SYLLABUS

Applied and Cloud Computing for Electrical Engineers

EE 547: Spring 2023 (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).

Lecture

Tuesday (section: 31250)
17:00 – 18:50

Discussion

Friday (section: 30404)
14:00 – 14:50

Enrollment is in-person ONLY. Attendance is mandatory to all lectures and all discussions. Taping or recording lectures or discussions is strictly forbidden without the instructor's explicit written permission.

Instructor:	Brandon Franzke	
Email:	franzke@usc.edu	
Office:	EEB 504B	
Hours:	Wednesday:	14:00 – 15:30 (in-person, remote)
	Thursday:	12:00 – 14:00 (remote only)

Course materials

- "*Node.js Design Patterns*", 3nd edition, Mario Casciaro and Luciano Mammino, Packt Publishing, 2020. (*required*).
- "*Cloud Native Patterns: Designing change-tolerant software*", 1st edition, Cornelia Davis, Manning, 2019. (*required*).
- "*Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement*", 2nd edition, Luc Perkins, Eric Redmond, and Jim R. Wilson, Pragmatic Bookshelf, 2018. (*required*).
- "*Release It!: Design and Deploy Production-Ready Software*", 2nd edition, Michael Nygard, Pragmatic Bookshelf, 2018. (*optional*).
- "*Eloquent JavaScript*", 3rd edition, Marijn Haverbeke, No Starch Press, 2018. (online: https://eloquentjavascript.net/index.html).
- "*Speaking JavaScript*", 1st edition, Axel Rauschmayer, O'Reilly Media, 2014. (online, updated: http://speakingjs.com/).

- "*Learning JavaScript Design Patterns: A JavaScript and jQuery Developer's Guide*", 1st edition, Addy Osmani, O'Reilly Media, 2012. (online, updated: https://www.patterns.dev/posts/classic-design-patterns/).
- "*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.

Teaching assistants

TA:	Xiou Ge	Grader/CP:	Lifuling Wei
Hours:	TBA	Hours:	by appointment
Email:	xiouge@usc.edu	E-mail:	lifuling@usc.edu

Piazza https://piazza.com/usc/spring2023/ee547

Canvas https://canvas.usc-ece.com

Electronically submit homework and view grades. You will receive a registration email during the first week of classes. Contact Dr. Franzke with technical issues.

Autolab https://autolab.usc-ece.com

Electronically submit programming homework for *auto-grading*. You will receive a registration email during the first week of classes. Contact Dr. Franzke with technical issues.

Learning objectives

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

- Understand the role of cloud services in a modern application stack.
- Develop interactive applications that expose backend state and data storage to distributed clients.
- Develop client-side programming skills to deploy applications with asynchronous input.
- 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.
- Manage the lifecycle of a software application from concept, deployment, maintenance, and end of life.
- 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 Outline (tentative)

	Topics	Homework	Deliverables
Week 1 10 Jan	Architecture (local vs. distributed), containers, virtualization, cloud PaaS. Demo: create a backend.	HW 1 assigned.	
Week 2 17 Jan	JavaScript and Node.js.	HW 2 assigned.	HW 1 due (21 Jan).
Week 3 24 Jan	Backend development: Node.js, express middleware.		
Week 4 31 Jan	Databases I: NoSQL. Document. Mongo.	HW 3 assigned.	HW 2 due.
Week 5 07 Feb	Frontend overview: HTML, CSS, JavaScript.		
Week 6 14 Feb	Databases II: Relational. MySQL, Maria, and Postgres.	HW 4† assigned.	HW 3 due.
Week 7 21 Feb	**Quiz #1 (weeks 1-5).**		
Week 8 28 Feb	AWS console and CLI. Walkthrough: VPC stand-up.		HW 4 due.
Week 9 07 Mar	Demo: VPC standup and application deployment	HW 5 assigned.	
(13 Mar)	**No class, Spring Break.**		
Week 10 21 Mar	Frontend development and application frameworks		HW 5 due.
Week 11 28 Mar	Backend API patterns: REST vs GraphQL.	HW 6 assigned.	Preliminary proposal due (27 Mar). Proposal return (31 Mar).
Week 12 04 Apr	Project meetings		Revised proposal due (10 Apr)
Week 13 11 Apr	Serverless, cloud storage. Application scalability.		HW 6 due.
Week 14 18 Apr	**Quiz #2 (weeks 6-13).** Authentication and Access control. Consuming Oauth. Project status report due (21 Apr).		
Week 15 25 Apr	Lifecycle: testing, continuous deployment, maintenance		

04 May	**Technical review and demos, 15:30 - 18:30.**
08 May	**Project deliverables, due 12:00.**

Grading Procedure

Homework

Homework is assigned every 2-3 weeks. Assignments include a mix of applied and programmatic 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 submit their own original work. Cheating warrants an "F" on the assignment. Turning in identical homework establishes a rebuttable presumption of cheating.

Quizzes

Quizzes are short (60-minute) non-cumulative assessments that cover the most recent material (approximately 5-weeks). Quizzes highlight important concepts and methods. They occur during weeks 7 and 14 (tentative). You may use a single 8.5"x11" reference sheet (front and back OK). You may not use any additional resources. Quizzes will include multiple-choice and/or short answer questions to assess progress toward the learning objectives. It will also include free-response or open-ended questions to demonstrate conceptual understanding. You may be expected to write reasonably correct code or determine expected behavior of novel computer code. Exam grading primarily follows correct reasoning but may include deductions for major syntax errors, algorithmic inefficiency, or poor implementation.

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 are encouraged to devise solutions to novel problems of personal interest to their background or research. But teams may build an application similar to existing services or tools provided their efforts demonstrate understanding of the development stack and the product lifecycle — from idea to deployment to maintenance. All projects must obtain the instructor's written approval. Teams will prepare and present/demo their approved project and show how it applies course material, concepts, and best-practices.

Course Grade

Homework	45%
Quizzes	20%
Final Project	35%

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 "–" at ≈ 2.5% 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.

Final Project

Project 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. Additional requirements and guidelines will be discussed closer to the commencement of the project.

All projects must use Node.JS as the primary language unless approved explicitly in writing by the instructor. But projects may use additional languages for tooling and support. The instructor may provide additional requirements when introducing the final project assignment.

Grading and Milestones

Topic Proposal (initial and revised)	week 11	14%
Status report - Design, components, integration	week 14	8%
Technical review and demo	final	30%
Project report		24%
Design and source code		20%
Video		4%

Deliverables and demo

Topic proposal: describe the problem, proposed technical approach, and expected outcomes. It should communicate that your topic is adequately prepared and it should outline immediate next steps. But the proposal is merely a guidepost and reasonable deviations in method, approach, and scope are expected.

Written report: 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 can understand the models, analytic decisions, outcomes, and implementation. Teams should provide quantifiable metrics to justify engineering tradeoffs.

Technical review and demo: Approximately 15 minutes (depends on class size) to describe the topic problem and solution. It should provide only what is necessary to understand the *what* and *why* and include minimal theoretical background. The instructor will provide a *technical reference* slide-deck template that must be completed in advance of the demo session.

Source code: submitted as a GitHub repository archive file (zip). It must include README file(s) that describe the repository structure, execution instructions, and special technical requirements.

Video: a 4-minute video that describes the topic, your implementation, and your results. You may choose to upload this to a video sharing site such as YouTube but that is not required.

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.