0:41

Hi, and welcome to the Coursera specialization on Software Design and Architecture. I'm Ken Wong. It's my goal to help you become an experienced software architect who thinks critically about the design and architecture of your products in order to create great software. I'll be doing this with the help of our learning navigator, Sam Jeffery. You'll meet her in a moment. You've probably already worked on some software projects in the past. You may have worked on small programs, so you may worked on a larger scale system. Take a minute and think of the projects that you worked on. Did they have a good design? Could the design be done better? Was there even a design at all? How do you know if the software was well-designed? Think of how easy it was to make changes to your code. Did a small code change produce a ripple-effect for changes elsewhere in the code? Was your code hard to reuse? Was the software difficult to maintain after a release? If you answered yes to any of these questions, chances are you could benefit from a better design. Good design isn't just about code. It is about being able to express ideas for your software with other developers, other teams, and your clients. Having a well-thought design makes your software easier to implement, reduces a need for major changes later or the project and it saves you from headaches down the line. Whether you're a new developer looking to expand your knowledge or an experienced industry veteran, knowledge of Software Design Architecture will help your software become flexible, reusable, and maintainable. Hi, and welcome to the Coursera specialization on Software Design and Architecture. I'm Sam Jeffery, your learning navigator through this specialization. It's my goal to help you become an experienced software architect who thinks critically about the design and architecture of your products in order to create great software. So, what is software design and architecture? How does it improve your software products? Consider this scenario. You join a project that's been in development for a while. You look at the code and become instantly overwhelmed. You can't tell what the purpose of the pieces are, things are unorganized, and design documentation is non-existent. You don't even know where to begin. These are all signs that the project was not well-designed from the outset. Or let's say that you are now working on a personal development project. When you began, you weren't quite sure what the functionalities would be, but you just started coding. It didn't matter that the code was unorganized because you were the only one working on it and you know how it works. You come up with a great new feature for your product, but in implementing it, you broke the program elsewhere. You should have designed it right. I'm guessing that you have experienced scenarios like these. They are quite common in the software industry, which shows you why software design and architecture is so beneficial. In this specialization, you will learn how to apply design principles, patterns, and architectures to create reusable and flexible software applications and systems. You will learn how to express in document the design and architecture of a software system using a visual notation. We will give you lots of practical examples and opportunities for you to apply this knowledge that you leave this specialization with employable skills and relevant knowledge to confidently work in the software industry. It's important to note that this specialization primarily covers software design and not user-interface design. Although the two aspects of design are closely related, we will mostly focus on software design. We have tried to align our course with the ACM/IEEE Software Engineering Curriculum Guidelines. It recommends the topics when teaching software design and architecture. However, with that being said, this specialization has no affiliation with ACM or IEEE. I provided a reference to these guidelines in the course resources if you'd like to read more about it. This specialization consists of four courses. Each course features four modules with the first three focused on content and initial parts of the capstone project and the fourth to complete the rest of the capstone project. The capstone projects will give you the opportunity to apply the knowledge that you learn in each course to a real functional software product. In the first course, you will learn about object-oriented design. This course will build upon the basics of Java and take you to the next level by covering object-oriented analysis and design. You will discover how to create flexible, reusable, and maintainable software by applying object oriented design principles. You will learn how to communicate these designs by expressing them in a visual notation known as Unified Modeling Language or UML. In the first capstone project, you will be challenged to apply your knowledge of object-oriented design by evolving and documenting a Java codebase with corresponding UML documentation. The second course focuses on design patterns. Design issues and applications can be resolved through design patterns commonly applied by experts. The second course extends your knowledge of object-oriented analysis and design by covering design patterns used in interactive applications. Through a survey of established design patterns, you will gain a foundation for more complex software applications. Finally, you will learn how to identify problematic software designs by referencing a catalog of code smells. In the second capstone project, you will be challenged to redesign an application to use design patterns. The third course covers software architecture. By the third course, you will be equipped with software design patterns and principles and will be ready to learn how architecture can be used as a basis for organizing the software systems found in industry today. Through a review of architectural styles, you will explore the structure and behavior of large scale software systems. Specific UML diagrams will express important architectural perspectives. You will learn how to analyze and evaluate a given architecture by examining the trade-offs between competing quality attributes such as modifiabilty and performance. In the capstone project in this course, you will document and evaluate the architecture of a layered Java system. Finally, in our fourth course, you will learn about service-oriented architecture. Based on the understanding of architectural styles that you learned in the third course, you will review architectures for web applications. You will then explore the basics of service-

oriented architecture or SOA in two approaches, web services and representational state transfer or REST architecture. In the final capstone project, you will connect a mobile application with a variety of REST interfaces to access storage, search, and computation services. By the end of this specialization, you'll know how to design it right. Our courses are designed to be taken in sequential order. However, the courses do stand alone. If there was a specific area of software design and architecture that you were interested in learning about, you could just take that course. In this introductory course and throughout the specialization, we will assume that you possess basic Java programming knowledge. In this module, there is a Java pre-test that you can take to see if your Java programming skills are sufficient to succeed in the specialization. The pre-test is not meant to discourage you from taking our specialization, but is intended to set you up for success by identifying areas that you may need to review before taking this specialization. We understand that our learners will come from various backgrounds and experience levels. In order to make sure that all our learners are on the same page, we have tried to be explicit when introducing new terms. Based on your experience level, you might find that you should complete additional readings to fully grasp the topics or if you are an experienced developer, you may find that some topics you already know. In a specialization of this size, it's difficult to ensure the material is appropriate for everyone. We've done our best to provide you with the resources you need to understand the material. We've found that even if you are an experienced developer, there's still a lot to learn. So, we encourage you to engage with the lessons even if you've covered the material before. Who knows? Maybe the material has changed since you learned it. I also encourage experienced learners to share their expertise in the discussion forums. Let's build a strong community of mentorship and camaraderie. Well, I think you're ready to embark on this journey. So once again, welcome to the Software Design and Architecture specialization brought to you in partnership by the University of Alberta and Coursera. I hope you enjoy it and I wish you the best of luck.