SDLC

What is Software Testing?

- It is the process of executing a program or application with the intent of finding software bugs/defects using functional and automation tools. Or we can also say that it is the process of validating/verifying a software application.
- Testers should have a "test to break" approach instead of "test to pass" so that they can find the bugs.

What is Requirement?

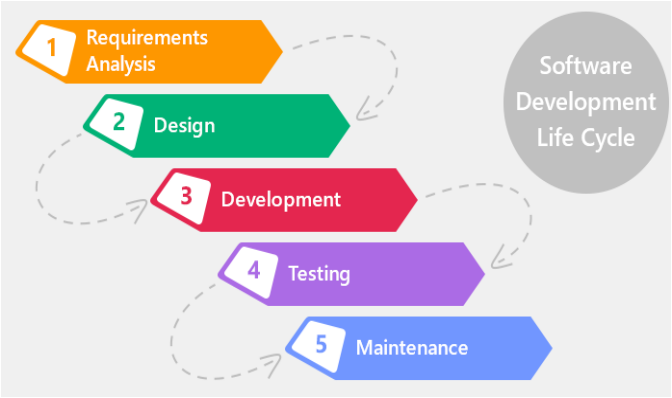
- The software requirements are description of features and functionalities of the target system. Requirements convey the expectations of users from the software product.
- The process to gather the software requirements from the client, analyze and document them is known as requirement engineering.
- The goal of requirement engineering is to develop and maintain a descriptive System Requirements Specification document.

What is the Software Requirements Specification (SRS)?

- A software requirements specification (SRS) is a detailed description of a software system to be developed with its functional and non-functional requirements.
- The SRS is developed based on the agreement between customer and contractors. It may include the use cases of how the user is going to interact with the software system.
- The software requirement specification document is consistent with all necessary requirements required for project development.
- To develop the software system we should have a clear understanding of Software system. To achieve this we need continuous communication with customers to gather all requirements.

What is the Software Development Life Cycle (SDLC)?

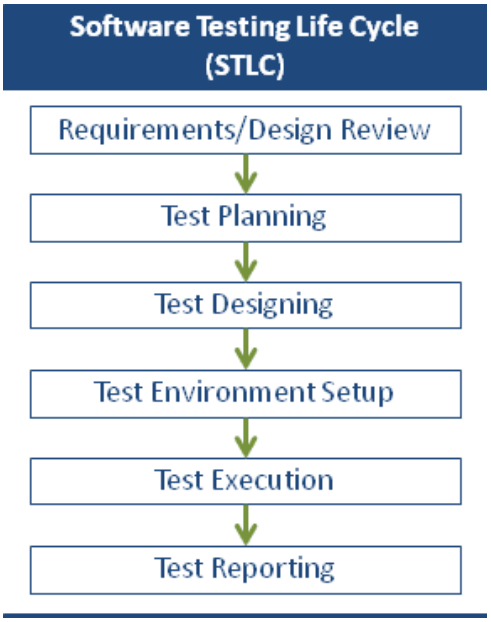
- SDLC defines the **phases** in building of software or application.
- The Phases are:
  - Project Planning
  - Requirement Gathering
  - Design (how the application will be built)
  - Coding or Developing
  - **Testing**
  - Production (deployment or releasing the product)
  - Maintenance



What is the Software Testing Life Cycle (STLC)?

STLC defines the phases in testing of software or application.

1. Requirement / Design Analysis
2. Test Planning
    a. Test Plan
    b. Test Estimation
    c. Test Schedule
3. Test Case Development (Designing)
    a. Test Cases / Test Scripts / Test Data
    b. Requirements Traceability Matrix
4. Test Environment Setup
5. Test Execution
    a. Test Results
    b. Defect Reports
6. Test Closure Activity (Reporting)
    a. Test Results (Final)
    b. Test Metrics
    c. Test Closure Report



What is the difference between STLC and SDLC?

- STLC is part of SDLC. It can be said that STLC is a subset of the SDLC set.
- STLC is limited to the testing phase where quality of software or product ensures. SDLC has a vast and vital role in complete development of a software or product.
- STLC is a very important phase of SDLC and the final product or the software cannot be released without passing through the STLC process.
- STLC is also a part of the post-release/ update cycle, the maintenance phase of SDLC where known defects get fixed or a new functionality is added to the software.

When Should The Testing Start?

- It starts with testing the requirement.
- We have to make sure the requirement is correct in the first place. With the wrong requirement it is impossible to build a bug free application.

What is a tester's main responsibility?

- To find bugs as early as possible. Make sure most of the bug gets fixed.
- To satisfy the end user and client by delivering a bug free and user-friendly application.

Is 100% testing possible?

- We can't test the application 100% since there are unlimited scenarios that we can't even imagine.
- The application is dynamic and developing in time so the new functionalities will require testing.

What is the testing hierarchy?

1. **Unit testing:** Developers test each module or block of code during development.
2. **Component Testing:** Component is a standalone functionality that can work by itself. Ex. Amazon Buyer Functionality, Seller Functionality, Prime Video Functionality.
3. **Integration Testing:** Combine all of the Functionalities. When I integrate them, can I still use all of the functions? Make sure they all still work.
4. **System Testing:** End-to-End testing. Test everything from beginning to end.
5. **Acceptance Testing:** Hire a UAT (User Acceptance Testing) Team or Business Analyst can also do Acceptance Testing. After testing has been completed you have to get another team to do acceptance testing so they can confirm the QA teams testing was successful and have the product ready for the customer.

How Long Did It Take To Build the Regression Suite?

- It took 2 years with; 3 testers 1 manual tester + 2 automation tester
- when we run:
  - before release
  - after major bug fix
  - after major new functionality
- where we keep test scenarios and where we as a team take decisions which will be executed more than once, in one sprint you test some scenarios.

Tell us one challenge while running your regression suite?

- Failures. Because our regression suite was developed so long ago, and you don't know what has changed. The properties of a button may have changed.

How Many Environments Do You Have?

1. Development Environment
   a. Unit testing
   b. Less stable than test environment
2. Test Environment
   a. Manual testing happens here
   b. Replicates the production environment
   c. Changes are deployed in intervals
   d. Automated **smoke tests** are ran here
      i. Runs against the test environment to make sure if the application is stable enough to perform other major testing activities.
      ii. Run every time changes are deployed to Test environment
      iii. Can be ran in dev environment
   e. Automation tests are run here
   f. Automated Integration tests run here
3. Pre-production Environment
   a. UAT environment
   b. Demo happens here
   c. load/performance testing happen here
   d. Changes are deployed in big intervals
   e. Automated major **regression tests** here (before release)

i. Runs against the UAT environment
ii. To find out if new changes result in any defects
iii. Runs after major bug fixes and every release
iv. This Test Is Decided Test Plan

f. Very stable

4. Production environment

Black Box Testing (Behavioral Testing)

- In Blackbox Testing, the tester only knows what the software is supposed to do, he doesn't care what is going on inside the box, how it works and what the logic is. He only cares if the software is doing its job.
- Testing smartphones by using it. Can I make calls? Can I browse on the web? Testing the application like an end-user.
  TECHNIQUES:
  - Equivalence CLASS Partitioning:
    It is a software test design technique that involves dividing input values into valid and invalid partitions and selecting representative values from each partition as test data. *You can not test everything. You can not test the whole grape basket before buying.*
  - Boundary VALUE Analysis:
    It is a software test design technique that involves the determination of boundaries for input values and selecting values that are at the boundaries and just inside/outside of the boundaries as test data. Your credit card spending limit is 5000 and you should at least spend 50 to avoid monthly fees. 50 is Lower Boundary. 5000 is Higher Boundary or higher limit
  - Cause-Effect Graphing:
    It is a software test design technique that involves identifying the cases (input conditions) and effects (output conditions), producing a Cause-Effect Graph, and generating test cases accordingly.

White Box Testing

- Whitebox Testing is the software testing method in which internal structure is being known to the tester who is going to test the software.
- Testing smartphone chips, memory etc.
- Testing the codes.

Gray Box Testing

-

Ad-Hoc Testing (Random, Monkey)

- Is a method of software testing without any planning and documentation.
- This method is normally used during Acceptance Testing.
- Conducted informally and randomly without any formal expected results.
- The success of ad hoc testing depends on the creativity and tenacity of the tester (and, of course, luck)

What is Functional Testing?

- In functional testing basically the testing of the functions of a component or system is done. It refers to activities that verify a specific action or function of the code.

What is Non-functional Testing?

- Performance testing is testing that is performed, to determine how fast some aspect of a system performs under a particular workload. For example,
  - Can 2000 concurrent - user login to the application at the same time?
  - Can a user move to the next page in 1second?

Performance Testing

- Load Testing
- Stress Testing
- Endurance Testing
- Virtual Users
- Transaction Response Time

Unit Testing

- A unit is the smallest testable part of an application like functions, classes, procedures, interfaces.
- Unit tests are basically done by developers to make sure they eliminate the bug before testers find it.
- Normally Testers don't perform Unit Tests, everyone is responsible for their own Unit Test.

Component Testing

- Component testing is a method where testing of each component in an application is done separately.
- Component testing is also known as module and program testing. It finds the defects in the module and verifies the functioning of software.
- Before integrating the components of the software the single component has to be working fine.
- Amazon is a huge website which has separate components like: My Account, Search, Payment, Prime, etc. Testing each of them separately!

Integration Testing

- Integration testing tests after integrating multiple components if the system works fine or not.
- Single component might work fine by itself but when integrated with multiple components it might fail.
- For example, testing Amazon's all components together.

Acceptance Testing

- After the system test has corrected all or most defects, the system will be delivered to the user or customer for acceptance testing.

- Acceptance testing is basically done by the user or customer although other stakeholders may be involved as well.
- Also known as UAT User Acceptance Testing.
- The Acceptance testing will be performed after QA testing. In my current project it is done by the UAT team. After the UAT team performing the acceptance testing the code will go to production.
    - Development environment(where developers write code and perform unit testing)
    - QA environment (where we test the application.)
    - UAT environment (after the code is tested in the QA environment it will be deployed to the UAT environment. The UAT testing team will perform testing to make sure it fits the business requirement. It is also called a staging environment.
    - Production environment(is when the end user can see the real application)

Regression Testing

- If any new functionality is added we have to make sure the new functionality did not break existing functionality, unintentionally.
- If there is any bug fixed we have to make sure the bug fixed did not break other functionality that was working fine.
- If there is any change in the code we have to make sure the developer did not break the other codes unintentionally.

Smoke Test

- Smoke testing is the initial testing process exercised to check whether the software under test is ready/stable for further testing.
- Before putting the entire team to testing effort, the system needs to be stable enough or worth testing.
- Also called Sanity Check. In Fannie Mae it is called Shakeout.

Positive Testing

- Positive testing can be performed by testing the application with valid input.
- If you try to login with a valid username and password it is positive testing.

Negative Testing

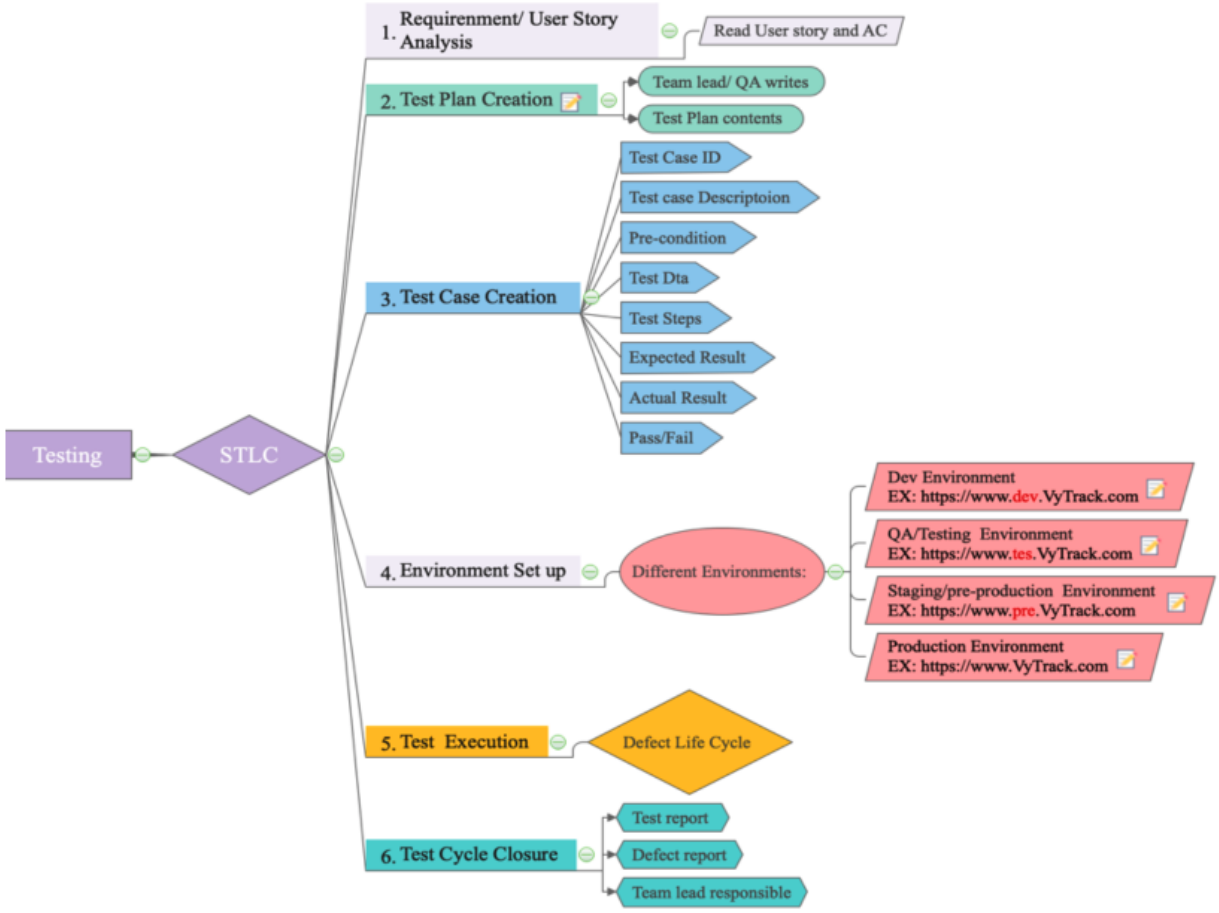
- Negative Testing can be performed on the system by providing invalid data as input.
- It checks whether an application behaves as expected with the negative input.
- This is to test the application that does not do anything that it is not supposed to do so.
- How many children do you have? 3.5 (trying to input this value to the application is negative test)
- Login with invalid username and password.

What is Verification and Validation?

- **Verification** happens during developing by testers and developers; it is a process of evaluating software at development phase and to decide whether the product of a given application satisfies the specified requirements.

- **Validation** by testers; is the process of evaluating software at the end of the development process and to check whether it meets the customer requirements.

Software Testing Life Cycle



What is Software Testing?

- Software Testing is a part of the Software **Development Life Cycle (SDLC)**.
- Software testing is an activity to **detect** and **identify the defects** in the Software.
- The objective of testing is to release a **quality (bug free) product** to the client.

Software Testing Life Cycle

- Software Testing Life Cycle refers to a testing process which has specific steps to be executed in a definite sequence to ensure that the quality goals have been met.
- In the STLC process, each activity is carried out in a planned and systematic way. Each phase has different goals and deliverables.

STLC Phases:

- Requirements Analysis
- Test Plan Creation
- Test Case Creation
- Environment Setup
- Test Execution
- Test Cycle Closure

Step 1: Requirement Analysis:

- Analyze and study the requirements. Have brainstorming sessions with other teams and try to find out whether the requirements are testable or not.
- This phase helps to identify the scope of the testing.
- If any feature is not testable, communicate it during this phase so that the mitigation strategy can be planned.
- PO or team lead assigns you user stories to test , or you may choose user stories to test. *You need to read the user story carefully and make sure to understand it well.*

Step 2: Test Plan Creation:

- Describes the scope, approach, resources, schedule, test items, features to be tested and not tested, tasks, and contingencies for the entire testing process.
- It identifies test **items**, the **feature** to be tested, the testing tasks, who will do each task, and any risks requiring contingency planning.
- It is written by a team lead or tester.
- A detail of **how** the test will proceed, **who** will do the testing, **what** will be tested, in how much **time** the test will take place, and to what quality level the test will be performed.
  Types;
  - ➔ Master Test Plan
  - ➔ Testing Level Specific Test Plans (Unit TP, Integration TP, System TP, Acceptance TP)
  - ➔ Testing Type Specific Test Plans; like Performance TP and Security TP
- Example test plan for one of our VyTrack projects. This test plan is only for one of the modules in Vytrack called Fleet. click on me:)

TEMPLATE:

- Test Plan Identifier
- Introduction
- References
- Test Items
- Features to be Tested
- Features Not to be Tested
- Approach
- Item Pass/Fail Criteria
- Suspension Criteria and Resumption Requirements
- Test Deliverables
- Test Environment
- Estimate
- Schedule
- Staffing and Training Needs
- Responsibilities
- Risks
- Assumptions and dependencies
- Approvals

Step 3: Test Case Creation

- A Test Case is a documentation which is a set of actions executed to verify a particular feature or functionality of your software application.
- Test cases are written by test engineers.
- They are written formats: excel sheet format, **Jira** with **Gherkin** Language.
- Test case describes the functionality and test steps.
  - ➢ Test Case ID
  - ➢ Step number
  - ➢ Description of the functionality
  - ➢ Expected result

- ➢ Actual Result

❖ Best Practices to Write Test Cases

- ➢ Test Cases need to be simple
- ➢ Create a Test Case with the End User in Mind.
- ➢ Avoid test case repetition
- ➢ Ensure 100% Coverage
- ➢ Test Cases must be identifiable

What is a Test Scenario?

❖ Possible area to be tested in a user story.
❖ Answer for"**What to test**?"
❖ Before writing a test case,you have to come up with test scenarios and "Review" with your team.
❖ After you have all scenarios, dev-team and testing team together review test scenarios.
❖ Review could be anytime, so you have to check your email frequently.

Positive and Negative Scenario:

- **Positive scenario** determines that your application works as expected. If an error is encountered during positive testing, the test fails.
- **Negative testing** ensures that your application can gracefully handle invalid input or unexpected user behavior.
- Both positive and negative scenarios should be PASS when you test it.

Step 4: Environment Setup

❖ Developers should push their codes from **dev** environment to **QA** environment

Activities

- Understand the required architecture, environment set-up and prepare hardware and software requirement list for the Test Environment.
- Setup test Environment and test data
- Perform smoke test on the build

Deliverables

- Environment ready with test data set up
- Smoke Test Results.

Step 5: Test Execution

❖ During this phase test team will carry out the testing based on the test plans and the test cases prepared testmanuallyor/and automatically

Activities

- Execute tests as per plan
- Document test results, and log defects for failed cases
- Map defects to test cases in RTM
- Retest the Defect fixes
- Track the defects to closure

Deliverables

- Completed RTM with the execution status
- Test cases updated with results
- Defect reports

Step 6: Test Cycle Closure

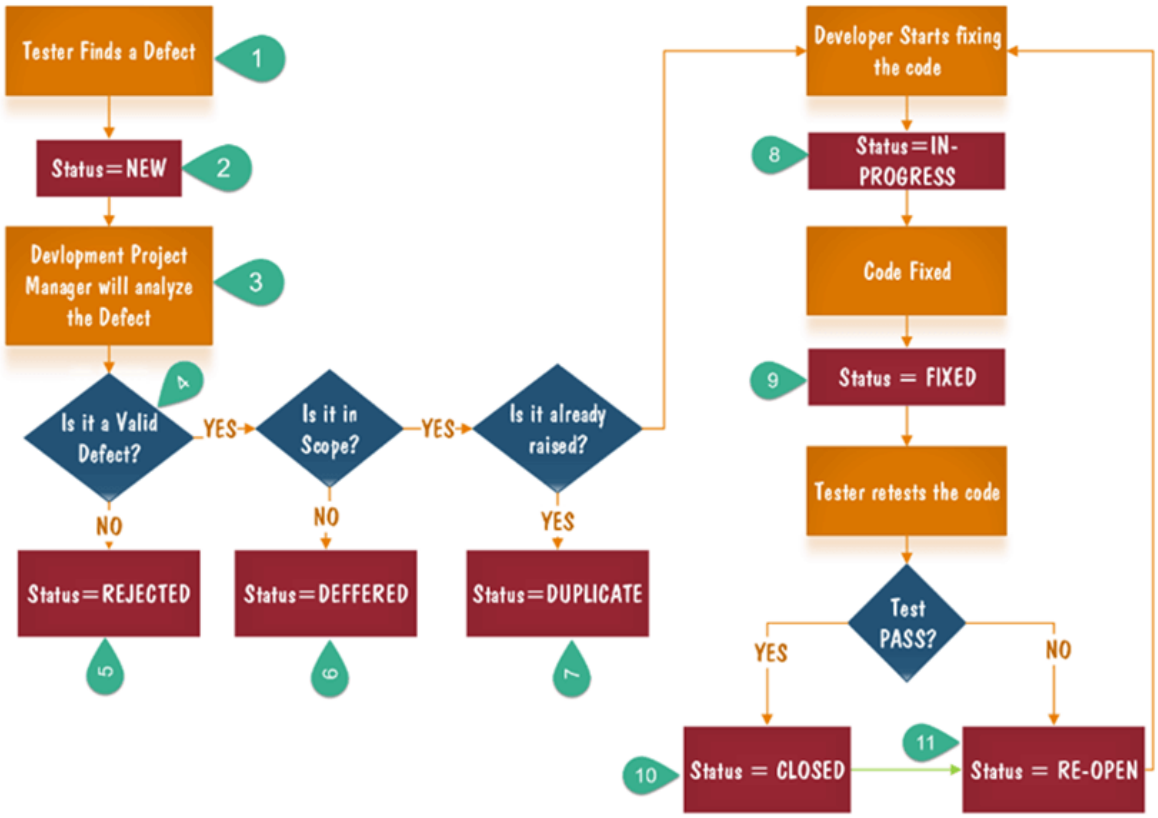
- Test Report on **Jira**
- Defect Report
- All of the bugs are handled
- Test Lead is responsible

What is a defect?

- When the expected result does not match the actual result, it is a defect.

What is Defect Life Cycle (DLC)?

- There is a systematic process for dealing with defects and how it goes from one phase to the other is called “Defect Life Cycle”.
- Defect changes various status from its origin to its closure, and are taken care by various teams.



Defect Life Cycle Status:

- **New**: This is given by QA, all new defect status will be given as New.
- **Assigned**: Once the bug is posted by the tester, assigned to the development team to address it but not yet resolved.
- **Open**: The developer starts analyzing and works on the defect fix.
- **Fixed**: When a developer makes a necessary code change and verifies the change, he or she can make bug status as “Fixed.”
- **Re-test**: Tester does the retesting of the code at this stage to check whether the defect is fixed by the developer or not and changes the status to “Re-test."
- **Reopen**: If the bug persists even after the developer has fixed the bug, the tester changes the status to "reopened". Once again the bug goes through the life cycle.
- **Closed**: If the bug no longer exists then the tester assigns the status "Closed."
- Rejected: If the developer feels the defect is not a valid defect then it changes the defect to “rejected."
- **Duplicate**: If the defect is repeated twice or the defect corresponds to the same concept of the bug.
- **Not a bug**: If it does not affect the functionality of the application then the status assigned to a bug is "Not a bug”.
- **Deferred**: If the present bug is not of a prime priority and if it is expected to get fixed in the next release or in the future.

Bug/Defects Categories:

A: The Severity of the Bug:

Severity is defined as the degree of impact a defect has on the development or operation of a component application being tested.

The QA engineer (tester) usually determines the severity level of the defect.

Defect Severity Level:

- **Critical**: This defect indicates complete shut-down of the process, nothing can proceed further.
- **Major**: It is a highly severe defect and collapses the system. However, certain parts of the system remain functional.
- **Medium**: It causes some undesirable behavior , but the system is still functional.
- **Minor**: It won’t cause any major break-down of the system.

B: The Priority of the Bug:

Priority is defined as the order in which defect should be fixed. ***How fast does it have to be fixed ?*** *Higher the priority, the sooner the defect should be resolved.*

Developers or Product Owner determine the priority level of defects.

Defect Priority Level:

- **High** : The defect must be resolved as soon as possible as it affects the system severely and cannot be used until it is fixed.

- **Medium**: During the normal course of the development activities defect should be resolved.
- **Low**: The defect is an irritant but repair can be done once the more serious defect has been fixed.

What do you do when you find a defect?

- If I find a defect, before reporting it I reproduce the bug that I need to make sure that it is a valid defect.
- If it is a small issue, I will go to the developer desk, and he can fix it right away.
- If it is a big issue, then I open my JIRA and log the defect.
- If I am not sure if it is a bug or not, I will talk to the SME (*subject matter expert it means the person who knows the application better than anyone*).

If the developer says not a defect, what to do?

- I always make sure that it is a real defect that's why I reproduce it.
- I take screenshots and give all the steps to reproduce the defect.
- Actually, one of my biggest challenges that I faced in my current project is that.

What will you do when the script fails?

- First of all, I will identify the failure, if it is this due to application error, sync error, script issue or environment is down. I will rerun the failed tests only to look deep into the report and to find the reason for the failure.
- Let's say, if it is due to synchronization issues, I will add extra time by using implicit, explicit or some custom expected conditions.
- If it is a script issue, I will start debugging (identify) my script and fix it, analyze the exceptions but if it is a real defect then I will log the defect.

What is the difference between debugging and testing?

- The main difference between debugging and testing is that debugging is typically conducted by a developer who also fixes errors during the debugging phase. Testing on the other hand, finds errors rather than fixes them. When a tester finds a bug, they usually report it so that a developer can fix it.

What do you know about "Planning Poker" technique?

- Planning poker, also known as Scrum Poker, is a card based agile technique that is used for planning and estimation. To start a session of planning poker technique, the agile user story is read by the product owner.
- The steps performed in the poker planning technique are –
  - Each estimator has a deck of poker cards with the values such as 0, 1, 2, 3, 5, and so on, to denote story points, ideal days or something else that the team uses for estimation.
  - Each estimator has a discussion with the product owner and then privately selects a card on the basis of their independent estimation.
  - If the cards with the same value are selected by all estimators, it is considered as an estimate. If not, the estimator discusses the high and low value of their estimates.

- Then again, each estimator privately selects a card and reveals. This process of poker planning is repeated to reach a general agreement.

Agile

What is Waterfall Methodology?

- Waterfall project management entails mapping out a project into distinct, sequential phases, with each new phase beginning only when the prior phase has been completed.
- The waterfall system is the most traditional method for managing a project, with team members working in a linear fashion towards a set end goal.
- Each participant has a clearly defined role and none of the phases or goals are expected to change.
- Waterfall project management works best for projects with long, detailed plans that require one phase to be done before another can start.
- These projects require a single timeline and changes are often discouraged and costly.

What is Agile Methodology?

- Agile is a process by which a team can manage a project by **breaking it up into several stages** and involving constant **collaboration** with stakeholders and continuous improvement and iteration at every stage.
- The Agile methodology begins with clients describing how the **end product** will be used and what problem it will solve. This clarifies the customer's expectations from the project team.
- Once the work begins, teams **cycle** through a **process of planning, executing, and evaluating** — which might just change the final deliverable to fit the customer's needs better.
- **Continuous collaboration** is key, both among team members and with project stakeholders, to make fully-informed decisions.

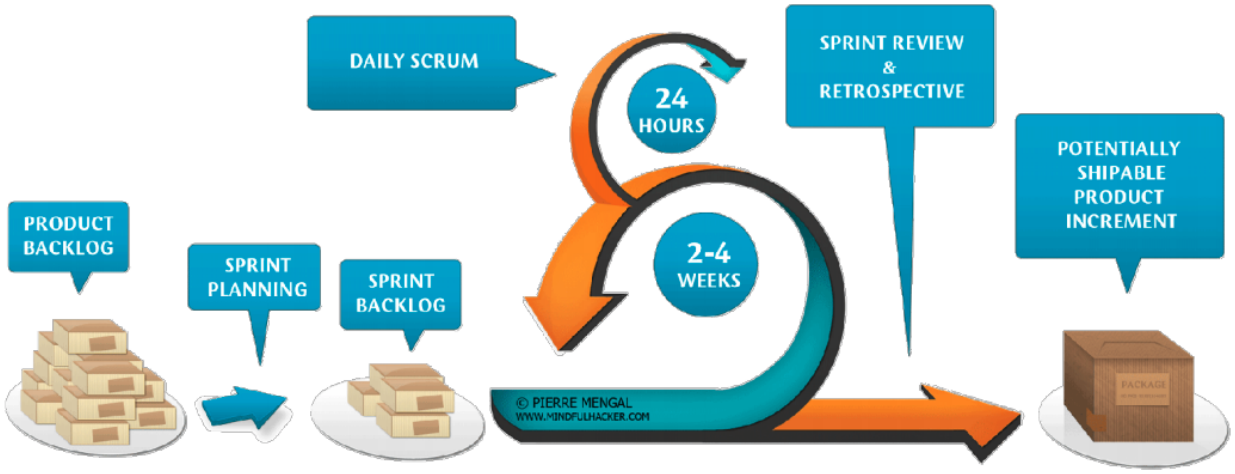
Agile Values & Highlights:

- Individuals and Interactions
- Working Software
- Collaboration
- Responding to change
- Process and tools
- Comprehensive Documentation
- Contract Negotiation
- Following a plan

What is Scrum?

- Scrum is an agile process that allows us to focus on delivering the **highest business** value in the **shortest time**.
- It allows us to **rapidly** and **repeatedly** inspect actual **working software** (every two weeks to one month).
- The business sets the priorities. Teams **self-organize** to determine the best way to deliver the highest priority features.
- Every two weeks to a month anyone can see real working software and decide to release it as is or continue to enhance it for another sprint.

Agile - Scrum Development Visual



Agile Framework (Scrum)

In my company, we follow the Scrum framework. There are other frameworks like Kanban, Extreme Programming (XP), and Adaptive Project Framework (APF) but I never worked with them.

- **Roles**:
  Product Owner, Scrum Master, Team.
- **Ceremonies**:
  Sprint Planning, Sprint Review, Sprint Retro, Daily Scrum, Grooming Session
- **Artifacts**:
  Product Backlog, Sprint Backlog, Burnout Chart

Scrum Roles:

Product Owner:

- Define the features of the product.
- Decide on release date and content.
- Be responsible for the profitability of the product (ROI).
- Prioritize features according to market value.
- Adjust features and priority every iteration, as needed.
- Accept or reject work results.
- Leading the team to the right direction.
- Motivating team with clear direction.
- Consistently communicates with the client and stakeholders to make sure the Development team is in the right business direction.
- Being a bridge between the Development team with Client and Stakeholders.

Scrum Master:

- Responsible for enacting Scrum values and practices
- Removes Impediment
- Ensure that the team is fully functional and productive
- Enable close cooperation across all roles and functions

- Shield the team from external interferences
- Represents management to the project
- Do everything possible to make sure Development Team performs in highest level
- Coordinates the Collaboration among the team members
- Serve to the team, he has no authority over the team

Scrum Development Team:

- This is a group that works together for creating and testing incremental releases of the final product.
- Formed by the collective contribution of individuals who perform for the accomplishment of a particular project.
- The team is bound to work for the timely delivery of the requested product.

Scrum Artifacts:

Product Backlog:

- This refers to what remains on the "to be done" list. Or it is a list of work that has to be done in the project.
- During a product backlog grooming session, the development team works with the business owner to prioritize work that has been backlogged.
- The product backlog may be fine-tuned during a process called backlog refinement.
- Change is welcome on the product backlog.

Sprint Backlog:

- Sprint Backlogs are tasks that a scrum team needs to complete in **2 weeks** sprint or **4 weeks sprint**.
- These are divided into time-based user stories.

Burnout Chart:

- It is a visual representation of the amount of work that still needs to be completed.
- A burn-down chart has a Y axis (work) and an X axis (time).
- Ideally, the chart illustrates a downward trend, as the amount of work still left to do over time burns down to zero.

Product Increment:

- This refers to what's been accomplished during a Sprint -- all the product backlog items -- as well as what's been created during all previous Sprints.
- The product increment reflects how much progress has been made.

Scrum Ceremonies:

Sprint Planning Meeting:

- Discuss and decide what can be realised in the next sprint and how this should be done.
- Team needs to know how much capacity and commit some points (each point represents some hours (individually).
- Jointly establish a Sprint Goal Discuss and decide what can be realised in the next Sprint and how this should be done.

Daily Stand up (Daily Scrum) Meeting:

- Everyday but the first day of the sprint and around 15 mins
- Same Place and time with Full Team
- Run by SCRUM MASTER
- Provide Update on the sprint backlog work progress
- Focus on 3 Main Questions
  - What did you do yesterday?
  - What will you do today?
  - Do you have any blocker (impediment)?

Sprint Demo (Review) Meeting:

- Usually it happens at the end of the Sprint (like after the Every 2 sprint) § Usually everyone will join (+ customer, stakeholders ...)
- All new functionalities of the app will be shown
- Usually run by the developer
- Automation engineer can DEMO their automation script as well

Sprint Retro (Retrospective) Meeting:

- Will be after every DEMO (ideally)
- Scrum Team will be present in the meeting
- Opportunity for the Scrum Team to inspect and improve itself.
- What went well?
- What went wrong?
- How can we improve?
- Start doing more of what?
- Stop doing what?
- Continue doing what?

What is Epic?

- Epic is a big user story that you cannot complete in one sprint or we can say that it is the biggest user story which is almost everything about one project. For Example:

  As a customer I should be able to buy from Amazon, so I don't have to go to local stores.

  Above user story is everything about amazon.com. It includes Login, Logout, Search Product, View Product, Basket, Credit Card, Shipping Method and Return functionalities.

What is Feature?

- Feature is multiple user stories that you can put together and it becomes a stand alone component of one application.

  As an Amazon Prime member I should be able to stream movies online, so I don't have to use Netflix.

  It is not totally away from amazon, it is under amazon project, but it can be stand alone component of amazon.com

What is a User Story?

- Note: basically, a user story is just a requirement.
- User story is a short simple description of a minimum shippable product.

- It normally looks like this:
  As **<end-user>** I want to do **<action>** So that I can **<benefit>**
  As **amazon user** I should able to **login**, so I can **buy** stuffs online

What is an Acceptance Criteria?

- Acceptance criteria is the way that we know the user story is successfully developed or not.
- When BA or PO write user stories, s/he writes AC together.
- Statements of requirements that are described from the point of view of the user to determine when a story is "done" and working as expected

Example User Story:
As a customer I should be able to pay with a credit card so I don't have to pay cash.

Acceptance Criteria:
User should be able to pay with a Visa card.
User should be able to pay with Mastercard.
User should be able to pay with Amex.

User should be able to pay with any above credit card by entering following information:
Card Holder Name, Card Number, Expiration Date and 3 digit security code in the back.

What is Rat Hole and Parking Lot Item?

- **Rat Hole:**
  It is in the meeting when the team gets into too much argument and wasting time we call it Rat hole. As soon as someone says it is Rat hole we should stop arguing.
- **Parking Lot Item:**
  It is a valid problem in the meeting but it is your specific problem, so if you keep people discussing your problem in the meeting because we are wasting other people's time.
  < Let's make it a parking lot item > means whoever is interested in that issue can talk after the meeting.

Tell me about your Agile experience in your recent project:

- There are 9 people in my team: 4 developers (Brian, Yasin, Zeynep, Vasyl); 3 testers (Ozzy, Aaron, and myself); 1 Product Owner: Mike; 1 Scrum Master: Sam
- Before each Sprint, we have our Sprint Planning Meeting:
  - We discuss the priority features and backlog items.
  - We learn that specific part of the application which we are going to work on.
  - We choose our stories based on velocity and capacity:
    - Velocity: Number of story points delivered/demo in a sprint. For example: if team planned 30 story points (Business value); worth of user stories in a sprint and able to deliver as planned then team's velocity is 30
    - Capacity: Total number of available hours for a sprint is Team's capacity. Calculate holiday and PTO hours
  - This meeting is held every week and lasts for almost 1 hour. We get a general idea then we do a Sprint Grooming meeting for giving some estimates for the tasks. ???

- Team, SM, and PO get together to ensure work items are relevant and useful
- Ask questions to P.O. of user stories
- Re-define acceptance criteria
- Writing Stories
- Breaking Epics Into User Stories
- Understand the story to give proper estimation/to prevent under/over estimate

- **How Do You Estimate?**
  - Based on my experience and complexity of the story and it is something I worked on before.
- After sprint starts, we do Daily Stand up Meeting
  - Every morning we discuss what we did yesterday, what we will do today and any blocker.
- End of the sprint, we usually do a Sprint Demo/Review Meeting.
  - It is just to show customer what we build sprint (PO can put feedback)
  - As an SDET in my team, I have done presentations sometimes and go over the functionalities in the conference room.
  - Clients or stakeholders or business people ask questions that they don't know.
- After the Sprint Demo, we do a Sprint Retrospective Meeting .
  - In Sprint Retro, we talk about what was good in the last sprint, what kind of mistakes we made.
  - We go over them and make sure that we don't make the same mistakes again.
  - If we did something good and improvements, we would continue doing it.
  - This meeting is held at the sprint review meeting or at the end of the sprint; it lasts for 2-3 hours.

What is a Test Script?

- Specifies test procedures, typically for automated testing.
- A test script can be as simple as the one below:

```
def sample_test_script (self):
type ("TextA")
click (ImageButtonA)
assertExist (ImageResultA)
```

What is Requirement Traceability Matrix (RTM)

- RTM is used to make sure that all test cases cover the requirement or not. It is like an excel sheet.