https://anywhere.epam.com/en/blog/software-developer-interview-questions-answers

1. What is a framework?

A framework is a tool that gives software developers access to prebuilt components or solutions designed to expedite development.

As a knowledge-based question, a simple answer demonstrating your knowledge is generally enough.

2. Name the stages of the software development lifecycle (SDLC)

The stages of the software development life cycle are:

- Planning
- Requirements gathering and analysis
- Design
- Coding and implementation
- Testing
- Deployment
- Maintenance

Like the question above, this one tests your knowledge. As long as you outline the stages, that's usually sufficient.

3. Compare waterfall and agile models and provide examples of their use cases

The waterfall methodology is a sequential process where tasks are handled in a linear fashion. Generally, it's best used when the requirements are clear, well-known, and entirely fixed.

The agile methodology uses an iterative process that relies on cyclic patterns with a high degree of collaboration. Agile provides ample room for feedback and future adjustments, making it a better fit in cases where goals and requirements may shift, or other unknowns are likely to arise.

With this answer, you can dive into examples from your past work to serve as use cases. That can add something extra to your response and may help you stand out.

4. What is refactoring?

Refactoring is the process of taking care of existing code and restructuring it, typically to improve the code through small changes without altering its underlying behavior.
With your answer, you can give an example of a previous experience that shows you know how to refactor code the right way. However, a simple definition is also enough.

5. How do functional requirements differ from non-functional ones?
Generally, functional requirements define a system's operation, while non-functional requirements outline how a solution should perform or behave.
One way to make this answer more impactful is to use a past project as an example. Discuss its functional and non-functional requirements, showing you understand the difference beyond the definitions.

6. Explain the concept of object-oriented programming (OOP)
Object-oriented programming is a model that centers on data fields with distinct behaviors and attributes — referred to as objects — instead of logic or functions. Developers focus on the objects that need to be manipulated instead of the processes required to manipulate them.
Like the question above, you can use examples from past projects to make your answer more compelling and show your understanding of differences between functional programming and OOP.

7. Have you ever created unit tests?
Unit testing involves taking small parts of an application — referred to as units — and independently scrutinizing them to ensure correct operation.
Since this question is experience-related, you'll want to confirm whether you have experience in this arena. If so, provide an example from your past work where you used unit testing to achieve a goal.

8. What debugging tools do you use?
Here are some common debugging tools you might discuss:

Affinic

GDB
LLDB
Radare2
Valgrind
WinDbg

This is another question that focuses on individual experience. You'll want to outline which debugging tools you've used during your career as a starting point. If you're familiar with several, discussing use cases for each one can fill your answer out a bit.

9. What are the OSI model layers?

The Open Systems Interconnection (OSI) model layers are:

Physical, transmitting raw bit data over a physical medium
Data Link, defining the data format
Network, defining the physical path for the data
Transport, transmitting data using protocols
Session, responsible for ports and sessions control
Presentation, displaying data in a usable format
Application, enabling human-computer interaction

10. Name API architectural approaches

Here is an overview of some API architectural styles:

- Event-driven
- Hypermedia
- Pragmatic REST
- Web API

11. What is CORS?

CORS, or cross-origin resource sharing, is an HTTP-header-based mechanism that lets a server define origins outside of itself where the browser should allow the loading of resources. Within CORS is a mechanism that ensures browsers reach out before a user request to confirm the server will support the request for the cross-origin resource.

If you have an example from a past project that involved CORS, discuss it to give your answer more depth.

12. What software security protection methods do you know?

There are numerous software protection methods. Some options you could discuss include:

- Code signing certificates
- Error handling
- Hashing passwords
- Input sanitization
- User authentication
- Whitelisting

With this question, you'll want to outline your personal experience regarding software security. The phrasing of the question is broad enough to allow you to steer your answer in any appropriate direction.

13. What is virtual DOM?

Virtual DOM, also referred to as VDOM, is a programming concept where a virtual representation of a DOM object is kept in memory before it syncs with a "real" DOM object. The VDOM contains all of the properties found in the real counterpart but lacks the ability to change what's on a screen. VDOMs are used because DOM manipulation is slow, so working with a VDOM speeds up development.

You can take your answer up a notch by discussing a specific work-related example. However, a simple definition can work well in most cases.

14. Do you have any experience with distributed systems technologies (including cloud)?

In the simplest sense, distributed systems include any technologies involving several computers coming together to work as if they are a single system. They rely on shared states and operate concurrently, though a single failure won't bring down the entire system.

Since this question is focused on your experience, you'll need to draw from your career when creating an answer. Generally, if you have relevant expertise, you'll want to outline which distributed systems you're familiar with, giving some extra details about the tasks involving them. If you don't have experience, be honest about that fact, then pivot to discuss related skills that you can transition into that area, upcoming training courses that'll get you up to speed, or a general willingness to learn.

15. What are the key code quality tools you use?

Here are some widely-used code quality tools you might discuss:

- Codacy
- Crucible
- DeepScan
- DeepSource
- Embold
- PVS-Studio
- SonarQube
- Upsource
- Veracode

Most software-related work requires a number of product development tools, and this question is designed to dig into your expertise in that area. As with many of the previous questions, this one is focused on tools you've used previously. As a result, you'll want to discuss those you're highly familiar with, potentially presenting use cases outlining your experience with them.

16. How do you approach project estimations?

A well-designed project estimation will include an outline of the tasks involved, resources required, cost rate, project duration, and any required third-party services. Several strategies are available, including the bottom-up, three-point, parametric, and analogous estimation methods.

With your answer, you'll want to outline the approach or approaches you traditionally use. Discuss projects where you applied the methods and discuss the results to showcase the effectiveness of your strategy.

17. List design patterns you know and/or have used in your work

Design patterns are repeatable solutions to common software design problems. They aren't complete designs but can be transformed into effective code directly, not unlike using a template to address various challenges.

Some design patterns you might include in your answer are:

- Abstract Factory
- Adapter
- Bridge
- Decorator
- Factory Method
- Interpreter
- Mediator
- Null Object
- Private Class
- Proxy
- Singleton
- Visitor

You'll want to discuss your experience with various design patterns, not just list them, whenever possible. Since there are so many, focus on those most relevant to the role you want to land.

18. Explain the Big O notation

Big O notation allows developers to analyze how long an algorithm takes to run or the memory required. It effectively describes the complexity of code, relying on algebraic terminology. Generally, it's used to define the limiting behavior of functions when arguments tend toward a specific value or infinity, serving as an upper bound.

Discussing your experience with Big O notation can give the hiring manager more insights into your capabilities, so feel free to follow up the definition with an applicable example. Here's one in our collection of backend developer interview questions comparing LinkedList and ArrayList in Java using Big O notation.

19. Why did you become a software developer?
With this software developer job interview question, you want to discuss your core inspirations for joining the field, whether you're interviewed by a company hiring Go developers, Swift engineers, or big data analysts. Reflect on why you started this career journey. Were you motivated by your experience with a particular technology? Did a person from your life lead you to walk this path?
Ultimately, you want to ensure that your passion and excitement for the field shine through. Use a compelling narrative to outline your motivations, making your answer engaging and relevant.

20. What responsibilities did you have on the last project?
This requires you to outline your role in whatever project you handled last. Discuss the nature of the project, the tasks you managed, collaborative efforts, obstacles and solutions, and the overall outcome to present a thorough answer.

21. What professional achievement are you most proud of?
With this question, you'll want to choose a relatively recent accomplishment that aligns with the job you want to secure and showcases your skills.
Often, using the STAR method to answer is your best strategy. Introduce the project or event that put you on the path. Discuss the requirements and challenges, followed by the actions you took and the skills you used. Then wrap up with the outcome, quantifying the details when possible.

22. What is your greatest weakness, and what have you done to overcome it?
With this question, you'll want to choose a relatively recent accomplishment that aligns with the job you want to secure and showcases your skills.
Often, using the STAR method to answer is your best strategy. Introduce the project or event that put you on the path. Discuss the requirements and challenges, followed by the actions you took and the skills you used. Then wrap up with the outcome, quantifying the details when possible.

23. Describe a challenging task you've had to work on recently. Was it completed successfully? What did you do to solve the problem?

With this question, you'll want to present a recent struggle, preferably one you were able to overcome. Discuss the situation, outlining why it was challenging. List the steps you took to work through it, highlighting relevant skills along the way. Then, go over your solution and the final outcome.

24. Do you prefer working alone or in a team?

While it may seem like you need to pick one or the other, that isn't necessary. Instead, talk about tasks where you find working independently is the best choice and then present scenarios where you value collaboration. Use examples from your past work to show that you can thrive in both situations.

https://www.interviewbit.com/software-engineering-interview-questions/

Software Engineering Interview Questions for Freshers

1. What are the various categories of software?

Software products are mainly categorized into:

- System software: Softwares like operating systems, compilers, drivers, etc. fall into this category.
- Networking and web development software: Computer networking software offers the necessary functionality for computers to communicate with one another and with data storage facilities.
- Embedded Software: Software used in instrumentation and control applications such as washing machines, satellites, microwaves, TVs, etc.
- Artificial Intelligence Software: Expert systems, decision support systems, pattern recognition software, artificial neural networks, and other types of software are included in this category.
- Scientific software: These support a scientific or engineering user's requirements for performing enterprise-specific tasks. Examples include MATLAB, AUTOCAD, etc

2. What are the characteristics of software?

There are six major characteristics of software:

- Functionality: The things that software is intended to do are called functionality. For example, a calculator's functionality is to perform mathematical operations.
- Efficiency: It is the ability of the software to use the provided resources in the best way possible. Increasing the efficiency of software increases resource utilization and reduces cost.
- Reliability: Reliability is the probability of failure-free operational software in an environment. It is an important characteristic of software.
- Usability: It refers to the user's experience while using the software. Usability determines the satisfaction of the user.
- Maintainability: The ease with which you can repair, improve, and comprehend software code is referred to as maintainability. After the customer receives the product, a phase in the software development cycle called software maintenance begins.
- Portability: It refers to the ease with which the software product can be transferred from one environment to another.

Apart from the above-mentioned characteristics, the software also has the following characteristics:

- Software is engineered, it is not developed or manufactured like hardware. Development is an aspect of the hardware manufacturing process. Manufacturing does not exist in the case of software.
- The software doesn't wear out.
- The software is custom-built.

3. What is a framework?

A framework is a well-known method of developing and deploying software. It is a set of tools that allows developing software by providing information on how to make it on an abstract level, rather than giving exact details. The Software Process Framework is the basis of the entire software development process. The umbrella activities are also included in the software process structure.

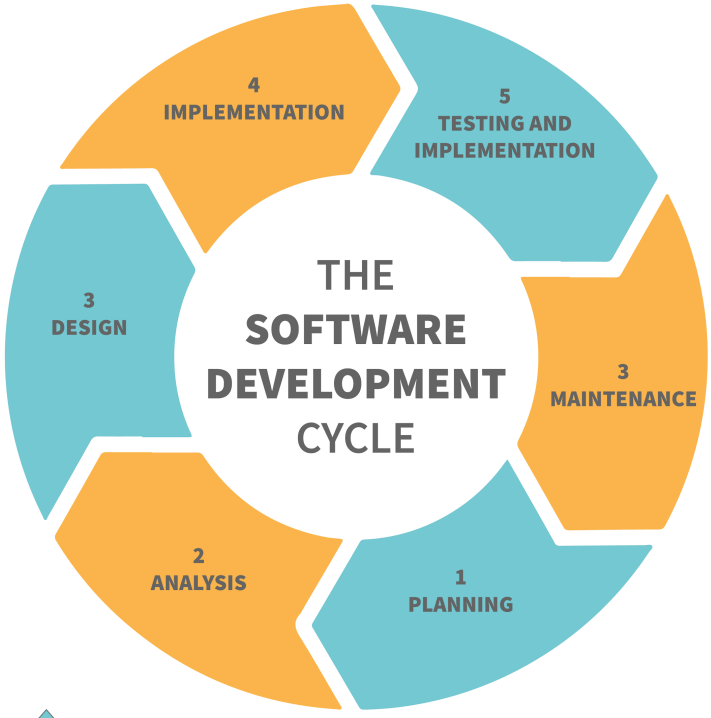
4. What is the main difference between a computer program and computer software?

The key difference between software is a collection of several programs used to complete tasks, whereas a program is a set of instructions expressed in a programming language. A program can be software, but software the vice versa is not true.

5. Describe the Software Development Process in Brief.

The Software Development Life Cycle (SDLC) is a number of fundamental phases that teams must follow in order to produce and deliver high-quality software. Software typically goes through the following phases:

- Requirements Gathering: The team identifies, collects, and defines core issues, requirements, requests, and customer expectations related to the software application or service during this stage of the project. Generating software specifications, creating a thorough strategy, documentation, issue tracking, and project or product planning, including allocating the resources, are some tasks done during this phase.
- Software Design: The team makes software design decisions regarding the architecture and make of the software solution throughout this design phase of development.
- Software Development: Teams develop software solutions based on the design decisions made during earlier stages of the project
- Testing and Integration: Software is packaged and tested to ensure quality. Quality assurance, often known as testing, ensures that the solutions deployed fulfil the specified quality and performance criteria.
- Deployment: The software is installed in a production setting. The gathered, designed, built, and tested work is shared with the software service's customers and users.
- Operation and Maintenance: The software is installed in a production setting. The work is shared with the software service's customers and users.



6. What are different SDLC models available?

- Waterfall model
- Spiral model
- Incremental model
- Agile Model
- Big bang model
- Iterative model

7. Which SDLC model is the best?

According to the annual State of Agile report, Agile is the best SDLC methodology and also one of the most widely used SDLC in the IT industry. The reason is that it is a hybrid of incremental and iterative approaches and works well in a flexible environment. That being said, select the model that suits your requirements.

8. What is Software prototyping and POC?

A software prototype is a working model with limited functionality. The prototype may or may not contain the exact logic used in the final software program, and therefore is an additional work that should be considered in the calculation. Users can review developer proposals and try them out before they are implemented through prototyping. It also helps in comprehending user-specific details that may have been missed by the developer during product development.

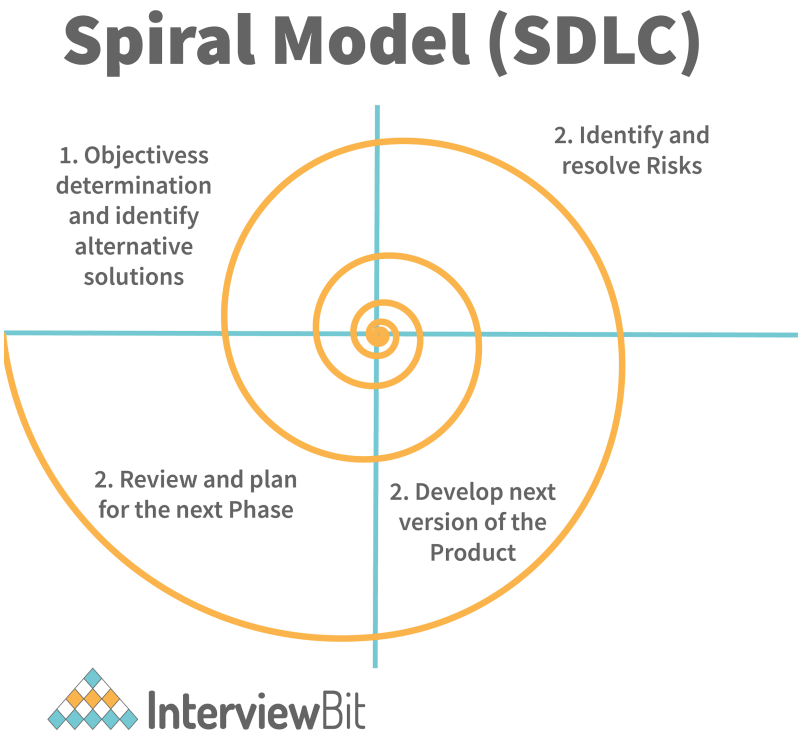
POC (Proof of Concept) is a method used by organizations to validate an idea or concept's practicality. The stage exists prior to the start of the software development process. On the basis of technical capability and business model, a mini project is built to see if a concept can be executed.

14. What are the drawbacks of the spiral model?

The spiral model is a hybrid of the iterative development process and the waterfall model, with a focus on risk analysis. In the SDLC Spiral model, the development process begins with a limited set of requirements and progresses through each development phase. Until the application is ready for production, the software engineering team adds functionality for the increased requirement in ever-increasing spirals.

Drawbacks of the spiral model are:

- It's significantly more complicated than other SDLC models. The procedure is intricate.
- Due to its high cost, it is not recommended for small projects.
- Risk Analysis is overly reliant, and it necessitates a high level of skill.
- Time estimation is challenging
- The spiral could continue endlessly.



9. What is baseline in Software Development?

A baseline is a software development milestone and reference point marked by the completion or delivery of one or more software deliverables. The main objective of the baseline is to decrease and regulate vulnerability, or project weaknesses that can easily damage the project and lead to uncontrollable changes.

10. What is SRS?

SRS is a formal report that serves as a representation of software that allows customers to assess whether it meets their needs. It is a list of requirements for a certain software product, program, or set of apps that execute specific tasks in a specific environment. It also includes user needs for a system, as well as precise system requirements specifications. Depending on who is writing it, it fulfils a variety of purposes.

11. What are CASE tools?

CASE tools are a collection of software application programs that automate SDLC tasks. Analysis tools, Design tools, Project management tools, Database

Management tools, and Documentation tools are a few of the CASE tools available to simplify various stages of the Software Development Life Cycle.

12. What are Verification and Validation?

- Verification: The process of ensuring that software accomplishes its objectives without defects is known as verification. It's the procedure for determining whether the product being developed is correct or not. It determines whether the resulting product meets our specifications. It is mainly focused on functionality.
- Validation: Validation is the process of determining whether a software product meets the required standards, or in other words, whether it meets the product's quality criteria. It is the process of verifying product validation or ensuring that the product we are building is correct. Validation focuses on the quality of the software.

13. What do you mean by Software Re-engineering?

The process of updating software is known as software reengineering. This procedure entails adding new features and functionalities to the software in order to make it better and more efficient.

Software Engineering Interview Questions for Experienced

1. What is the feasibility study?

As the name implies, a feasibility study is a measurement of a software product in terms of how useful product development will be for the business from a practical standpoint. Feasibility studies are conducted for a variety of reasons, including determining whether a software product is appropriate in terms of development, implementation, and project value to the business. The feasibility study concentrates on the following areas:

- Economic feasibility
- Technical feasibility
- Operational feasibility
- Legal feasibility
- Schedule feasibility

2. Define black box testing and white box testing?

- Black box testing is a type of high-level testing in which the primary goal is to evaluate functionalities from a behavioural standpoint. In black-box testing, the tester does not test the code; instead, they utilize the program to see if it works as expected.
- When you have insight into the code or broad information about the architecture of the software in question, you can perform white box testing, also known as clear box testing. It falls under the category of low-level testing and is mostly concerned with integration and unit testing. White box testing requires programming expertise or at the very least a thorough grasp of the code that implements a particular functionality.

3. What is Concurrency?

In software engineering, concurrency refers to a set of techniques and mechanisms that allow the software to do many tasks at the same time. Concurrency can be achieved by using languages like C++ or Java because these languages support the concept of thread. New hardware and software features are required to achieve concurrency.

4. What are Software Metrics?

A software metric is a quantitative measure of program properties. Software metrics can be used for a range of things, such as analyzing software performance, planning, estimating productivity, and so on. Load testing, stress testing, average failure rate, code complexities, lines of code, etc. are some software metrics. The benefits of software metrics are many, some of them being:

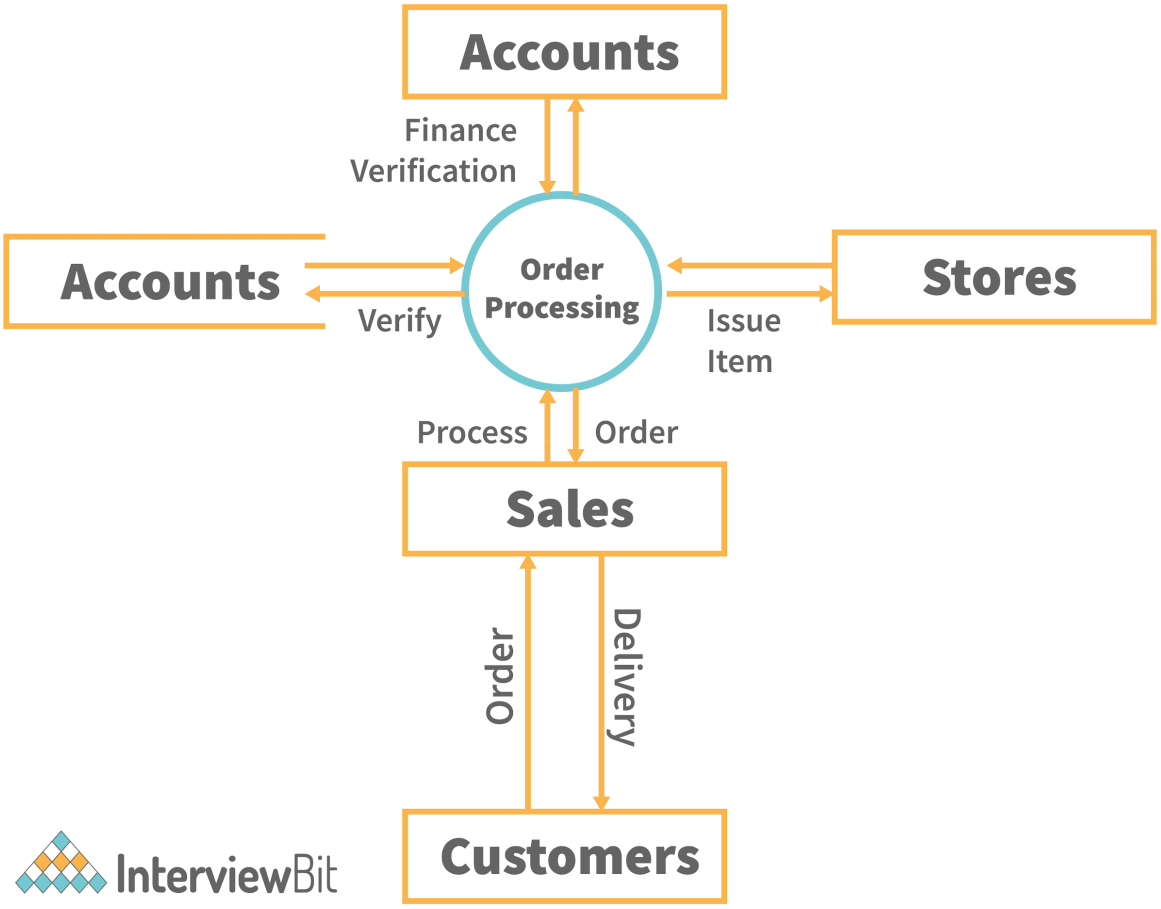
- It reduces cost.
- It increases ROI (return on investment).
- Reduces workload.
- Highlights areas for improvement.

5. What is the difference between cohesion and coupling?

Cohesion	Coupling
Cohesion refers to the relationship within modules.	Coupling refers to the relationship between modules.
Increasing cohesion is good for the software.	Coupling should be avoided.
Modules focus on a particular thing in cohesion.	Modules are coupled to one another through coupling.
Example: A function that checks file permission and then opens it, or a function to decrypt messages.	Example: Two models sharing data with each other.

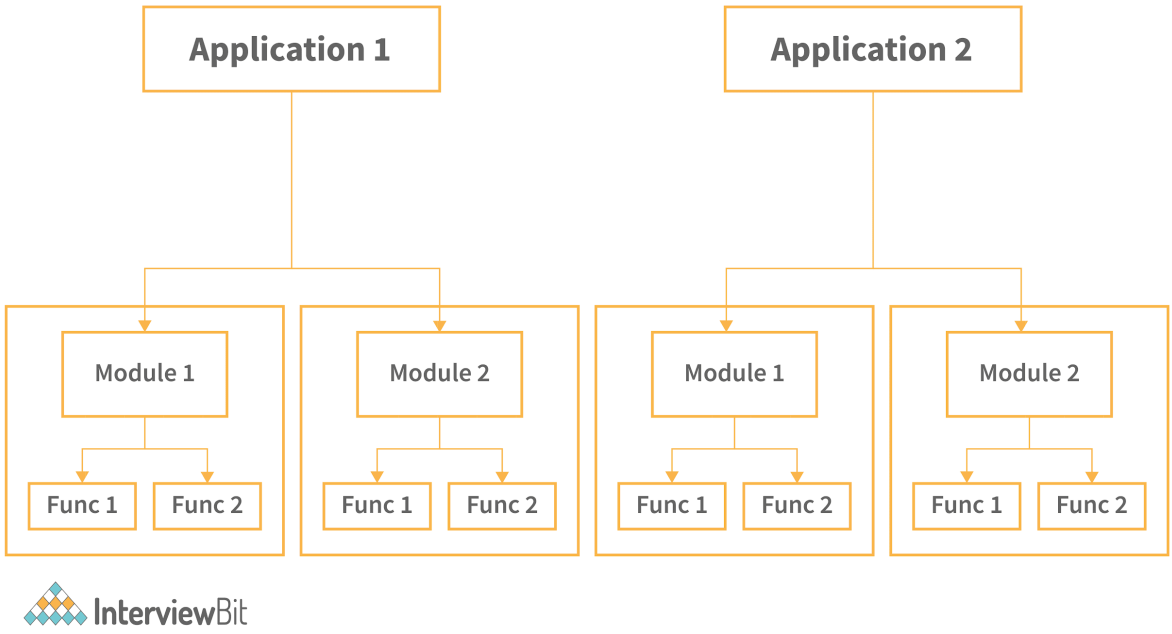
6. What is Data Flow Diagram?

A Data Flow Diagram (DFD) shows the flow of information flows through a system. It shows data inputs, outputs, storage sites, and paths between each destination using symbols such as rectangles, circles, and arrows, as well as short text labels. Data flowcharts can range from simple to in-depth DFDs that go deeper into how data is processed. They can be used to evaluate a current system or to create a new system. A DFD can effortlessly express things that are difficult to describe in words, and it can be used by both technical and non-technical audiences.



7. Explain the concept of modularization.

Modularization is breaking down a program's functionality into separate, independent modules, each of which includes just the information needed to carry out one part of the intended capability. In simple terms, it is the practice of dividing the program into smaller modules so that we can deal with them separately. We can simply add independent and smaller modules to a program using modularization without being hampered by the complexity of the program's other functionalities. Modularization is based on the notion of designing applications that are easier to develop and maintain, self-contained components. In monolithic design, on the other hand, there's always the risk of a simple change knocking the entire application down. The final step would be to combine these independent modules.



In the above diagram, both the applications have been divided into smaller modules. These modules can then be dealt with separately.

8. What is Software Configuration Management?

When a piece of software is created, there is always room for improvement. To modify or improve an existing solution or to establish a new solution for a problem, changes may be required. Changes to the existing system should be examined before being implemented, recorded before being implemented, documented with details of before and after, and controlled in a way that improves quality and reduces error. This is where System Configuration Management is required.

During the Software Development Life Cycle, Software Configuration Management (SCM) is a technique for systematically managing, organizing, and controlling changes in documents, codes, and other entities. The main goal is to enhance production while making as few mistakes as possible.

9. What are functional and nonfunctional requirements?

Functional Requirements	Non-functional Requirements

These are the needs that the end-user specifies as essential features that the system should provide.	These are the quality requirements that the system must meet in order to fulfil the project contract.
The user specifies the functional requirements.	Technical individuals, such as architects, technical leaders, and software engineers, specify non-functional requirements.
Functional Requirements are mandatory. For example, the client might want certain mandatory changes in UI, like dark mode.	Non-functional requirements are not Mandatory. For example, the requirement to enhance readability is non-functional.

10. What is the difference between Quality Assurance and Quality control?

Quality Assurance	Quality Control
Quality Assurance focuses on assuring that the end product (software) will be of the requested quality.	Quality control focuses on controlling the processes, methods, or techniques used in the development of software so that the quality requested is fulfilled.
It is a preventive measure.	It is a corrective measure.
It applies to the full software development life cycle.	It is applied in the testing phase.

Software Engineering Interview Questions (tutorialspoint.com)

Q.What is computer software?
A. Computer software is a complete package, which includes software program, its documentation and user guide on how to use the software.

Q.Can you differentiate computer software and computer program?
A. A computer program is piece of programming code which performs a well defined task where as software includes programming code, its documentation and user guide.

Q.What is software engineering?
A. Software engineering is an engineering branch associated with software system development.

Q.When you know programming, what is the need to learn software engineering concepts?
A. A person who knows how to build a wall may not be good at building an entire house. Likewise, a person who can write programs may not have knowledge of other concepts of Software Engineering. The software engineering concepts guide programmers on how to assess requirements of end user, design the algorithms before actual coding starts, create programs by coding, testing the code and its documentation.

Q.What is software process or Software Development Life Cycle (SDLC)?
A.Software Development Life Cycle, or software process is the systematic development of software by following every stage in the development process namely, Requirement Gathering, System Analysis, Design, Coding, Testing, Maintenance and Documentation in that order.

Q.What are SDLC models available?

A. There are several SDLC models available such as Waterfall Model, Iterative Model, Spiral model, V-model and Big-bang Model etc.

Q.What are various phases of SDLC?
A. The generic phases of SDLC are: Requirement Gathering, System Analysis and Design, Coding, Testing and implementation. The phases depend upon the model we choose to develop software.

Q.Which SDLC model is the best?
A. SDLC Models are adopted as per requirements of the development process. It may vary software-to-software to ensure which model is suitable.
We can select the best SDLC model if following answers are satisfied -

- Is SDLC suitable for selected technology to implement the software ?
- Is SDLC appropriate for a client's requirements and priorities ?
- Is the SDLC model suitable for the size and complexity of the software ?
- Is the SDLC model suitable for the type of projects and engineering we do ?
- Is the SDLC appropriate for the geographically co-located or dispersed developers ?

Q.What is software project management?
A. Software project management is process of managing all activities like time, cost and quality management involved in software development.

Q.Who is software project manager?
A. A software project manager is a person who undertakes the responsibility of carrying out the software project.

Q.What does software project manager do?
A. Software project manager is engaged with software management activities. He is responsible for project planning, monitoring the progress, communication among stakeholders, managing risks and resources, smooth execution of development and delivering the project within time, cost and quality contraints.

Q.What is software scope?

A. Software scope is a well-defined boundary, which encompasses all the activities that are done to develop and deliver the software product.

The software scope clearly defines all functionalities and artifacts to be delivered as a part of the software. The scope identifies what the product will do and what it will not do, what the end product will contain and what it will not contain.

Q.What is project estimation?

A. It is a process to estimate various aspects of software product in order to calculate the cost of development in terms of efforts, time and resources. This estimation can be derived from past experience, by consulting experts or by using pre-defined formulas.

Q.How can we derive the size of software product?

A. Size of software product can be calculated using either of two methods -

- Counting the lines of delivered code
- Counting delivered function points

Q.What are function points?

A. Function points are the various features provided by the software product. It is considered as a unit of measurement for software size.

Q.What are software project estimation techniques available?

A. There are many estimation techniques available.The most widely used are -

- Decomposition technique (Counting Lines of Code and Function Points)
- Empirical technique (Putnam and COCOMO).

Q.What is baseline?

A. Baseline is a measurement that defines completeness of a phase. After all activities associated with a particular phase are accomplished, the phase is complete and acts as a baseline for next phase.

Q.What is Software configuration management?

A. Software Configuration management is a process of tracking and controlling the changes in software in terms of the requirements, design, functions and development of the product.

Q.What is change control?
A. Change control is function of configuration management, which ensures that all changes made to software system are consistent and made as per organizational rules and regulations.

Q.How can you measure project execution?
A. We can measure project execution by means of Activity Monitoring, Status Reports and Milestone Checklists.

Q.Mention some project management tools.
A. There are various project management tools used as per the requirements of software project and organization policies. They include Gantt Chart, PERT Chart, Resource Histogram, Critical Path Analysis, Status Reports, Milestone Checklists etc.

Q.What are software requirements?
A. Software requirements are functional description of proposed software system. Requirements are assumed to be the description of target system, its functionalities and features. Requirements convey the expectations of users from the system.

Q.What is the feasibility study?
A. It is a measure to assess how practical and beneficial the software project development will be for an organization. The software analyzer conducts a thorough study to understand economic, technical and operational feasibility of the project.

Economic - Resource transportation, cost for training, cost of additional utilities and tools and overall estimation of costs and benefits of the project.

Technical - Is it possible to develop this system ? Assessing suitability of machine(s) and operating system(s) on which software will execute, existing developers' knowledge and skills, training, utilities or tools for project. Operational - Can the organization adjust smoothly to the changes done as per the demand of project ? Is the problem worth solving ?

Q.How can you gather requirements?
A. Requirements can be gathered from users via interviews, surveys, task analysis, brainstorming, domain analysis, prototyping, studying existing usable version of software, and by observation.

Q.What is SRS?
A. SRS or Software Requirement Specification is a document produced at the time of requirement gathering process. It can be also seen as a process of refining requirements and documenting them.

Q.What are functional requirements?
A. Functional requirements are functional features and specifications expected by users from the proposed software product.

Q.What are non-functional requirements?
A. Non-functional requirements are implicit and are related to security, performance, look and feel of user interface, interoperability, cost etc.

Q.What is software measure?
A. Software Measures can be understood as a process of quantifying and symbolizing various attributes and aspects of software.

Q.What is software metric?
A. Software Metrics provide measures for various aspects of software process and software product. They are divided into –

Requirement metrics : Length requirements, completeness

Product metrics :Lines of Code, Object oriented metrics, design and test metrics
Process metrics: Evaluate and track budget, schedule, human resource.

Q.What is modularization?
A. Modularization is a technique to divide a software system into multiple discreet modules, which are expected to carry out task(s) independently.

Q.What is concurrency and how is it achieved in software?
A. Concurrency is the tendency of events or actions to happen simultaneously. In software, when two or more processes execute simultaneously, they are called concurrent processes.
Example
While you initiate the print command and printing starts, you can open a new application.
Concurrency is implemented by splitting the software into multiple independent units of execution namely processes and threads, and executing them in parallel.

Q.What is cohesion?
A. Cohesion is a measure that defines the degree of intra-dependability among the elements of the module.

Q.What is coupling?
A. Coupling is a measure that defines the level of inter-dependability among modules of a program.

Q.Mentions some software analysis & design tools?
A. These can be: DFDs (Data Flow Diagrams), Structured Charts, Structured English, Data Dictionary, HIPO (Hierarchical Input Process Output) diagrams, ER (Entity Relationship) Diagrams and Decision tables.
Q.What is level-0 DFD?

A. Highest abstraction level DFD is known as Level 0 DFD also called a context level DFD, which depicts the entire information system as one diagram concealing all the underlying details.

Q.What is the difference between structured English and Pseudo Code?
A. Structured English is native English language used to write the structure of a program module by using programming language keywords, whereas, Pseudo Code is closer to programming language and uses native English language words or sentences to write parts of code.

Q.What is a data dictionary?
A. Data dictionary is referred to as meta-data. Meaning, it is a repository of data about data. Data dictionary is used to organize the names and their references used in systems such as objects and files along with their naming conventions.

Q.What is structured design?
A. Structured design is a conceptualization of a problem into several well-organized elements of solution. It is concerned with the solution design and based on the ‘divide and conquer’ strategy.

Q.What is the difference between function oriented and object oriented design?
A. Function-oriented design is comprised of many smaller sub-systems known as functions. Each function is capable of performing significant task in the system. Object oriented design works around the real world objects (entities), their classes (categories) and methods operating on objects (functions).

Q.Briefly define top-down and bottom-up design model.
A. Top-down model starts with generalized view of system and decomposes it to more specific ones, whereas bottom-up model starts with most specific and basic components first and keeps composing the components to get higher level of abstraction.

Q.What is the basis of Halstead’s complexity measure?

A. Halstead's complexity measure depends up on the actual implementation of the program and it considers tokens used in the program as basis of measure.

Q.Mention the formula to calculate Cyclomatic complexity of a program?
A. Cyclomatic complexity uses graph theory's formula: $V(G) = e – n + 2$

Q.What is functional programming?
A. Functional programming is style of programming language, which uses the concepts of mathematical function. It provides means of computation as mathematical functions, which produces results irrespective of program state.

Q.Differentiate validation and verification?
A. Validation checks if the product is made as per user requirements whereas verification checks if proper steps are followed to develop the product. Validation confirms the right product and verification confirms if the product is built in a right way.

Q.What is black-box and white-box testing?
A. Black-box testing checks if the desired outputs are produced when valid input values are given. It does not verify the actual implementation of the program. White-box testing not only checks for desired and valid output when valid input is provided but also it checks if the code is implemented correctly.

Criteria	Black Box Testing	White Box Testing
Knowledge of software program, design and structure essential	No	Yes
Knowledge of Software Implementation essential	No	Yes
Who conducts this test on software	Software Testing Employee	Software Developer

baseline reference for tester	Requirements specifications	Design and structure details

Q.Quality assurance vs. Quality Control?
A. Quality Assurance monitors to check if proper process is followed while software developing the software.
Quality Control deals with maintaining the quality of software product.

Q.What are various types of software maintenance?
A. Maintenance types are: corrective, adaptive, perfective and preventive.

Corrective
Removing errors spotted by users
Adaptive
tackling the changes in the hardware and software environment where the software works
Perfective maintenance
implementing changes in existing or new requirements of user
Preventive maintenance
taking appropriate measures to avoid future problems

Q.What is software re-engineering?
A. Software re-engineering is process to upgrade the technology on which the software is built without changing the functionality of the software. This is done in order to keep the software tuned with the latest technology.

Q.What are CASE tools?
A. CASE stands for Computer Aided Software Engineering. CASE tools are set of automated software application programs, which are used to support, accelerate and smoothen the SDLC activities.