**What is Manual Testing?**

* Before getting build from developers test plan and test cases are developed
* Test Cases – Step by step actions that needs to be performed on application
  + Understand requirements and flow of application
  + Actions and corresponding results
  + If there are any issues, report to developers
* White Box Testing - Unit Testing and Integration testing is done by developers and Black Box Testing - System testing is done by tester to determine, if its working according to customer requirements. Tester doesn’t have knowledge of coding
* After System Testing is completed - UAT testing – Black Box Testing – Once testing team has completed testing, they will hand over product to customer. Customers executes their own test cases. User Acceptance Testing – Once it’s successful, it will go live
* **SDLC** – Software Development Lifecycle: it started with V-Model, but companies are now using Agile
  + Requirement Phase:
    - BRS – Business/Customer Requirement Specification – Provided by customer and contain customer requirements. Everyone in team can understand it.
  + Design
    - HLD & LLD – Design Document
  + Coding/Verification
    - Review documents and start coding
    - Are we doing this correctly or not – Insure completeness and accurateness
    - Review, walk-through, inspection
      * Whether it is complete and accurate- Test Plan/Case Review, Code Review, Design Review
      * Formal Review- Everyone sits together and goes through document and if anyone has any question, it’s brought up.
  + After product is developed: Validation: Checking according to requirements its working. More focus on software
    - Unit Testing, System Testing, Integration Testing, UAT Testing
  + Static – Document Testing. Verifying document part for checking accuracy. Dynamic Testing – Once software is ready we will test it dynamically (Unit Testing, System Integration, etc.)
* **QA/QC/QE**
  + Quality Assurance (Product Manage, Project Managers) – QA Testing. QC – Quality Control Team (Testers). QE – Quality Engineer
    - QA is process – Process oriented. Start to end. Build Quality. Develop/Define process. Prevention of defects, What test to perform and when to perform, etc.
    - QC – Validating software by providing different outputs. Implement process. Detection activity
    - QE - Automation – They’ll write automation code to follow test script and writing code to test it
* **System Testing (Black Box Testing)**
  + Understand Requirements (Depends on Quality of Test Cases and focus is to find more and more defects). Verify behavior of application according to customer requirements
    - Find defects and report to developer
    - Participate review meetings
    - Writing automation scripts and execute those scripts and generate reports, status reports and send to management team (Chose test to automate)
* Functional (Links are working, etc.) and Non-Functional Testing (Performance, Security, etc.) – What to test!
* **Test Design Technique (Reduce test data, which will cover maximum coverage)**
  + Boundary Value Analysis/Equivalence Class Partitioning
    - Based on testing at boundary. Talking about range of data. What is minimum and max. Example: 4 – 8 characters are allowed. Testing data: 4, 8, 3, 7, 5, 9
    - Equivalence – Similar to boundary but it focuses on actual value of data. Division of what is valid and what is invalid. Insure coverage of all possible scenarios. Invalid = 3 & 9. Valid = 4-8 characters
  + Decision Table based – Cause-Effect table. Decision-Action. Gives you clear picture of what you’ve tested and what you haven’t.
  + State Transition – State changes based on condition. Example: After 3 attempts user access gets locked. First 3, it will take to homepage. 4th attempt- Contact administrator. Main focus is on state and event.
  + Error Guessing – Based upon previous experience. Mostly done by Unit Testing.
* **Testing Terminologies:**
  + Smoke Testing Vs Sanity Testing: Smoke is first entry point and sanity comes after. Smoke testing is done at early stages. First few builds will not be stable and in those cases, we need to identify critical functionalities. During this testing we install build in our environment and verify basic stuff is working. Sanity is done at later stages, if bug fixes are done and all new functions are working or not. Both are basic functionality test.
  + Regression Vs Retesting: Repeated testing of already tested program after modification to discover any new defects. Retesting: Only checking that particular change. If you test all other stuff that comes under it, then its regression testing. If any new feature is added in addition to fixing other functionality then its regression testing.
  + Globalization vs Localization Testing: Global/Internationalization testing: - Localization supported in local language or not. Date field will be different from 1 country to another. If the application is opened in any other country: different format is allowed (Globalization) – Local along with specific to their culture.
  + End-End Testing: Testing overall functionalities of system. 1 Test case that will cover entire workflow of your application. Each and every functionality of test case.
  + Random/Informal Testing done – If you’ve more time:
    - Exploratory Testing: Testing application without any test case or documents. Discovery, investigation and learning. Getting familiar with application
    - Ad-Hoc Testing: Random testing. No plan, flow, structure or anything. Informal testing. We know about application, but do not have documentation. Main intention is to find bugs/break system. It is unplanned-done when you’ve time.
    - Monkey Testing: Similar to exploratory testing. Intention to break or find bugs
* **STLC: Software Testing Lifecycle:** 
  + Requirement Analysis: Common for QA and Developers. Understand requirements.
  + Test Planning: What, How and When to test: Tools needed and what type of testing we do an application and when to test. Areas to be tested and timeline. Project Plan and functional requirement specifications. Identify resources – Who will be working on testing in the team and software and hardware needed to conduct testing and what skills are required for testing. How many people and how much work we have and budget – Normally, task done by management team. Test plan document: All information about area to be tested, team members involved, etc.
  + Test Design: Project Plan, functional requirements, test plan document (Areas to be tested) and design document (Use cases and pictures/diagrams). How application will be, what will be behavior and etc. Start writing test cases. To understand requirement clearly. Based on review of the documents: Preparation of test scenarios: Describes what to test and different scenario needed to be tested based on functional requirement. Step by step action and expected result. Conduct peer review of documents and review with development and project management team. Should be able to add new scenarios and once reviewed it should be ready. Traceability Matrix: mapping of requirement and test cases to verify did we cover all test cases. Sign-off on test cases
  + Test Execution: Functional Requirements, Test Plan, Test Cases, Build from Development Team. Prepare test log document that says how many test cases were run, how many bugs were occurred, how many passed, etc. Identify defects is the main cause of going through this process and anything that comes as defect will be reported to developer.
  + Defect Reporting: Execute test cases, re-run and provide report (Until you reached end of test cycle). Tracking report after the bug is reported, till bug has been closed. Defect ID, Description, Screenshots, etc. Defect reporting – how many we raised, what’s pending, etc.
  + Test Closure/No more bugs: Analyze reports and verify if there are no more bugs. Evaluate entry and exit criteria and finally send test summary report to client.
  + Build is then forwarded to UAT for testing.
* **Scope of testing**: What’s to be tested, what’s not be tested, schedules, environments, etc.
* **Use Case, Test Scenario and Test Case:** Use case describe requirement: functional requirement – gives more information about requirements. For login case: User is able to login and will land on home page. Actor: User, Action/Flow: Login, Outcome: Getting home page – prepared by BA. Test Scenario (What to test) – different area needs to be tested, Test Case (How to test, test steps, expected and actual results) – created by referring to use cases
* **Test Suites**: Group of test cases that belong to same category
* **Requirement Traceability Matrix**: Contain Requirement ID, Test Case ID. It will give you clear picture whether you wrote test cases for all the requirements or you missed something. Mapping between requirement ID and test cases. Confirms 100% test coverage and verifies nothing was missed. It highlights any missing requirements
* **Test Environment**: Common server in testing. Once installation is done, everyone will view application through URL. It is called test environment. Testing server needs to be installed in individual machine. Environment required for software testing is called: Test Bed.
  + Test Execution always happens in QA Environment. Test Execution happens in multiple cycles.
* Test Case should be able to find defects in software:
  + Error: Related to program
  + Bug: Related to functionality of software – Mismatch between actual vs expected. Unique ID, Steps Executed, Date, Reference (Part of requirement), Status of defect, Severity (How much it has impact on the application- Seriousness of impact on application), Priority (Timeline – P1 (needs to be fixed right away)). Low severity is color or size, etc.
  + Failure: Something found by customer
* **Bug Contents**: ID, Description, Steps, Date, Reference, Status, fixed by, Date Closed, Severity and Priority.
* **Severity**: Impact on the application. If there is blocker it is usually always high priority. If you are able to still continue with application, severity is not that high.
  + Critical/High: Major functionality is not working. Example: Login with invalid credentials
  + Medium: Not aligned with requirements
  + Low: Spelling mistake, etc.
* **Priority**: Timeline. How early developer should fix the problem. Depending on the time, you will give priority level.
  + P1: Immediately, P2: Medium, P3: Low
* **Defect Resolution**: Sometimes developer will accept or reject deject
  + Duplicate, Accept, Reject, need more info (Screenshots, etc.), Fixed, As designed, Enhancement (Can be added in later builds), Not reproduceable (They are not able to get same error) = Debugging needs to be done and test in multiple systems
* **Defect Metrix:** 
  + Defect reject ratio: Out of defects reports more than half were actual defects. Invalid defects say poor performance # of defects rejected/ # of defects raised
  + Defect leakage ratio: How many defects we are not able to identify but customer detected it. # of defects missed/total defects in software
* **Test Closure:** All test cases are completed and defects are solved.
* **Test Metrics:** 
  + % of test cases executed
  + % of test cases not executed
  + % of test cases passed
  + % of test cases failed
  + % of test cases blocked
  + Defects identified per requirement
  + Defect removal efficiency: (Fixed defects) / (fixed defects + Missed defects) \* 100