* **Software Testing**
* **All module included**

1. **What Is Sdlc**: Software Development Life Cycle. in Sdlc they have a 6 types of Resources to full fill the client requirment ( 1.Requirement Gathering and Analysis 2. Design 3. Development 4. Testing 5. Deployment 6. Maintance ) > 1 Requi gathering And Analysis.. we Understanding What the Client Needs and his Requirement and analysis about client the use our infomation. In IT its Possible Or Not we studying it. and we Need Team And Resource to Work on it. and We Decide Deadline And Costing About Project > 2 Design.. we need best Architecture Of project. And System Architecture design the flow to create website They Create user Friendly website to access easly.and Framework Database use the readymade Architecture and using the 1 database like mysql,postgres,mongodb > 3 Development.. developer use the coding and two types of developer 1.Frontend and 2.Backend > 4 Testing.. Tester are check to software working or not then second thing they finding and fixing bugs to ensure the software working proprely > 5 Deployment.. Hosting the software system to working in google and other website poprely > 6 Maintance.. In project Client decide about the maintance of work and maintance on working days and exits time decide and they wont more work they giving more maintance
2. **What Is Software Testing** : Software Testing have 4 type of tesing 1.Unit Testing 2 2.Integration Testing 3.Sytem Testing 4.Uat ( User Accptance Testing)
3. **What is a SQl** : Structure query language, and sql use the only to create table and rows. no Sql is second method of non using method. No Sql Are using Json method in database 4. What is RDBMS..? Ans: Relational Database managment System. Rdbms Are using the Mysql,Oracl,Postgrey,sqlite that types of database 5.Write SQL Commands..? Ans: here a 5 Types Of Commands Using In SQL 1.Create a Table 2.Insert Data 3.Select Data 4.Update Data 5.Delete Data.... \* Create Database = Data name \* Drop Database = Data name 3 \* Create Table = Employe Details ( Id,name,Add ) \* Primary Key \* \* Insert Into Employe details ( name,add,Desig ) Values ("name","add","desig") \* Select \*From = Data \* Delete From data Where = data \* Update Data Set Data Where Data
4. **Integration testing:** This testing phase combines and tests individual software modules or components as a group to identify defects in their interactions and ensure they work together seamlessly. It focuses on validating the data flow and communication between integrated parts.
5. **What determines the level of risk?** The level of risk is primarily determined by the probability of a negative event occurring and the potential impact or consequences if that event does happen. Other factors include the nature of the project, available resources, and external influences.
6. **Alpha testing:** This is an early stage of software testing performed by internal teams (developers, QA) in a controlled, simulated environment. Its goal is to identify major bugs, issues, and ensure basic functionality before releasing the product to external users.
7. **Beta testing:** This is the final stage of testing before general release, where the software is given to a select group of real end-users outside the development team. The aim is to gather feedback on usability, performance, and uncover any remaining bugs in a real-world environment.
8. **Functional System Testing:** This testing verifies if the entire software system functions according to its specified requirements and business rules. It's a black-box testing type, focusing on "what" the system does rather than "how."
9. **Non-Functional Testing:** This testing evaluates the "how" of a system's operation, assessing aspects like performance, usability, reliability, and security. It ensures the system meets quality attributes beyond basic functionality.
10. **GUI Testing:** This involves testing the Graphical User Interface (GUI) of a software application to ensure all visual elements and interactions work as expected. It checks layout, navigation, and user-friendliness.
11. **Adhoc Testing:** This is an informal and unstructured testing approach performed without formal documentation or test cases, relying on the tester's intuition and experience to find defects. It's often used for quick checks or to explore new areas of the software.
12. **Load Testing:** This type of performance testing assesses how a system behaves under a specific, expected load or number of concurrent users. It measures response times, throughput, and resource utilization.
13. **Stress Testing:** This testing evaluates a system's stability and robustness by pushing it beyond its normal operating capacity, often with extreme loads or limited resources. It aims to find the breaking point and how the system recovers.
14. **White Box Testing:** This testing method examines the internal structure, design, and code of a software application. Types include Unit Testing, Static Analysis, Dynamic Analysis, Statement Coverage, Branch Testing, Path Testing, and Loop Testing.
15. **Black Box Testing:** This testing method evaluates software functionality without knowledge of its internal code structure. Techniques include Equivalence Partitioning, Boundary Value Analysis, Decision Table Testing, State Transition Testing, and Error Guessing.
16. **7 Key Principles of Software Testing:**
17. **Testing shows the presence of defects:** Testing can only reveal existing defects, not prove their absence.
18. **Exhaustive testing is impossible:** It's impractical to test all possible inputs and paths.
19. **Early testing:** Testing should start as early as possible in the SDLC to find defects cheaply.
20. **Defect clustering:** A small number of modules often contain most of the defects.
21. **Pesticide paradox:** Repeating the same tests will eventually cease to find new bugs.
22. **Testing is context-dependent:** Testing approaches vary based on the software's type and risk
23. **Absence-of-errors fallacy:** Finding and fixing many defects doesn't guarantee a successful product if it doesn't meet user needs.
24. **Difference between QA vs QC vs Tester:**
25. **QA (Quality Assurance):** A proactive process focused on preventing defects throughout the software development lifecycle by establishing processes and standards.
26. **QC (Quality Control):** A reactive process focused on identifying defects in the finished product through testing and inspection.
27. **Tester:** An individual who performs the actual testing activities to find bugs and verify functionality.
28. **Difference between Smoke and Sanity:**
29. **Smoke Testing:** A quick, high-level test of core functionalities after a new build to ensure the build is stable enough for further testing.
30. **Sanity Testing:** A narrow, deeper test performed after minor code changes or bug fixes to ensure the fix works and hasn't introduced new issues.
31. **Difference between Verification and Validation:**
32. **Verification:** "Are we building the product right?" It ensures the software conforms to specifications and standards.
33. **Validation:** "Are we building the right product?" It ensures the software meets user needs and requirements.
34. **Difference between Priority and Severity:**
35. **Priority:** Defines the urgency with which a defect needs to be fixed, based on business impact or project timelines.
36. **Severity:** Defines the impact of a defect on the software's functionality or user experience, typically classified as critical, major, or minor.
37. **What is Bug Life Cycle:** The Bug Life Cycle (or Defect Life Cycle) describes the various states a bug goes through from its discovery by a tester to its final resolution and closure. It typically involves states like New, Assigned, Open, Fixed, Retest, Reopen, Closed, Deferred, and Rejected.
38. **What is software testing?** Software testing is the process of evaluating a software application to find defects, ensure it meets specified requirements, and verify its quality and performance. It aims to identify errors, gaps, or missing requirements against the actual results.
39. **What is agile methodology?** Agile methodology is an iterative and incremental approach to software development that emphasizes collaboration, flexibility, customer feedback, and rapid delivery of working software. It focuses on adapting to change rather than strict adherence to a plan.
40. **What is SRS?** SRS (Software Requirements Specification) is a comprehensive document detailing all functional and non-functional requirements of a software system. It serves as a blueprint for development, ensuring all stakeholders have a clear understanding of what needs to be built.
41. **What is OOPs?** OOPs (Object-Oriented Programming) is a programming paradigm based on the concept of "objects," which can contain data and code to manipulate that data. It aims to organize software design around data, rather than functions and logic.
42. **Basic Concepts of OOPs:** The fundamental principles of OOPs are Encapsulation, Inheritance, Polymorphism, and Abstraction. These concepts allow for modularity, reusability, and easier management of complex software systems.
43. **What is object?** An object is a real-world entity and a basic run-time entity in an object-oriented system, representing an instance of a class. It has a state (attributes) and behavior (methods).
44. **What is class?** A class is a blueprint or a template from which objects are created, defining their common properties (attributes) and behaviors (methods). It acts as a logical construct for creating instances.
45. **What is encapsulation?** Encapsulation is the bundling of data (attributes) and the methods that operate on the data within a single unit or class, restricting direct access to some of the object's components. It promotes data hiding and security.
46. **What is inheritance?** Inheritance is a mechanism where one class (subclass/child) acquires properties and behaviors (attributes and methods) from another class (superclass/parent). It promotes code reusability and establishes an "is-a" relationship.
47. **What is polymorphism?** Polymorphism means "many forms," allowing objects of different classes to be treated as objects of a common type, enabling a single interface for different data types. It allows methods to behave differently based on the object calling them.
48. **What is JOIN?** A JOIN clause is used to combine rows from two or more tables based on a related column between them. It allows you to retrieve data from multiple tables in a single result set.
49. **Write types of JOINs.** The main types of JOINs are: INNER JOIN (returns rows when there is a match in both tables), LEFT JOIN (returns all rows from the left table, and the matched rows from the right table), RIGHT JOIN (returns all rows from the right table, and the matched rows from the left table), and FULL OUTER JOIN (returns all rows when there is a match in one of the tables).
50. **How Many constraints and describes it self** There are typically five main types of constraints in SQL:
    1. **NOT NULL:** Ensures that a column cannot have a NULL value.
    2. **UNIQUE:** Ensures that all values in a column are different.
    3. **PRIMARY KEY:** Uniquely identifies each record in a table (NOT NULL and UNIQUE combined).
    4. **FOREIGN KEY:** Links two tables together by referencing the primary key of another table.
    5. **CHECK:** Ensures that all values in a column satisfy a specific condition.
51. **Difference between RDBMS vs DBMS** **DBMS (Database Management System)** is a software system that allows users to define, create, maintain, and control access to a database. **RDBMS (Relational Database Management System)** is a type of DBMS that stores data in tables with rows and columns, and uses relationships between these tables. RDBMS adheres to E.F. Codd's 12 rules for relational databases, offering better data integrity and relationships compared to traditional DBMS.
52. **Full Forms**
53. **.ipa:** iOS App Store Package
54. **.apk:** Android Package Kit