EE599: Computing and Software for Systems Engineers

Units: 2
Term—Day—Time: **Spring 2020**
Lecture: M 6:30 - 8:30 pm
Discussion: W 6:30 - 8:30 pm

Location: TBD

Instructor: Arash Saifhashemi
Office: TBD
Office Hours:TBD
Contact Info: TBD

Teaching Assistant:
Office: TBD
Office Hours: TBD
Contact Info: TBD

IT Help: TBD
Hours of Service:
Contact Info: TBD

Course Description

The goal of this course is to train EE students to design complete and practical software solutions including both frontend and backend. The course consists of two main parts: (1): Foundations of software engineering. (2): Algorithms and Data Structures with an emphasis on modern software engineering methodologies such as source control and testing using programming languages such as C++, Python, and Javascript.

Assignments are highly analytical and designed to prepare students to design optimized solutions in terms of run time and memory space.

Projects will be close to real-life and industry level problems for which students are asked to provide and implement a complete software solution.

Learning Objectives

The main objective of this course is to broaden job opportunities for our students towards software companies either as their own startups or bigger software companies such as Google, Amazon, etc.

The students will master data structures and algorithms (design, implementation, and analysis) as well as basics of software engineering such as modular design, object oriented design, unit tests, and source control.

Coding skills are of high importance in this course: programming languages such as C++, Python, and Javascript will be covered.

By the end of this course, it is expected that the students will be able to design and implement a complete software solution which typically includes front-end design (web or mobile app) and back-end design (database, cloud implementation, API design).

Prerequisite(s): None
Co-Requisite(s): None
Concurrent Enrollment: None

Recommended Preparation

Participants are expected to have a reasonable degree of mathematical sophistication, and to be familiar with the basic notions of algorithms and data structures, discrete mathematics, and probability. Specifically, the following will be assumed:

- Mathematical Proofs, in particular induction and contradiction.
- Basic data structures: arrays, linked lists, and trees.
- Basic graph theory and graph algorithms: connected components.
- Discrete mathematics: evaluating sums and simple recurrences.

Undergraduate classes in these subjects should be sufficient. If you have doubts about meeting these prerequisites, please contact the instructor.

Course Notes

All students should have an active Github account. You can use your student email address to get free credit for Github and other services such as virtual private servers using this link:

https://education.github.com/pack

Also, for Google cloud, you can use the free plan available here:

https://firebase.google.com/pricing

Technological Proficiency and Hardware/Software Required

Prior experience with the following is recommended:

- Linux operating system
- Common Linux shell commands (cp, mv, ls, find, grep, ...), and shell scripts (bash or tcsh).
- One of the popular IDEs such as Visual Studio Code

Required Readings and Supplementary Materials

Slides and class notes are the main reference. Supplementary material will be provided. The following is optional:

- Algorithm design:
    - Jon Kleinberg/Eva Tardos: Algorithm Design (available at the campus store)

Description and Assessment of Assignments

Most homework assignments are submitted electronically. In most cases we expect a complete working code in C++, Python, or javascript, a brief analysis of your algorithm, its runtime analysis, and some unit tests.

Grading Breakdown

Assessment Tool (assignments)	**Points**	**% of Grade**
Labs and Homework		25
Midterm and Final		45
Project		30
TOTAL		100

Grading Scale

Final grades will be assigned by a combination of student score distribution (curve) and the discretion of the instructor. Final grades are non negotiable. The following scale is just an example and subject to change.

A	95-100
A-	90-94
B+	87-89
B	83-86
B-	80-82
C+	77-79
C	73-76
C-	70-72
D+	67-69
D	63-66
D-	60-62
F	59 and below

Assignment Submission Policy

Homework is assigned every week except for that last few weeks of the semester to allow time for working on the project. Usually, you are asked to give an algorithm or a design, give a proof or the proof sketch of correctness, provide runtime analysis, and provide the implementation and unit tests.

All assignments to be submitted via Blackboard. The details will be provided for each assignment. Late submissions (with penalty) allowed for two days.

Grading Timeline

Grading and solutions to the homework assignments will be provided in about two weeks after the submission deadline. We may not make any solutions available in electronic formats.

Students have two weeks to resolve their grading issues, after the marks are announced.

Project Topics

A team of 2 to 3 students would look at an industrial problem that requires a complete software solution, including front-end and back-end. The topics will be proposed by the instructor, but students can also come up with their own topics provided that they get it approved by the instructor.

Project Requirements and goals: The goal of the project is design and implementation of a complete software system. Your project should cover all of the following topics:

- Frontend design (HTML, CSS, Javascript):
    - Simple UI
        - We only cover web Angular in class, but students can choose to use a mobile platform such as Ionic, react native, or Flutter.
    - Asynchronous function calls (Javascript) to backend
- Backend design (Python or NodeJS):
    - NoSQL Database design and usage
    - Modular design: provide APIs for frontend
    - User and usage analytics
    - Creating automated reports
- Using cloud, NoSQL database, and hosting:
    - The frontend and most of the backend can be hosted on your own laptop, but students are highly encouraged to use one of the freely available VPS providers such as DigitalOcean using this link.

    - Backend: The stored information about users (user id, email address, etc) should be stored on Google Cloud using NoSQL.
- Exception handling
- Source control (Please use github)
- Frontend/backend communication
- Modular design
- Unit tests and documentation

The following is an example topic:

- **PDF Compactor:** Implement a web or mobile application that allows users to upload a pdf file which might be huge (up to 30 MB) and return a compacted version of the file based on the user specified constraints (such as DPI).
    - Frontend:
        - Users should be able to sign up and create a profile using their Facebook or email address.
        - Users should be able to see a history of their previous actions, their quota, and read or delete the files that they have already uploaded.
        - The pdf file and the user constraints should be sent to the frontend using an asynchronous call.
        - Reporting any errors.
    - Backend:
        - The uploaded files should stay on your own (NodeJS or Python) server (either your laptop or one of the free VPSs available using this link).
        - The backend should implement API endpoints to list, download, and remove files for each user. It can be deployed either on your own laptop or one of the free VPSs available using this link.
        - The backend should generate daily analytical summary for the system admin (e.g number of active users per day, number of files converted per day) and email the report to the admin at the end of the day.
        - The backend can call existing library functions for the actual pdf compaction (You don't need to implement this part)
        - The backend should occasionally send emails to users and notify them of new features.
        - The backend should remove all files that are older than 90 days and notify the user.

Project grading, and milestones

- Proposal and topic: 10%
- Phase I (list of API definitions, Database design): 20%
- Phase II (API implementation, Database details, Unit tests) : 25%
- Phase III and Demo: 45%

Project deliverables and demo

- Video:
    - Each team should create a 3 to 4 minute video that clearly describes the system, its design, and implementation. You can upload this on Youtube and provide an unlisted link.
    - Every member of the team should participate in the video, introduce themselves, and discuss the part related to him/her.
- Source code should be submitted by a GitHub link.
- Each team will get about 5 minutes to present their project in front of the class.

Additional Policies

The lecture and discussion sessions are highly interactive and based on in-class collaborations. Attendance is mandatory. The students are not allowed to miss any of the lectures or discussions, unless there is a family or medical emergency. The main source is course slides, notes, blackboard and online discussion forums.

Course Syllabus and Schedule: A Weekly Breakdown

The following syllabus is meant as an outline. Depending on progress, material may be added or removed. Also, there will often be interesting tangents to follow.

- Basics of Software Engineering
  - Version control (git)
  - Unit tests and modular design concepts
  - Exception handling
  - Overview of common programming languages
    - C++
    - Javascript (NodeJS)
    - Python
  - Coding style and readability
  - Introduction to frontend frameworks
    - Web apps (Angular)
    - Mobile apps (Ionic)
  - Introduction to backend design
    - API endpoint implementation using NodeJS
  - Intermediate/Advanced concepts in software engineering
    - C++ STL
    - Asynchronous programming in Javascript
- Foundations in computing (algorithms and data structures)
  - Data structures
    - Trees, linked-lists, hash tables, heaps
  - Algorithms, Complexity theory, and common problems
    - Runtime analysis and Big-Oh notation
    - NP-hardness
    - Greedy Algorithms: Shortest Paths and Minimum Spanning Trees
    - Example problems on graphs – coloring, TSP, Cliques, Clusters
  - Search methods
    - Graph search: DFS, BFS, Topological
    - Binary search and binary search trees
  - Recursion, Dynamic programming and memoization
- Introduction to Modern Cloud Computing
  - Cloud computing (Google Firebase: Cloud functions and NoSQL database)

	Lecture	**Discussion**	**Homework (Topics from Lecture and Discussion of the same week)**
Week 1	Motivation, Introduction, Goals	Setup Bash scripts and shell commands	HW 1 is assigned

	Example: design, algorithm, implementation, analysis Unit tests and Version control	Version control commands (Git)	
Week 2	Big-Oh notation and NP Hardness C++ introduction: classes, member variables and methods. Pointers and references. C++: Functions and operator overloading.	Solved examples in C++ (classes, functions, pointers, and references)	HW 2 is assigned HW 1 is due
Week 3	C++ constructor, copy constructor, and destructor. Functors and Lambda functions. Inheritance, polymorphism	Examples of C++ classes, inheritance and polymorphism	HW 3 is assigned HW 2 is due
Week 4	Data Structures: array, stack, linked list, queue, tree C++ STL (string, list, vector, set, map, algorithm)	Graph representations using STL	HW 4 is assigned HW 3 is due
Week 5	Data Structures: heap, tree and traversal algorithms, graph traversal (DFS, BFS, topological) Greedy algorithms: MST, shortest path	Tree representations using STL STL Priority queues	HW 5 is assigned HW 4 is due
Week 6	Binary search, and binary search trees Sorting algorithms	Summary and review of sorting algorithms	HW 6 is assigned HW 5 is due

Week 7	Recursion, Divide and conquer	Solved examples on recursion and divide and conquer	HW 7 is assigned HW 6 is due
Week 8	Backtracking, dynamic programming and memoization Summary and comparison of shortest path algorithms	Solved examples on backtracking, dynamic programming and memoization	HW 8 is assigned HW 7 is due
Week 9	Review and midterm preparation Project discussion	Midterm	
Week 10	Introduction to Javascript and NodeJS. JSON Asynchronous programming in Javascript	Basics of HTML: headings, paragraphs, lists, buttons, attributes, formatting, links, images, div, span Basics of CSS: inline, internal, external. Fonts, colors, borders, padding, margins.	HW 8 is due Project topic and proposal is due No homework - working on the project
Week 11	Introduction to backend design in NodeJS Introduction to cloud computing (NoSQL database using Google Firebase, Cloud functions)	Debugging using chrome dev tools Unit tests in Javascript	No homework - working on the project
Week 12	Frontend design framework: Angular Frontend-backend communication	More examples on Angular	No homework - working on the project Project Phase II is due
Week 13	Exception handling Introduction to Google cloud NoSQL database	More examples on NoSQL database	No homework - working on the project
Week 14	Introduction to Python. Data types, Functions and argument passing.	Solved problems in Python	No homework - working on the project

	Sequences, Iteration and String Formatting		
Week 15	(slack)	(slack)	Project Phase III is due
Week 16	Project demo	Review	
FINAL			Refer to the final exam schedule in the USC *Schedule of Classes* at classes.usc.edu.

Statement on Academic Conduct and Support Systems

Academic Conduct:

Students are encouraged to collaborate on general solution strategies for homework. The writeup, however, must be your own - you may not copy someone else's solution. In addition, your homework should list all the fellow students with whom you discussed the solutions.

Plagiarism – presenting someone else's ideas as your own, either verbatim or recast in your own words – is a serious academic offense with serious consequences. Please familiarize yourself with the discussion of plagiarism in SCampus in Part B, Section 11, "Behavior Violating University Standards" policy.usc.edu/scampus-part-b. Other forms of academic dishonesty are equally unacceptable. See additional information in SCampus and university policies on scientific misconduct, policy.usc.edu/scientific-misconduct.

Support Systems:

Counseling and Mental Health - (213) 740-9355 – 24/7 on call
studenthealth.usc.edu/counseling
Free and confidential mental health treatment for students, including short-term psychotherapy, group counseling, stress fitness workshops, and crisis intervention.

National Suicide Prevention Lifeline - 1 (800) 273-8255 – 24/7 on call
suicidepreventionlifeline.org
Free and confidential emotional support to people in suicidal crisis or emotional distress 24 hours a day, 7 days a week.

Relationship and Sexual Violence Prevention and Services (RSVP) - (213) 740-9355(WELL), press "0" after hours – 24/7 on call
studenthealth.usc.edu/sexual-assault
Free and confidential therapy services, workshops, and training for situations related to gender-based harm.

Office of Equity and Diversity (OED)- (213) 740-5086 | Title IX – (213) 821-8298
equity.usc.edu, titleix.usc.edu
Information about how to get help or help someone affected by harassment or discrimination, rights of protected classes, reporting options, and additional resources for students, faculty, staff, visitors, and applicants. The university prohibits discrimination or harassment based on the following *protected*

characteristics: race, color, national origin, ancestry, religion, sex, gender, gender identity, gender expression, sexual orientation, age, physical disability, medical condition, mental disability, marital status, pregnancy, veteran status, genetic information, and any other characteristic which may be specified in applicable laws and governmental regulations. The university also prohibits sexual assault, non-consensual sexual contact, sexual misconduct, intimate partner violence, stalking, malicious dissuasion, retaliation, and violation of interim measures.

Reporting Incidents of Bias or Harassment - (213) 740-5086 or (213) 821-8298
usc-advocate.symplicity.com/care_report
Avenue to report incidents of bias, hate crimes, and microaggressions to the Office of Equity and Diversity |Title IX for appropriate investigation, supportive measures, and response.

The Office of Disability Services and Programs - (213) 740-0776
dsp.usc.edu
Support and accommodations for students with disabilities. Services include assistance in providing readers/notetakers/interpreters, special accommodations for test taking needs, assistance with architectural barriers, assistive technology, and support for individual needs.

Campus Support & Intervention - (213) 821-4710
campussupport.usc.edu
Assists students and families in resolving complex personal, financial, and academic issues adversely affecting their success as a student.

Diversity at USC - (213) 740-2101
diversity.usc.edu
Information on events, programs and training, the Provost's Diversity and Inclusion Council, Diversity Liaisons for each academic school, chronology, participation, and various resources for students.

USC Emergency - UPC: (213) 740-4321, HSC: (323) 442-1000 – 24/7 on call
dps.usc.edu, emergency.usc.edu
Emergency assistance and avenue to report a crime. Latest updates regarding safety, including ways in which instruction will be continued if an officially declared emergency makes travel to campus infeasible.

USC Department of Public Safety - UPC: (213) 740-6000, HSC: (323) 442-120 – 24/7 on call
dps.usc.edu
Non-emergency assistance or information.