminar, exam, presentation or field trip, and placement working days for internship courses, as well as submitting all coursework assignments. Students must report to BU London any instance where they are unable to attend or complete an assignment as scheduled.

Students arriving more than 15 minutes after the posted class start time, or leaving more than 15 minutes early, will be marked as 'late'. Two lates will be treated as one absence.

Religious Holidays

In accordance with the guidance from BU Marsh Chapel & Religious Life, students are required to inform BU London in writing, of conflicts with the course or work schedule and requirements due to their religious observance as early as possible in the semester, and *no later than one week in advance of the conflict*, so that accommodations can be made.

Special Accommodations

Students needing to request accommodations for the semester they are abroad must contact the Disability & Access Services in Boston. BU London can only uphold special accommodations if we have received the appropriate documentation from the BU-DAS. Students will not automatically receive the same accommodations as they do in a regular semester, and we cannot accept letters from other universities/centres.