SYLLABUS (07 Jan 2022)

Applied and Cloud Computing for Electrical Engineers

EE 547: Spring 2022 (2 units)

This course introduces tools and concepts to deploy and maintain full stack software solutions in modern computing environments. This project-driven course guides students through the process of taking ideas from concept to product. The course is intended for graduate electrical engineering students with prior programming experience. This course uses Python and Node.js as primary coding languages. It assumes no prior knowledge of these languages but prior exposure is beneficial. It exposes students to technologies and practices relevant to modern application deployment. The course consists of three main parts: (1) introduction to cloud technologies and integration methods, (2) frontend and backend programming, and (3) deployment within the global computing infrastructure. The course will introduce students to cloud environments and teach cross-service concepts applicable across platforms and within the medium-term future (4-6 years).

Instructor:	Brandon Franzke
Email:	franzke@usc.edu
Office:	EEB 504B
Zoom:	meet: 998 5176 5591
	code: 574987
Hours:	Wednesday: 14:00 – 15:30
	Thursday: 10:00 – 12:00 (remote)

Lecture

Tuesday (section: 31250)
16:00 – 17:50

Discussion

Friday (section: 30404)
14:00 – 14:50[§]

Piazza

https://piazza.com/usc/spring2022/ee541

Piazza enables fast and efficient help from classmates and instructors. Use Piazza to post questions about course material, homeworks, and policies instead of emailing questions to the teaching staff.

Canvas

https://canvas.usc-ece.com

Use Canvas to electronically submit your homework and view course grades. You will receive an email to register during the first week of classes. Contact Dr. Franzke with any technical issues.

Autolab

https://autolab.usc-ece.com

Use Autolab to electronically submit programming portions of homework for "auto-grading". You will receive an email to register during the first weeks of the course. Contact Dr. Franzke with technical issues.

TAs and staff

Grader: Zeyu Wang
Hours: by appointment
E-mail: zwang716@usc.edu

Learning objectives

Upon completion of this course a student will be able to:

- Know how to manage the lifecycle of a software application from concept, deployment, maintenance, and end of life.
- Understand the role of cloud services in a modern application stack.
- Develop client-side programming skills to deploy applications with asynchronous input.
- Develop interactive applications that expose backend state and data storage to distributed clients.
- Distinguish standard databases and apply option(s) that best represent a given data model, cost requirement, or compatibility.
- Ability to work within common cloud platforms, understand their limitations, and how choices affect the scope or reach of their software solution.
- Be comfortable working within virtual or containerized environments and have knowledge to access host level devices.
- Understand how applications exist within and interact with the global computing infrastructure.

Course materials

- "*Mastering Node.js*", 2nd edition, Sandro Pasquali and Kevin Faaborg, Packt Publishing, 2017. (*required*)
- "*Designing Data-Intensive Applications*", Martin Kleppmann, O'Reilly Media, 2017. (*required*)
- "*Release It!: Design and Deploy Production-Ready Software*", 2nd edition, Michael Nygard, Pragmatic Bookshelf, 2018. (*optional*)
- "*Speaking JavaScript*", 1st edition, Axel Rauschmayer, O'Reilly Media, 2014. (*required*)
- "*The Road to GraphQL*", Robin Wieruch, 2018. (online: https://github.com/the-road-to-graphql/the-road-to-graphql)

Note: The texts are secondary to in-class lecture material and homework sets.

Course Outline (tentative)

	Topics/Daily Activities	Homework	Deliverables
Week 1 (11 Jan)	Architecture (local vs. distributed), containers, virtualization, and cloud PaaS. Walkthrough: Create a backend.	HW 1 assigned	
Week 2 (18 Jan)	Language overview: JavaScript and Node.js.	HW 2 assigned	HW 1 due.
Week 3 (25 Jan)	Backend development: Node.js and express middleware.		
Week 4 (01 Feb)	Databases I: NoSQL. Mongo.	HW 3 assigned	HW 2 due
Week 5 (08 Feb)	Databases II: Relational. MySQL, Maria, and Postgres.		
Week 6 (15 Feb)	Databases III: Graph and specialized.	HW 4 assigned	HW 3 due
Week 7 (22 Feb)	AWS console and CLI. Walkthrough: VPC stand-up.		
Week 8 (01 Mar)	Walkthrough: VPC standup and application deployment		HW 4 due Preliminary proposal due
Week 9 (08 Mar)	**Midterm Exam (16:00 - 17:30)**		
(15 Mar)	**No class, Spring break, University holiday.**		
Week 10 (22 Mar)	Backend API patterns: REST vs GraphQL.	HW 5 assigned	Revised proposal due
Week 11 (29 Mar)	Frontend overview: HTML, CSS, JavaScript		
Week 12 (05 Apr)	Frontend development and application frameworks	HW 6 assigned	HW 5 due Phase 1 status due
Week 13 (12 Apr)	Serverless compute, cloud storage. Application scalability.	HW 7 assigned	HW 6 due
Week 14 (19 Apr)	Authentication and Access control		
Week 15 (26 Apr)	Lifecycle: testing, continuous deployment, maintenance		HW 7 due Phase 2 status due
FINAL (10 May)	**Project presentations, 16:30 - 18:30 (manadatory)**		
(11 May)	**Project reports and videos due**		

Attendance and Participation

Attendance is mandatory to all lectures and discussions. You are responsible for missed announcements or changes to the course schedule or assignments. Taping or recording lectures or discussions is strictly forbidden.

Grading Procedure

Homework

Homework is assigned every 1-2 weeks. Assignments include a mix of applied, analytical, and computational problems. Your total homework score sums your best homework scores (as a percentage) after removing the one lowest score (of minimum 50%). Late homework will be accepted with a 10% deduction per 24-hours for up to 48-hours. You may discuss homework problems with classmates but each student must do their own original work. Cheating warrants an F in the course. Turning in identical homework establishes a rebuttable presumption of cheating.

Midterm Exam

You may use a single 8.5"x11" reference sheet (front and back OK). You may not use any additional resources. The midterm exam will include multiple-choice and/or short answer questions to demonstrate progress toward the learning objectives. It will also include free-response or open-ended questions to demonstrate comprehensive mastery. You may also be asked to determine expected behavior of novel computer code. Students are expected to write correct code (abstract pseudo-code, Node.js, Python, etc.) as well as have familiarity with Bash scripts. Exam grading primarily follows correct reasoning but may include deductions for major syntax errors, algorithmic inefficiency, or poor implementation. You must show how you arrived at your answers to receive full credit. You are expected to bring a non-graphing scientific calculator.

Final Project

This course culminates with a final project in lieu of a final exam. Teams of three students (teams of two with instructor approval) design and implement a complete software product that connects two or more independent asynchronous components (often "frontend" and "backend"). The instructor will guide teams with difficulty identifying a suitable application. Teams may build an application similar to existing services or tools but their must efforts demonstrate understanding of the entire development stack and the product lifecycle from idea to deployment to maintenance. Though teams are encouraged to devise problems of particular interest to their backgrounds, interest, or research. All projects must obtain the instructor's written approval. Teams will prepare and present their approved project and show how it applies course material, concepts, and best-practices. Attendance and participation during the project presentation session(s) are mandatory.

Requirements

Project topics must include sufficient scope and apply course knowledge to a useful end. The project must compose at least two distinct units that operate and act independently but provide greater function when acting together. The project must demonstrate comprehensive understanding of the entire development stack and the product lifecycle from idea to deployment to maintenance. You may use whatever computer language you like but deviations from Python, C++, Node.js, GoLang, or other frameworks used in class require prior instructor approval.

Grading and Milestones

Topic Proposal	week 8	10%
Phase 1 – Design, components, classes, and tests	week 13	15%
Phase 2 – Integration and deployment	week 15	15%
Demo and presentation	final	25%
Project report and video		35%

Deliverables and demo

Written project report: the project report should summarize the topic, provide relevant background (theoretical or applied), timeline and contributions, and document challenges and extensions. It should provide discussion sufficient that an uninformed expert could understand the logic, algorithmic decisions, and implementations. Teams should provide quantifiable metrics to justify engineering tradeoffs.

Presentation: Approximately 10 minute (depends on class size) presentation to describe to the class their topic problem and their solution. It should provide only what is necessary to understand the "what" and "why" and include minimal theoretical background.

Video: 3-4 minute video that describes the problem, your design, and implementation. You may choose to upload this to a video sharing site such as YouTube but that is not required. All team members must participate equally.

Source code: submitted to instructor by providing link to pull from github.

Example projects

Neural network platform UI: Design and implement a frontend and backend to interface an existing 3rd party Maching Learning API such as Amazon SageMaker or Microsoft Azure ML. It should be a responsive design and deliver a premium user experience. The application may allow or enforce different policies based on authentication, provide visualization of models or results in addition to the platform tools. It may also integrate information from multiple services such billing and quota.

BigData insight tool: Design a tool suite (frontend) that integrate with a potentially large backend dataset. The frontend should expose a UI that lets a user validate multiple hypotheses. The backend should cache and mange the datastore in a way to provide an optimal user experience. The tools might include a rudimentary statistics engine to perform regressions or estimation and may also report abnormal cases or data that violates common statistical assumptions such as zero-correlation, homoscedasticty, and non-normality.

Course Grade

Homework	45%
Midterm Exam	25%
Final Project	30%

A	if 90 – 100 points
B	if 80 – 89 points
C	if 70 – 79 points
D	if 60 – 69 points
F	if 0 – 59 points

("+" and "–" within approx. 3% of grade boundary)

Cheating

Cheating is not tolerated on homework or exams. Penalty ranges from F on assignment or exam to F in course to recommended expulsion.

Academic Conduct

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 Section 11, Behavior Violating University Standards https://scampus.usc.edu/1100-behavior-violating-university-standards-andappropriate-sanctions. Other forms of academic dishonesty are equally unacceptable. See additional information in SCampus and university policies on scientific misconduct, http://policy.usc.edu/scientific-misconduct. Discrimination, sexual assault, and harassment are not tolerated by the university. You are encouraged to report any incidents to the Office of Equity and Diversity http://equity.usc.edu or to the Department of Public Safety http://capsnet.usc.edu/department/department-public-safety/online-forms/contactus. This is important for the safety of the whole USC community. Another member of the university community – such as a friend, classmate, advisor, or faculty member – can help initiate the report, or can initiate the report on behalf of another person. The Center for Women and Men http://www.usc.edu/studentaffairs/cwm/ provides 24/7 confidential support, and the sexual assault resource center webpage http://sarc.usc.edu describes reporting options and other resources.

Academic Integrity

Academic integrity is critical the assessment and evaluation we perform which leads to your grade. In general, all work should be your own and any sources used should be cited. Gray-areas occur when working in groups. Telling someone how to do the problem or showing your solution is a VIOLATION. Reviewing examples from class or other sources to help a fellow classmate understand a principle is fine and encouraged. All students are expected to understand and abide by these principles. SCampus, the Student Guidebook, contains the University Student Conduct Code in Section 10, while the recommended sanctions are located in Appendix A. Students will be referred to the Office of Student Judicial Affairs and Community Standards for further review, should there be any suspicion of academic dishonesty.

Support Systems

A number of USC's schools provide support for students who need help with scholarly writing. Check with your advisor or program staff to find out more. Students whose primary language is not English should check with the American Language Institute http://dornsife.usc.edu/ali, which sponsors courses and workshops specifically for international graduate students. The Office of Disability Services and Programs http://sait.usc.edu/academicsupport/centerprograms/dsp/home_index.html provides certification for students with disabilities and helps arrange the relevant accommodations. If an officially declared emergency makes travel to campus infeasible, USC Emergency Information http://emergency.usc.edu will provide safety and other updates, including ways in which instruction will be continued by means of blackboard, teleconferencing, and other technology.

Academic Accommodations

Any student requiring academic accommodations based on a disability is required to register with Disability Services and Programs (DSP) each semester. A letter of verification for approved accommodations can be obtained from DSP. Please be sure the letter is delivered to me as early in the semester as possible. DSP is located in GFS 120 and is open 08:30 – 17:00, Monday through Friday. The phone number for DSP is (213) 740-0776.