MODULE 1

Summary & Highlights

Congratulations! You have completed this module. At this point, you know:

- Software engineering is the application of scientific principles to the design and creation of software.
- Responsibilities of a software engineer include designing, building, and maintaining software systems.
- Using the SDLC can improve efficiency and reduce risks by:
  - letting team members know what they should be working on and when
  - facilitating communication between the customer, other stakeholders, and the development team
  - letting stakeholders know where they fit into that process and
  - letting cross-domain teams know when they have completed their tasks so development can move to the next phase.
- Common software engineering processes are requirements gathering, design, coding, testing, releasing, and documenting.
- The requirement gathering process entails identifying stakeholders, establishing goals and objectives, eliciting requirements from the stakeholders, documenting the requirements, analyzing, prioritizing, and confirming the requirements.
- An SRS is a document that captures the functionalities that the software should perform and also establishes benchmarks or service levels for its performance.
- A URS is a subset of the SRS that details user specification requirements.
- The SysRS contains the same information as an SRS, but can also additionally include system capabilities, interfaces, and user characteristics, policy requirements, regulation requirements, personnel requirements, performance requirements, security requirements, and system acceptance criteria.
- Waterfall, V-shape model, and agile are all different methodologies for implementing the software development life cycle.

- Functional testing is concerned with inputs and corresponding outputs of the system under test, non-functional testing tests for attributes such as performance, security, scalability, and availability. Whereas regression testing confirms that a recent change to the application, such as a bug fix, does not adversely affect already existing functionality.
- Types of documentation include requirements, design, technical, quality assurance, and user.
- There are many different roles involved in a software engineering project. Some of them include project manager or scrum master, stakeholder, system or software architect, UX designer, software developer, tester or QA engineer, site reliability or Ops engineer, product manager or owner, and technical writer or information developer.

Go to next item

MODULE 2

Summary & Highlights

Congratulations! You have completed this module. At this point, you know:

- How websites are built and displayed, and how they communicate with the back-end servers.
- How different front-end technologies work together to create reactive and responsive websites.
- How back-end development covers a wide range of technologies including business logic, security, and database access.
- Effective teamwork can result in better quality code with fewer bugs, better-skilled team members, and less stress for everyone.
- Pair programming is a great way to share knowledge and skills between developers, resulting in better solutions and improved efficiency.
- You can use developer tools to track who makes what changes to your code withversion control software, access libraries of reusable code, and use frameworks to build and deploy applications in a standard way.

- CI/CD tools, build tools, packages, and package managers help you build and distribute your applications.
- A software stack is a combination of technologies for creating applications and solutions.

MODULE 3

Summary & Highlights

Congratulations! You have completed this module. At this point, you know:

- Interpreted programming languages create source code that runs through an interpreter and is built into your operating system (OS) on your computer or on your web browser.
- Compiled programming languages create executable files that are grouped in programs on your computer or device.
- Query languages, structured programming languages, and object-oriented programming languages are categorized as high-level programming languages and assembly languages are categorized as low-level programming languages.
- The two main methods of organizing and planning code are by developing flowcharts and by writing pseudocode. Flowcharts are pictorial representations of algorithms and pseudocode is an explanation of the function of each line of a program.
- To reference a program component, software developers use an identifier, which can either be a constant or a variable.
- A function is a piece of structured, stand-alone, and reusable code that will perform a single specific action.
- Object-oriented programming is a programming methodology that is focused on objects rather than functions.

MODULE 4

Summary & Highlights

Congratulations! You have completed this module. At this point, you know that:

- Software architecture functions as a blueprint and represents the importance of a good architectural design.
- Structured design breaks down a software problem into well-organized smaller solution elements whereas behavioral models describe the behavior of the system without explaining how the system implements the behavior.
- Developing UML diagrams saves time and money by helping developers quickly get up to speed on a project, plan features in advance of coding, and navigate source code easily. Types of UML diagrams include state transition, interaction, and class diagrams.
- Objects contain data, and they also have behaviors that prescribe the actions the object can take, whereas classes are blueprints for objects.
- A service-oriented architecture (SOA) consists of loosely coupled services that interface with each other via a communication protocol over a network. Distributed systems run on multiple services on different machines, but they appear to the end-user as a single coherent system.
- An architectural pattern is a repeatable solution to an architectural problem. Types of architectural patterns include 2-tier, 3-tier, event-driven, peer-to-peer, and microservices. Two or more patterns can be combined in a single system, but some are mutually exclusive.
- Application environments include development, testing or QA, staging, and production. Production environments tend to be more complex than pre-production because they must take into account non-functional requirements like load, security, reliability, and scalability.
- Application environments can be deployed either on-premises on traditional hardware, or on public, private, or hybrid cloud platforms.
- Common components needed for a production environment include a firewall, a load balancer, web and application servers, proxy servers, and database servers.

MODULE 5

Summary & Highlights

Congratulations! You have completed this module. At this point, you know:

- Software engineers design and develop software solutions and maintain and update existing software.
- Learning on the job is a key part of a software engineering role.
- A combination of hard and soft skills is essential for the role of a software engineer.
- There is high demand for software engineers in flexible and satisfying roles.
- A career in software engineering often takes a technical or managerial path, but software engineering skills can also apply to a wide variety of other roles.
- There are many different job titles under the umbrella term of software engineer, each of which has a specific set of skills and responsibilities.
- The software engineering code of ethics contains eight principles: public, client/employer, product, judgment, management, profession, colleagues, and self.