SE/CS575 Course Description & Syllabus

INSTRUCTOR: Dr. Brian Mitchell - twitter

General Information

When designing a significant modern software system, several design decisions about the structural, architectural, runtime and behavioral properties of the system are made and documented. In this course, techniques and notations are introduced for specifying these properties of software systems. Software systems are studied at various levels of abstraction from architectures to subsystem decompositions to module and class interfaces and dependencies.

In this course, students will learn to analyze, express, and implement software designs. The course will focus on the understanding the design needs of modern systems:

1. That must run at scale,
2. Require deployment flexibility (cloud, containers, etc),
3. Are highly distributed,
4. Run in potentially hostile or untrusted environments,
5. Promote design flexibility that enables rapid adoption of user-driven changes.

The course will not use a textbook - lectures will be formulated from material developed in academia and industry on Software Architecture and Software Design. We will be investigating architecture best practices though a number of modules that dive into many aspects of modern software design:

1. Modern SOA (REST, including async and gRPC)
2. Modern Web Architecture (SPA deep dive using the Google Angular Framework)
3. Reactive Architecures (including the reactive manifesto)
4. Pragmatic functional programming using Typescript, Kotlin, and Scala (sorry, I personally dislike Java so we will not be using it much in this course)
5. Eventing and Streaming
6. Using the right tool for the right job - principles of polyglot
7. Cloud Native Architectures (including cloud native runtimes such as docker/kubernetes)
8. If time permits, DevOps engineering concepts like CI / CD
9. If time permits, Designing for the Cloud, including serverless architecture concerns

Course Structure

This is an 11 week graduate course, weekly lectures will be provided via my GitHub site @ArchitectingSoftware. Course assignments and deliverables will be managed on Drexel's Learn Blackboard site. In addition to the course lectures, student participation and deliverables will include:

1. Independent review of research papers that will be posted on the class internal discussion board, I will be selecting 5-6 papers that range in publication date from 1972 to 2018.
2. NEW in 2018 - Feedback to the speaker. I will be selecting several YouTube accessible videos from some of my favorite speakers and ask students to create a short writeup critiquing the talk. The student will be expected to integrate learnings in the class with the talk to structure a virtual feedback email.

3. Midterm assignment. Students will design an IoT device. This course component has 2 deliverables:
    - Develop a reference architecture for IoT devices
    - Specify 5 features for a new IoT enabled outdoor light and then create an architecture description for the new product.
4. Final project is to construct a non-trivial implementation of a project of your choosing (or from a list that I will provide), demonstrating your ability to apply some of the foundational concepts that we covered in class. This project:
    - Can be individual or preferably team based
    - Must be hosted on a personal GitHub account
    - Must include full automation to build and deploy your solution.
    - Must include a link to an online demonstration of your solution that cannot exceed 5 min in length

Requirements

In order to be successful in this course:

- Students are expected to have some previous software engineering experience, either from a recent undergraduate course or from practical work experience.
- Students are also expected to have the ability to write source code in a modern programming language, including surrounding skills working with a git-based SCM repository and build/dependency tools. Examples: Java, modern Javascript, Python, C#, GitHub, git, gradle, maven, npm, etc.
- Students must be comfortable being exposed to source code in programming languages that they might not know. I will be reinforcing course concepts by showing and explaining sample code in a variety of programming languages (Scala, Kotlin, GoLang, TypeScript, modern JavaScript, etc) throughout this course. While I dont expect students to know all of these languages, I do expect that they can follow examples in multiple programming languages when explained.
- Students should also have a basic knowledge of working with web technologies such as HTML, and Javascript. While only a limited basic working knowledge is required, I use web architectures a good bit in this course to highlight key modern software design and architecture concepts.

Grade Determination

Student final grades will be derived by the approximate scale:

Course Component	**Approximate Weight**
Discussion Group Assignments, and Research Paper Reviews	35%
Assignments based on YouTube videos	15%
Midterm Assignment	15%
Final Project	35%

Note that all course deliverables will be submitted via the course Blackboard site and not email.

Last updated: October 2018