Why do you love testing?

It started without an option. As a fresher we are assigned to a Testing team without any preference. We have no option. Yeah! We are 'pushed' into Testing at career start.

Once started, it has been a roller coaster ride. It is different from what it seems – a tester is required to excel in different aspects – test process, techniques, planning, automation, mindset, logical understanding, end-user behaviour, client interaction, status reports, tools, programming, methodology, etc.

Testing too is a technical activity. There are multiple aspects to it – functional, performance, security, debugging, tools, automation, etc.

Testing is an activity that makes the software/product better.

How will you feel if you bought a new smartphone at a high price only to find a fault when used? Frustrated. Disappointed. Cheated. Angry. Sad? Yeah! A tester's job is to prevent you feeling all those negative aura ☺

What is the most challenging situation you have had in your Testing experience?

As you might have guessed, it depends on each tester's experience. The challenge can be technical, process-oriented, people's related, domain specific OR anything within your Testing experience.

- Testing a technically-challenging application such as social-connected applications, application involving multiple third-party interfaces, cloud-big data-IoT-Analytics-etc. applications.

- Maintaining a balance between design/execution efficiency and defect detection. Improving the Test coverage. Sticking to agile principles.

- Resources on unplanned leaves, some even abscond. Estimation gone wrong, requiring weekend & extended working hours. Rating discussions with unhappy employees.

- A team of freshers with limited domain knowledge. Planning for knowledge transfer sessions.

The thumb rule – be realistic & tell your genuine challenges!

What's your approach to finding defects?

Frankly speaking there is no silver bullet. No one tip, trick or technique. There is no one process or method. No single approach or strategy. Software Testing aimed at finding bugs is a vast discipline in itself. It takes years of experience, just to learn new aspects and get better at it every year.

In order to find defects, we need to cover some basic test buckets,

1. Go one module (or functionality) at a time. Quickly verify the happy flow.
2. Check all the validations.
3. Verify expected errors using no/invalid data, i.e., the Test Data.
4. Make sure you have covered all functional flows (i.e., options like Save, Edit, Submit, Cancel, Open, Close, Refresh, etc.)
5. Cover all possible alternate flows.
6. Check for other out-of-the-box features like concurrency, in-transit data, history/audit, compatibility, etc.
7. A tester's 'attention to details' & 'logical thinking' comes in handy here.
8. At last, do some ad-hoc tests with the intent of finding defects.

The application is currently in production and one module require code changes, i.e., Change request. Is testing the module-under-change enough to ensure quality delivery? Why or why not?

No. Testing the module-under-change is a must-have whereas the changes might have an impact on the integration with the other modules. Most of the development teams follow modular coding wherein same functions are re-used in different modules of the application. If a change is made to a particular module, it doesn't guarantee that the change is isolated to that module.

That's where 'Regression Testing' comes to rescue. After a thorough impact analysis of the changes, Test team needs to verify all the impacted areas – be it within same module OR outside it (ensuring that there are no side effects of the code changes).

Tip: Test team's understanding and Inputs from the development team are a must to identify all the impacted areas!

What's your approach to finding defects?

Frankly speaking there is no silver bullet. No one tip, trick or technique. There is no one process or method. No single approach or strategy. Software Testing aimed at finding bugs is a vast discipline in itself. It takes years of experience, just to learn new aspects and get better at it every year.

In order to find defects, we need to cover some basic test buckets,

1. Go one module (or functionality) at a time. Quickly verify the happy flow.
2. Check all the validations.
3. Verify expected errors using no/invalid data, i.e., the Test Data.
4. Make sure you have covered all functional flows (i.e., options like Save, Edit, Submit, Cancel, Open, Close, Refresh, etc.)
5. Cover all possible alternate flows.
6. Check for other out-of-the-box features like concurrency, in-transit data, history/audit, compatibility, etc.
7. A tester's 'attention to details' & 'logical thinking' comes in handy here.
8. At last, do some ad-hoc tests with the intent of finding defects.

The application is currently in production and one module require code changes, i.e., Change request. Is testing the module-under-change enough to ensure quality delivery? Why or why not?

No. Testing the module-under-change is a must-have whereas the changes might have an impact on the integration with the other modules. Most of the development teams follow modular coding wherein same functions are re-used in different modules of the application. If a change is made to a particular module, it doesn't guarantee that the change is isolated to that module.

That's where 'Regression Testing' comes to rescue. After a thorough impact analysis of the changes, Test team needs to verify all the impacted areas – be it within same module OR outside it (ensuring that there are no side effects of the code changes).

Tip: Test team's understanding and Inputs from the development team are a must to identify all the impacted areas!

Ever had an argument with your manager? What was the case & how did you handle it?

The most common: after all the hard work you get a bad rating. Manager states that it's 'relative'. There is no 'weakness' in IT industry, only improvement areas.

- Argument over some Test metrics, be it the status report, defects metrics, productivity, design efficiency. The conflict in understanding.
- Some process doesn't make sense to you?
- Micro-management is demotivating?
- Manager playing favoritism & supporting his aides?
- Too demanding? He/She just needs the work done, even if that means you doing extended hours or working on weekends.
- Task allocation or Project management conflicts?
- Manager: People are too careless for you to be strict?
- People don't understand how this works. It's relative. Even if you are good, only the best of the best gets 1 – rest we have to maintain a pyramid.

It can be anything. The Manager – Employee relation is somewhat tricky. Employee thinks Manager is clever/shrewd, Manager think employee has attitude problems. Guess "communication-to-understand" is the only key.

One of the most common Testing situations – How do you handle timeline crunch?

Since Testing is the last step before client demo, the Test team has to make-up for the delays encountered till the build is deployed in Test environment. Now how do you handle crunch timelines without impacting the quality?

- Risk-based Testing: Prioritize the Test efforts [executing the cases in order of priority] and communicate the same to all the stakeholders.
- Utilize Test automation to the fullest – to cover all regression aspects.
- Consider some buffer in planning phases to accommodate the usual delays.
- The most common: Extended & Weekend working hours.
- Plan & make efficient use of working 9 hours.
- Convey the situation as-is to the stakeholders, asking for some time to complete testing.
- Quicken the defects turn-around time by communicating it to the development counterparts.

Whatever be the approach, always take a retrospective look once the testing is complete in order to avoid delays the next time.

What to do if defect is not reproducible

Now that's a tricky one. Many a times we face these kind of one-off bugs ☺ which peep-out and then hide somewhere. "It was a one-off bug and now not reproducible – so what can I do?" Wrong! Though one-off but still it is present somewhere in the software and as a Tester it is our responsibility to investigate it. How?

- 'Think' of the exact steps and then try to replicate.
- Environment plays a vital role. Think of the environment, what has changed since then.
- Logs! Yes, logs are a great source of narrowing down on the specific module.
- Collaborate with a developer to validate the specific code.
- Test Data can be a root-cause.
- Identifying if there is a pattern to it. Say, after every 3 attempts.

Sometimes these one-off bugs can prove costly in the live environment. So, it is important to analyse carefully.

Note: Invest your time in a one-off bug depending upon its severity. It doesn't make sense to work on a low one-off defect for a complete day ;-)

The QA team starts testing a software/product and there are "way too many" defects. Every other scenario is failing, new flows are explored & clarifications sought. What would be the strategy now?

It's a complex application with tight schedule. Too many defects add cherry on the cake. The blame game starts. But at the end, development & QA are expected to work as a team & deliver the product/software. What are the options?

- Reject the build. But that would mean delay in delivery.
- Get into a war room, daily. It helps to triage the defects & increase the seriousness.
- Prioritize. Work as team & ensure priority flows are working as soon as possible.
- Identify super-devs & super-QAs to take up some additional tasks & ensure minimum turn-around.
- Involve the higher management (risk aware) just in case something goes wrong. It shouldn't come as a surprise.
- Since timelines are fixed, make sure turnaround time is as minimum as possible – be it defect fix, retesting, business clarifications, anything. Everybody has to be on their toes.

Note: Retrospective w.r.t. estimations, build quality, management screw-up, etc. is altogether a different discussion.

What to do if Agile User story is delayed?

First things First | Update Product Backlog & Re-prioritize - When a product backlog item has undone work at the end of an agile sprint, it should first technically be put back onto the product backlog. The Product Owner re-orders the backlog. Typically, unfinished items end up at the top, but not always.

Undone means Undone | The Undone Perspective - This model keeps things simple and keeps teams from gaming the numbers. The idea is to put back 'complete' story (do not resize the items to represent only the remaining undone work) back on the product backlog for re-prioritization. Do not include any effort spent on the undone user story in the velocity calculation of the current sprint. On average, over a number of sprints the velocity figures will average out in any case.

But something got done | The Done perspective - If we can take some work to done so it can be inspected, do it! Work was done and we want to know how much work remains in the release. Split the User story. There are many cases where the team realizes that a PBI can be decomposed into even smaller chunks of done work. If teams cannot break a story further into work that can be delivered as done and remaining undone user story, then it should go back on the Product Backlog wholesale. The team may choose to re-estimate due to the new knowledge they now have.

Do you have working experience in an Agile environment? What's your Team composition?

We do follow Scrum methodology with 2-week sprints including Sprint Planning, Daily Stand-ups, Demo sessions & Sprint reviews. The roles include,

Product Owner: Champion for Business! The communication bridge b/w the team and stakeholders – writes customer-centric user stories, prioritize, maintains Product Backlog, Demo sessions, define releases, communicates team status, etc.

Scrum Team: Champions for sustainable development & QA! Responsible for delivering potentially shippable increments of product at the end of each Sprint. A mix of software engineers, architects, programmers, analysts, QA experts, testers, and UI designers.

Scrum Master: Champion for Scrum! Coach the team, the product owner, and the business on the scrum process and look for ways to fine-tune their practice of it.

Scrum team is the car, product owner - the driver and Scrum Master - the chief mechanic :-)

Do you have any experience with Office friction? When a colleague was not happy with you. Or anything else?

"A person who is always fighting OR the one who never fights – both are not good in the long run."

Disagreement is the key to innovation. Debate is a powerful means to understand different perspective. Friction is important to develop inter-personnel skills.

With a lot of people working together, friction is bound to happen. And it has happened with me as well. It is a part & parcel of workplace.

It is not so much about the friction – but the learning that you get out of it. It can be a debate about which technical approach to take, or as simple as work timings. What if a junior is rebellious? Not adhering to certain policies? Or a senior who doesn't understand your perspective? It can be anything.

We don't learn as much from the success, as we learn from our failures. Friction is good, if you are learning something. Something practical.

Being a QA Lead, how do you handle a non-performing employee?

Do your best in a tough situation,

- Don't ignore the problem. Like a bad sore, if ignored too long - it can lower the morale and productivity of other employees.
- Don't make any assumptions.
- Listen. Communication is the key – talk with the person in private & find out the real problem. There may be external, personal factors influencing their performance.
- Keep it professional. Give clear, behavioral feedback. Coach the employee and lay out the plan together. Set consequences if things don't change.
- Help the problematic employee to get back on track.
- Follow-up. It's going to take time and ongoing help to change habits. Keep track of the performance in the specific feedback areas.
- Encouragement is important. Praise and reward positive change.
- Don't poison the well. Respect confidentiality - Don't trash talk to other employees. Just don't do it.
- If there isn't improvement, take action. Formally act on continued underperformance - work through the company's processes, if necessary.

- Lead by example: When you are open, honest and working, employees are more likely to do the same.

And hope for the best!

Ever worked with a challenging client? Who is adamant & rigid? About the design, requirements, process or timelines. What was the situation & How did you tackle it?

Sometimes we do get that unfortunate project (fortunately there is much to learn from it) where the client is uncompromising – they think they are Google who knows everything OR just that you are a labor working for them.

- Don't panic. Don't react. Instead respond. Think it through what client is demanding & analyze your options at hand.
- Consult your seniors on how to handle the situation.
- Be flexible in accommodating client requests, to whatever extent possible. Two adamant sides won't solve the problem.
- Pull in subject matter experts to talk it through with client, hoping that he/she will at least listen to the respective SMEs.
- Maintain a balance between your team & the client. No one should suffer for the other.
- Report it to the higher management to take care at their level.
- We are no laborers. Clear it with the client or else say No.

Guess there are so many different ways to either make it or break it.

How do you plan resourcing?

Resourcing, i.e., in simple terms how many people are required for how many days to deliver a product.

The first pre-requisite to resource-planning is **estimation**.

Say the total effort estimation comes to around 150PDs, i.e., 150 Person-days. A single person would take 150 working days to deliver the product.

The second pre-requisite to resource-planning is the **timelines**.

Say the project timelines are for a month, i.e., the product needs to be delivered in a month's time. 20-working days.

Combining both estimations & timelines,

A single person would take 150 days. But we want it delivered in 20 days. The calculation says we need 150/20 = 7.5 people to deliver.

Adding buffer as risk contingency, we need 8 people to deliver a product estimated at 150 PDs in a month's time.

Tip: It's not that easy though. Multiple other factors come into play – third-parties, systems involved, test automation, timelines, methodology being followed, etc.

What levels of testing have you worked on?

Before answering, first you should be aware of the levels of testing – i.e., unit tests, system testing, integration testing, end-to-end tests, user-acceptance-testing and business validations.

Next, we need to be true about our experience at difference levels. Say,

"System testing makes up the most of my professional experience including Selenium automation & sometimes database validations using SQL. Have also worked on integration tests as part of ABC & XYZ projects which involved web service testing using SOAPUI. Worked on end-to-end testing as part of MNO project which involved validation of end-to-end payment processing. With respect to UAT, we only did Sanity tests in the UAT environment before handing it over to the UAT team."

This gives interviewer an idea about your knowledge of Test levels and your practical experience of different levels-process-and-tools.

What is different between Web and Mobile testing?

Actually, Mobile testing brings in more complexities in terms of device fragmentation, cellular networks, mobile data, hardware complexities, different OS…there are way too many combinations when it comes to Mobile – different companies, hardware, software OS, screen size, networks, bandwidth, geographies, etc. That's where testing on emulators in the cloud helps.

What if,

Battery is drained out? Suddenly there is no (or minimal) network? What about call/message interruptions? Will the App behave similarly for Android & iOS? What if there isn't enough memory left? Yeah! Guess Mobile Testing is much more challenging than Web.

How do you prove the leadership that Testing was 'completed' diligently?

We do have a 'Requirement Traceability Matrix' to track requirements coverage and Test execution. Additionally, the Daily Reports and the Overall Test Report helps in tracking.

Interviewer: Let me rephrase - Say you have 100 Test cases. The team execute it all & you share the Test Report. But how you as a lead know that 100 test cases were executed diligently and not just for the sake of it?

Being a Lead, I work closely with my team members. I know the type of defects being raised, the clarifications being sought, the challenges being faced – which builds the overall confidence in the quality of testing. I couldn't think of any formal document/process to know about it.

What's the major difference between Agile & Traditional mindset?

As the name says, Agile is more responsive. Traditional approaches were more rigid in accommodating the changes.

Agile team calls for more collaboration and communication instead of documentation & well-defined milestones.

Agility embraces the change. It is more realistic. Traditional mindset is to first freeze the requirements and then accept changes only in later releases.

Instead of top-down management, Agile encourages bottom-up collaboration. It encourages self-organizing teams which are self-driven.

What can be done for Defect Prevention? I.e., to avoid defects reaching the Test phase itself.

- Analyse the requirements well & good. During refinement, discuss in detail about the possible impact of changes on other dependent modules. Might uncover some ambiguities.

- Build a good Unit Testing process (which is never done diligently. Developers know that it will be tested by QAs anyway).

- Test-driven development (TDD) might help in ensuring that the developed requirement has passed the test.

- Code reviews might help (but are seldom done due to timeline crunch).

- Retrospect about the release and implement the best practices. Learn from the past mistakes, improvise.

What metrics do you capture in your QA project?

Metric, i.e., measurement. Are metrics important in Software Testing? Yes, of course. Without metrics, how do you measure, define, showcase or report your efforts or the product quality?

- Estimating a Testing project, i.e., the Test Estimation.
- Requirement Traceability Matrix (RTM) to measure the requirements coverage.
- 'Defects' are at the epicentre of any Testing effort. So, defect metrics makes sense. Example – Total defects, P1/P2/P3 bifurcation and Defect leakage.
- Test Execution metrics (or Daily Execution Status) to measure the progress of QA efforts and the time remaining.
- Test Automation coverage metrics.
- Test Automation ROI [Return-on-Investment]
- Agile velocity and Burndown charts

How to handle multiple third-party dependencies?

Third-party as in – any organization other than you and the client. Multiple third-party interactions are most common in End-to-End testing. Say a payment originates from the front-office (managed by Org-A) >> is routed by Mid-office (Org-B) >> processed by Back-office (Org-C) >> Cleared & Settled (Org-D) >> Reported for Account Statement (Org-A). In between there can be other third-parties as well to fetch the FX rates, or some other inputs.

Communication & Coordination is the key here. Since multiple Orgs have different work culture and might have different locales – coordination is extremely important.

- Quick & clear communication among all helps in sharing the information/status.
- A daily catch-up meeting with all third-parties is important to gauge the readiness for today & progress till now.
- Any downtime has to be zeroed-in and resolved quickly, for others to progress.
- Having a Command-central is important to resolve any conflicts and avoid deadlocks.
- Integration between different systems should be clearly defined and communicated, to avoid any rework.

What do you do if a defect is caught by end-user in production environment?

Ah! The worst scenario for a Software Tester. There are situations when defects leak into production and are subsequently caught by end-users. Now what?

First & foremost, depending on the priority/severity - it has to be fixed & delivered as per the defined SLAs. For that, we need to replicate it in the test environment in order to identify the root cause.

It is fixed. Fix impact is analysed. Defect is retested thoroughly and a regression is performed on the impacted functionalities. You don't want it to reoccur OR impact any other working functionality. Automation is run to ensure all the happy flows.

And the night goes by…. ☺

Once delivered – now the blame game starts ☹ who missed it & why. Why it was not caught during testing. Once we know the reason – test cases are updated to include different permutations & combinations. The knowledge gap is filled.

What are some of the agile challenges?

Every project is following 'agile' now-a-days, at least this is what they think. Call it Agilish-waterfall OR Pseudo-agile.

Like everything else, Agile also is not a silver-bullet. It also has challenges.

- First & foremost – Implementation challenges. Not everybody understands agile as it was envisioned.
- From QA-perspective, the preference to shorter delivery cycle puts a timeline pressure on the QA team.
- Communication (with Client, Product Owner, Internal, etc.) is not always open & consistent.
- Slowly the focus shifts from 'Quality Product Delivery' to just 'Quick Product Delivery'.
- No process, i.e., anything to everything, in the name of being agile.

Would like to understand the techniques you have used while providing estimates. Apart from the simple/medium/complex division of scenarios and assigning them some hours.

Estimation, i.e., approximation - determines how much money, effort, resources, and time it will take to build a specific system or product.

The basic principle is to,

1. Break down the requirements (Use cases/Epics/Stories/etc.) into tasks.
2. Estimate these tasks in light of expertise available, systems involved & dependencies.
3. Sum up.

For Step-2, there are different techniques available. Guess S/M/C with hours (based on prior experience) is the most commonly used. Some use planning poker, Fibonacci, T-shirt sizes, buckets, etc. to categorize the efforts needed – more popular with Agile.

What are the best testing practices?

In simple terms - anything that helps you understand the product & client requirements better OR helps to find defects OR build confidence in what's built.

There is no specific list. Every tester, project or Orgs have their own set of best practices. To name a few,

- **Shift-left Testing**: Test early in the SDLC when cost of fixing is relatively less.
- **Test Automation**: To reduce redundant manual efforts so that it can focus on more effective testing.
- **Collaboration**: Intra-team, inter-team and client collaboration.
- **CI-CD**: To enable quick release cycles.
- **Structured process**: It can be agile scrum, V-model, JIRA lifecycle, release timelines, product roadmap, etc.
- **Retrospectives**: a great way to improve on the process.
- **Exploratory Tests**: allocate some time in order to understand the product and business better.

The list can go on-and-on…

The motto is to keep 'improving'. Anything that helps you test better is a best practice.

When to stop testing?

One answer is – NEVER. Once you are done with your planned test cycle, take some time for exploratory tests and understand the product.

And formal answer would be – when you meet the exit criteria. Timeline and Open issues are one of the major exit criteria to provide a positive sign-off.

Generally, the decision to stop testing for a particular cycle/release is based on multiple factors like timelines, open issues, test execution percentage and coverage, etc.

How do you define 'Software Quality'? what's your version of quality?

A defect-free product! But what if it doesn't meet the requirements?

Conformance to requirements! But what if requirements are too complex for a user to use?

Fit to use and fitness for purpose!

Quality is a moving target based on the targeted market, end-users, competition, evolving technology, etc.

What if you find a major defect just before the release date?

That's a tricky situation.

Why you didn't find it earlier if it is a major defect – let's keep that discussion separate.

First, no matter what, you need to report the bug.

Second, what you do depends a lot on,

- How likely it is that the bug will occur in production.
- How much damage it will do if it occurs?
- Whether there is a workaround.

If it's a 'show-stopper' - immediately escalate to stakeholders and block the release.

If it's 'not a show-stopper' (limited damage or a workaround is available) - document it as a known issue in the release notes, then notify stakeholders.

Third, as a vendor plan for an immediate fix post-release or the next version.

Note: Sometimes delaying the release is not an option if there are overriding concerns such as contracted delivery dates with severe penalties.

You must have heard about Shift-left Testing. What does it actually mean and why should we shift-left?

A typical SDLC looks like Initiation >> Requirement Definition & Analysis >> Development & Test Design >> Environment Setup >> Test Execution >> Reporting.

As you might have understood, shift-left testing implies that the QA phase be shifted left as much as possible. I.e., testing has to be started from the Initiation & Requirement definition phase itself.

It requires testers to be involved right from the start – providing valuable inputs to enable correct requirement definition in line with customer needs. A sort of Static testing you can say.

As they say – *"The cost of fixing a bug rises exponentially as you progress through SDLC."*

How important is 'Root Cause Analysis'?

RCA, i.e., a systematic process for identifying "root causes" of problems to find a way to prevent them in future.

In terms of testing, it is mostly employed for defects-RCA. Many defects can be prevented if action-items from an RCA are actioned upon. Some common root cause can be – environmental, configurations, permissions, invalid test data, incorrect test, ambiguous requirements, code issue, etc.

Defects falling under certain categories can be prevented from the start itself, hence saving both time & efforts and focusing more on valuable testing.

In that sense, RCAs help in continuous improvement of the overall process.

Do you ever freeze the scope in agile scrum?

Some say you freeze it at the start of sprint. But then it defeats the purpose of agility.

Others say, you never freeze the scope. Any change requested during the ongoing sprint should first be analyzed and then accepted/rejected based on its priority + how big is the change to accommodate in the current sprint + your sprint goals.

What are your thoughts on mitigating and avoiding production issues?

The 7-step exercise,

1. Defects are inevitable. They will occur at some time. The perception should change from 'fix the bug' to 'fix the root-cause' driven from top-to-bottom in the hierarchy.
2. Analyze Requirements. The team should be clear on what's expected with no ambiguity.
3. Practice frequent code refactoring. It helps in improving and optimizing your product apart from clearing the technical debt.
4. Perform aggressive regression testing. Utilize automation and exploration hand-in-hand to perform rigorous regression.
5. Execute defect analysis. Analyse the defects to identify and avoid the root cause in next iteration. Develop automated tests to verify all previous defects as part of the regression.
6. Consider continuous changes, i.e., continuous integration-test-deployment.
7. If possible, integrate error monitoring software. It helps support team in early-analysis of any failures noticed.

"User Acceptance Tests are generally manual". Is it correct?

Yes, User acceptance tests are generally manual. Why? Because at the end the software/product which you are building is for the end-users and not some machine.

UAT is generally performed by end-users (unless outsourced) who are more familiar with the business flows exercised on a day-to-day basis. They are well aware of the if's and but's of day-to-day operations. Before production deployment, it makes sense for end-users to test the system for the expected functionality and impact on their daily operations.

Though some of the major business flows can be automated, but ideally it the customer's team who can actually certify the product as meeting the exit criteria and ready for production deployment.

How do you give 'Business training' to your QA team?

You cannot test 'right' unless you 'understand' the business requirement. What does a particular requirement fulfil?

Business training is not a stand-alone activity rather an ongoing one.

- The initial domain-specific training once you join a project.
- Knowledge transfer sessions – within the team, with business analysts/clients, UAT teams, etc.
- UAT test cases can be referred for better understanding of the usage.
- Defects. Yes, defects too are a good source of how the application is supposed to work. Including UAT and production issues.
- Requirement analysis before you start the test design. Clearing your doubts and ambiguous requirements thereafter.
- Client demo's and subsequent feedback sessions.

Generally, if you pay attention to all these aspects – there should not be a need for exclusive business training. But you never know – exclusive trainings can be arranged for some typical projects.

Do you have Coverage discussion OR a brainstorming session before you start your Test design?

"Two heads are better than One".

It is important to discuss the coverage for user stories before writing test cases. Often, we see people distributing the user stories and directly writing the test cases. And then rework it based on the review comments. Reviews help, but precaution is better than cure. And we know how seriously reviews are conducted 😉

It's important to discuss the requirements, and subsequent coverage. It helps in avoiding the rework + also enhances overall team's understanding of the business requirements.

Does your team really act on 'Lessons Learned'? Or is it just a document...

It is important to have lessons learned session after major milestones in the project. It has three aspects,

- Discussion on the challenges faced.
- Brainstorm to find solution to the challenges and what can we do better the next time.
- ACT!

Most of the times first two steps are executed perfectly well but we miss out on the most IMPORTANT step – ACT. Or there is no tracking of the actions taken.

Note: If same challenges/problems/issues are being discussed in subsequent sessions, you really need to focus on 3rd step. It's of no use to just discuss the problems without any resolution.

What has been your biggest challenge from team management perspective?

'Challenges faced' is one of the common interview Q. It helps interviewer to gauge your hands-on exp in tackling tough situations.

From team management perse, few challenging times –

Aligning team member goals with organizational goals – one being the push towards test automation. Not everybody is interested OR good at it. So, need to strike a balance!

A team of freshers. Yeah! Once had a team of freshers – full of energy and enthusiasm but obviously inexperienced, required a lot of handholding, training and supervision – but in the end, the project was successfully delivered.

A lenient & reckless team member. Didn't complete the tasks. Irresponsible. No ownership or work ethics. Did multiple one-to-one conversations to understand the problem, laid out a plan of improvement, helped-and-tracked the progress but eventually got to know that he quit IT (was not interested).

Last & obvious – performance appraisals. Being transparent, recurring touchpoints and no-office-politics are some things that help here.

What according to you is the most difficult part of leading a team?

People management is not easy. You constantly need to maintain that balance between individual aspirations and project/org expectations.

Personally, the most challenging part is to keep the individuals motivated – by work satisfaction – a sense that you are growing as a professional with each passing year.

How?

By listening to them - identifying and working on their professional strengths. Want to work on new technologies/tools? Or more participation in stakeholder communication? Some are good at process improvement, while others might want to become a domain SME. Want to master test craftsmanship? Interested in trainings/mentorship? There are numerous possibilities.

It is when you work with them, hear them out and then work together on the strengths – by assigning relevant work – that people feel content & satisfied with their work.

How do you build a Test strategy?

'Strategy', as the name suggests – what's your approach to testing a software/product?

The first step would be to understand the high-level client requirements and the type of software/product [web/mobile/desktop/cloud-hosted/etc.]. and then the next steps follow,

- **Test methodology**: Are you following agile or a waterfall? How is your test cycle organized?
- **Testing Types** – functional, performance, security, database, API testing, etc.
- **Tools**: What tools you would be using. Say, Selenium, Rest Assured, JMeter, etc.
- **Test Management**: How will you manage requirements, cases, defects, traceability, etc. – HPE ALM, JIRA, Inhouse tools, etc.?
- **Team**: What all team roles and expertise are required.
- **Test Environment**: What will be the build deployment process from dev to production.
- **Test Approach**: How will you focus on Unit tests, static analysis, dynamic testing, risk-based approach, etc.
- **Hardware Requirements**: Will you use any stub/drivers? Or a mobile emulator/test lab? Virtual machines for test automation? Etc.

As a Tester, what's the most important achievement in your career?

There has been many – successful deliveries, automation ROI, being involved from project initiation till closure, training & blogs, guiding and mentoring fresh talent, team awards, etc.

But if I must choose one – it would be one of the projects back in 20XX. When I took over as Module Test lead – the project was headed to fail. Environment issues, unclear requirements, low build quality, too-many defects, adamant client, tight timelines, and what not.

The only saviour was – people were willing to make it happen. And then it started,

- Daily war-room meeting with Dev – to discuss open & fixed defects.
- Agreement with BAs to reduce the turn-around time of requirement clarifications.
- Process setup to accept build only after proper unit tests.
- Chat window with release team to quickly action on unwanted environment issues.

- Close tracking of defect ageing – to minimize the defect turn-around time.
- More focus on requirement-based exploratory tests to identify maximum defects.
- Involving higher management for client communication w.r.t. estimations, CRs and the progress.
- And of course – weekend hours

At the end, we delivered – successfully. And learned a lot about crisis management.

Any experience with co-located teams?

With more remote working options, IT is slowly adopting the co-located team structure. Yes, do have experience with co-located teams where-in our team was split geographically – Bangalore/Hyderabad/Mumbai/Gurgaon/US.

Important pointers would be,

- The most important - everybody should be clear of their daily-weekly tasks and targets. Daily stand-up helps here.
- A common chat window helps for any real-time issues.
- Recurring one-to-one with team members helps to align personal targets.
- Open & transparent leadership where anybody is open to contact you regarding anything – any time.

How do you maintain traceability with changing requirements?

The main motive of traceability is to ensure that we don't miss testing a requirement. How to ensure that? By tracking that you have executed cases corresponding to each requirement.

In case of changing requirements, we need to update the corresponding test cases OR write new ones and then pull these in your test campaign. For building traceability matrix, any tool like ALM can be used to check on any missing coverage.

The catch: Now that direct requirements are covered, what about the existing impacted functionalities? That's right – the test team need to diligently put together a regression campaign based on the impacted areas.

With updated/new cases + impact analysis + regression + traceability tool – guess we have it covered now.

Say, you have a situation where freshers/juniors come to you about a problem that senior folks are not supporting them in knowledge sharing. How will you go about it?

Though I am yet to come across such a situation, but I feel the key to any team issue like this is – proper communication.

Communication (to listen more) helps you understand different perspectives around the problem and then take corrective actions.

In this case it is important to hear both sides, and then seek help from the senior folks to bring team mates up to speed. To make them understand that mentoring is one of their responsibility at this level. On the other hand, guiding the juniors on how to make the most of this opportunity by learning from other team members.

Recurring knowledge sharing sessions [say once a month] among the team members help here in developing that attitude – it can be technical/process/retrospective/product/etc.

From QA perspective, what's required to achieve a 2-weeks' delivery cycle?

"There is no agile without Test automation".

- If you are targeting a 2-weeks' delivery cycle, you must shift-left and invest more in test automation.
- Follow the pyramid – Max unit tests > API automation > basic UI automation.
- Repetitive tasks/regression should always be automated – giving time for testers to do some business/exploratory/UI tests.
- Turn-around time should be kept minimum for any inter-team tasks – be it defect resolution, requirement clarification, environment issues, etc.

The goal should be to have a CI-CD pipeline which trigger your tests as soon as the build is deployed – in order to get quick feedback.

Any experience with localization/globalization testing? How were the cases structured?

- **Localization**: adapting a product's translation to a specific country or region.
- **Globalization**: product design to support any culture or locale (language, territory or code page).

How? By separating the code (logic) from the messages/content/or information.

Many countries have certain laws for products to align with their localization standards. And you need globalization for business expansion.

Once had a federal project with specific regulation around localization.

- Localization: cases were structured corresponding to each application page to verify the content translations in different languages.
- Globalization: cases were structured to run regression tests for each supported language.

What are some aspects (or measures) to deliver a good quality, quickly?

Yes, that's a broad Q – but to start with,

- **Clear requirements**: The most important. Unless you deliver what client expects, anything else won't work. Work with your client/product/business team to understand the requirements clearly.
- **Shift-left**: Test early. Test more at Unit & API level and then moving to system-integration-UI tests.
- **Test Automation**: There is no agile without test automation. Want a quick delivery? Automate your tests.
- **Test coverage**: Don't lose sight of your test coverage mapped to client requirements. Build an effective test design strategy to cover alternate/corner cases as well.
- **Defects**: Yes, defects are a great source of learning – on where to focus more & where to increase the coverage. Don't skip root-cause analysis.
- **Retrospective**: There is no top, only further heights to reach. Retrospective helps to avoid mistakes and highlight areas of improvement.

What would be some corner cases for a mobile app - feature testing?

Yeah, a broad Q but still let's give it a try.

- The most common - Interruptions – incoming call/message, etc.
- Limited memory left/Battery down.
- Varying connection speeds for different operators [3G/4G/5G/Wi-Fi/etc.]
- P&C of App permissions
- Error message configurations
- Monitoring the app crash report.
- User mobility (in case the app uses GPS or other mobile sensors)
- Backups & recovery – app upgrade, battery down, offline usage, etc.

Some straight-forward would be,

- Device fragmentation & OS platforms
- UI/UX Testing
- Functional Testing
- Localization/Internationalization
- Security & Performance

How will you go about resolving a conflict between two people within your team?

This reminds me of the movie 'Chak De' – climax. Where two ppl in your team have a conflict and you as a coach must do something to win the final.

Communication is the key to resolving any conflict. Don't make any assumptions - have a clear discussion with both the team members to understand the real problem – personal/professional/ego/etc.

Once you know the problem - it can be tackled – by proper work allocation and defining individual responsibilities. And reminding them that 'Team player' is one of the important criteria to grow in the professional ladder.

What could be some reasons that nothing happens after clicking a button in the application?

Debugging: identifying a problem, isolating the source of the problem, and then either correcting the problem or determining a way to work around it.

Sounds easy? But a task in itself. As a tester, it is important that you can debug a problem to zero-in on the root cause. For the given problem statement, there can be multiple reasons –

- No binding: just a show-piece button with no corresponding binding to trigger any action.
- Javascript failure: The associated Javascript validation fails silently without throwing any errors.
- HTML/CSS Issue: The button is clickable only at certain dimensions OR not interactable at all.
- Browser-compatibility: Tech-stack used isn't working with older browser versions/specific browser.
- Pre-requisites: Say it's a flash object and it isn't enabled on your machine. Or say Javascript execution is disabled.
- Firewall/Security/Blockers: Some security software or firewall settings are blocking the call-to-action. Some browser add-on you installed might be blocking the click.
- Cache/Browser refresh: Some cache entries disallowing button click. Or a browser refresh loading the page correctly again.
- Code-issue: it can be a code issue as well

After production, will you pick all test cases which you have written before the production for next cycle or only new requirements test case will be considered?

The new feature test cases + regression test cases from previous release.

Regression Testing (No side-effects)

You never know a change in one function (new requirement or defect fix or enhancement or change request) can impact multiple areas of the software. It's our duty as a Test team to ensure everything (apart from the new changes) impacted is working as expected. In other words, to ensure that previous delivered functionality is working even after the new changes. As you might have guessed, knowing the impact (Impact analysis) is a must-have to perform effective regression tests!

Why we need retesting when we already go through regression testing?

Retesting (Recovery Health check-up, post medicine)

This is the simplest to understand. What do you do in testing? Obviously find & log defects. After that? Yeah! Developer will fix the defect. As a Test team you need to verify that the defect fix is working fine, in other words you need to 'retest' the defect based on its steps to reproduce. Simple, right?

Regression Testing (No side-effects)

The code is developed >> Build is deployed >> Sanity is performed >> Full testing is done >> Defects are logged, fixed & retested. What else? Yeah! Truth is stranger than fiction, and so is the Software. You never know a change in one function (defect fix or enhancement or change request) can impact multiple areas of the software. It's our duty as a Test team to ensure everything (apart from the change) impacted is working as expected. In other words, to ensure there are no new defects introduced. As you might have guessed, knowing the impact (Impact analysis) is a must-have to perform effective regression tests!

How I handle team conflicts?

"Managing conflict is one of the biggest challenges a project manager faces."

- Solve the underlying problem: Take the time to thoroughly understand what is causing it. E.g., lack of training, poor communication, unclear expectations or goals, etc.
- Acknowledge the person: Some conflicts occur because a person's ideas and feelings are not being acknowledged as important. By taking the time to acknowledge your team member's problem, you could prevent any ensuing conflict from occurring.
- Call a meeting: Ask each party to present their side. The formal structure of a meeting helps people structure their thoughts. By getting everyone in the same room you have a better chance of coming to a resolution sooner than later.
- Listen: By gathering more information through listening, you'll be better equipped to solve conflicts.

- Understand each team member's viewpoint - listen and understand, to truly get to the bottom of the conflict. Now, it is time to ask the team for a solution. Since, everyone is in agreement that completing the project successfully takes priority, each team member would also be aware that the resolution strategy they are offering, is truly beneficial for everyone involved. Giving the respect and space to share their thought process also adds responsibility and accountability to each team member.
- Exercise authority when required. In high-risk situations, you cannot afford to keep the conflict dragging, hence it is advisable that you wield authority in order to maintain your stand on the proposed solution.

Conflict in the workplace is an ever-present fact. By implementing effective conflict management practices, you can turn your challenges and disagreements into positive resolutions for everyone.

How I will manage a large complex project end to end?

It depends on the size of the organization and the risks involved. For large organizations with high-risk projects, serious management buy-in is required and a formalized QA process is necessary. For medium-size organizations with lower risk projects, management and organizational buy-in and a slower, step-by-step process is required. Generally, QA processes should be balanced with productivity, in order to keep any bureaucracy from getting out of hand. For smaller groups or projects, an ad-hoc process is more appropriate. A lot depends on team leads and managers, feedback to developers and good communication is essential among customers, managers, developers, test engineers and testers. Regardless of the size of the company, the greatest value for the effort is in managing requirement processes, where the goal is requirements that are clear, complete and testable.

Which all Testing levels have you worked on?

Before answering, first you should be aware of the levels of testing – i.e. unit tests, system testing, integration testing, end-to-end tests, user-acceptance-testing and business validations.

Next, we need to be true about our experience at difference levels. Say,

"System testing makes up the most of my professional experience including Selenium automation & sometimes database validations using SQL. Have also worked on integration tests as part of ABC & XYZ projects which involved web service testing using SOAPUI. Worked on end-to-end testing as part of MNO project which involved validation of end-to-end payment processing. With respect to UAT, we only did Sanity tests in the UAT environment before handing it over to the UAT team."

This gives interviewer an idea about your knowledge of Test levels and your practical experience of different levels-process-and-tools.

Which difficult situation do you come across while handling a team?

Answer to this depends on your personal experience with team handling. Because anything fake will be caught in next counter-Q. If you have actually handled a team, of course there would have been challenges - team member thinking that you play favouritism, micro-management, everyone wanting to do test automation, someone wants to lead a module, team facing technical challenges w.r.t tools to be used, inter-team conflict, etc. etc. The list is endless…

So, you need to pick up a challenge from your personal experience and then discuss on how you solved it.

What if Developers reject your defects and you are able to reproduce it on your machine?

Obvious – Pick up the phone (or walk up to the developer) and discuss.

If it is something related to requirements – clear the understanding with product owner. Product owner holds the authority to call a requirement issue as valid/invalid.

If environmental – reproduce the issue with same environment setup and let know these environment variables to the dev.

Most of the times, developers reject a bug because it is not reproducible on their system. In that case compare the setup – browser, user, environment, exact steps, test data, etc. If it is reproducible on your system – there has to be a catch.

Tip: Checking the logs at failure time helps a lot!

What you will do if you have conflict with your manager on attitude or any other opinion?

The most common: after all the hard work you get a bad rating. Manager states that it's 'relative'. There is no 'weakness' in IT industry, only improvement areas!

- Argument over some Test metrics, be it the status report, defects metrics, productivity, design efficiency. The conflict in understanding.
- Some process didn't make sense to you?
- Micro-management is demotivating?
- Manager playing favouritism & supporting his aides?
- Too demanding? S(he) just needs the work done, even if that means you doing extended hours or working on weekends.
- Task allocation or Project management conflicts?
- Manager: People are too careless for you to be strict?
- People don't understand how this works. It's relative. Even if you are good, only the best of the best gets 1 – rest we must maintain a pyramid.

It can be anything. The Manager – Employee relation is somewhat tricky. Employee thinks Manager is clever/shrewd, Manager think employee has attitude problems. I guess "**communication-to-understand**" is the only key.

Will have a 'one-to-one' discussion with my manager to understand his perspective and put across mine. Open communication is the key to solve many of the team problems including this.

If yesterday the application was working fine and today it's not working, then what steps you will take to find out the root cause?

- First – check if recently any deployments happened. If yes, what were the changes deployed which might impact the functionality.
- Checking the application logs to get the error and then finding it's reason.
- There might be environmental issues such as server load, memory issues, etc. Try restarting the application servers.
- Once these factors are rules out – check your test data, steps taken, user permissions, etc. Try to identify what parameter changed from yesterday.
- Once you get the steps to reproduce – it is then easy to zero in on the exact root cause by discussing it with the developer.

What will you do when release is near, and you must execute all test cases?

- Adopt risk-based testing, i.e., prioritize the Test efforts based on P1-P2-P3 test cases and communicate the same to all the stakeholders.
- Utilize test automation to the fullest to run regression tests in parallel.
- The most common: Extended & Weekend working hours.
- Plan & make efficient use of working 9 hours.
- Quicken the defects turn-around time by communicating it to the development counterparts.
- Convey the situation as-is to the stakeholders, asking for some time to complete testing.
- Consider some buffer in planning phases to accommodate the usual delays.

Whatever be the approach, always take a retrospective look once the testing is complete in order to avoid delays the next time.

What are the different reasons a service can go down?

There might be different reasons,

- Server downtime
- Bad Test data
- Inconsistent dependent APIs
- Requests overload

What you can do?

- Check status codes that are not HTTP 200 OK to identify API transactions that fail.
- Monitor CRUD operations like POST/PUT/DELETE
- Validate payloads using JSON Schema validation
- Check payload data (using JSON Path or XPath)
- Identify latency by checking API response times

When there is out of memory exception then what would be your next step and how do you find the root cause?

Few Causes:

- Java heap space --- object could not be allocated in the Java heap.
- GC Overhead limit exceeded --- the garbage collector is running all the time and Java program is making very slow progress.
- Requested array size exceeds VM limit --- the application (or APIs used by that application) attempted to allocate an array that is larger than the heap size.
- Metaspace --- Java class metadata is allocated in native memory.
- Request size bytes for reason. Out of swap space: --- when an allocation from the native heap failed and the native heap might be close to exhaustion

Identify Root cause: If cannot find it from the stack traces or logs - Generate a heap dump on OutOfMemoryError exception > Reproduce the problem > Investigate the issue using the heap dump file, it has all information about the memory usage of the application.

How did you handle a difficult situation?

One of the Q to gauge your real-time experience with tough situations. It can be anything,

Way too many defects during a release impacted the test progress and product quality. War-rooms were set up for daily discussion on open issues, defects turn-around time was tracked for quick resolution, unit tests were made more streamlined, effective collaboration between Bas, Dev and QAs, etc.

A team member not performing well. Had a one-to-one discussion to understand the concern. Chalked out a plan for recovery with proper milestone tracking. If it didn't work, gave warnings and checked with HR for the company policy on PIP.

Crunch timelines for one of the releases. Utilized test automation to the fullest to cover regression tests. Prioritized the test efforts based on P1-P2-P3 test cases. Extended hours and some weekend. But yes delivered it on time.

You need to think of a tough situation you were in and what steps were taken to tackle it. It can be anything – technical, process, team, etc.

How to deliver quality product when there is less time?

Since Testing is the last step before client demo, the Test team has to make-up for the delays encountered till the build is deployed in Test environment. Now how do you handle crunch timelines without impacting the quality?

- Consider some buffer in planning phases to accommodate the usual delays.
- Prioritize the Test efforts (risk-based testing) and communicate the same to all the stakeholders. I.e., execute the Prio-1 cases first followed by P2 and P3.
- The next important task to do is to utilize test automation effectively. Test Automation should cover your regression quickly.
- The most common: Extended & Weekend working hours.
- Highlight the delays to buy some extra time.
- Plan & make efficient use of working 9 hours.
- Quicken the defects turn-around time by communicating it to the development counterparts.
- Convey the situation as-is to the stakeholders, asking for some time to complete testing.

Whatever be the approach, always take a retrospective look once the testing is complete in order to avoid delays the next time.

What makes You a Good Tester?

A good Tester is built on not just one, but many important traits.

Curiosity. Or in other words - an appetite to learn. Always curious about anything new (or different).

Technical. A good Tester is technically strong. Technically doesn't mean only programming - he/she understands the Tech stack.

Observation. A keen observation to identify the anomaly ;-)

All these are important, but - "**Being Logical**" has the utmost importance. I have never seen an 'irrational' good Tester :) We as Testers have to understand the business, technology, product & process. Being logical helps in every area :)

Why Manager is expected to have hands-on automation experience when all s(he) has to do is managing?

Industry has moved to automation-centric interviews & hiring, but is it really required at Manager+ Level?

A manager should know about what s(he) is managing. One of the biggest bottlenecks to automation success is its mis-understanding at higher levels. When you have hands-on experience, you know –

- Automation takes time, it's not auto-magic.
- How to set realistic targets w.r.t. automation.
- Invest in automation from a long-term perspective.
- The challenges faced with automation.
- And, when the team member is bluffing about some automation aspect ☺

Overall, if a manager has hands-on automation experience – it's really good.

QA is NOT just to give the Sign-off!

"We have a release on XYZ date. Please provide the sign-off by ABC date."

A single line (either written or oral) from the higher management (often delivery managers) defeats the overall purpose of software testing.

- We have a release on XYZ date --- it has already put a pressure on the QA/Testing team about the time-box available for test activities.
- Please provide the sign-off --- it sounds like people just want the sign-off to go live instead of a test report providing information about the build quality. It is taken for granted that we need a sign-off.
- By ABC date: What about the quality of testing? Apart from the QA team itself, most of them are not bothered about it.

Projects with this attitude are bound to fail in the long-run. Projects who don't pay attention or don't give due importance to the QA/Testing activities are bound to fail in the long-run.

What's your approach to Testing – Execution or Defect Identification?

Test Design & Execution: Understand the requirements >> Write all necessary Test cases >> Execute the Test cases >> Mark them either Pass or Fail >> If failed - raise a defect >> Report.

Defect Identification: Understand the requirements >> Write all necessary Test cases >> Execute the Test cases and explore the application with the aim to find defects >> Raise defects >> Report.

At first glance, both looks the same. But there is a slight difference between priorities.

How do you bring order to exploratory testing?

By definition, exploratory means experimental, probing or investigative. People usually treat their ad-hoc tests as exploration. To some degree – yes. But exploratory tests are much more than that...

- **Session-based**: Uninterrupted block of test time with a particular mission. Charters are used to define the scope for exploration.
- **Scenario-based**: Parallel scenario identification as you explore the application.

Can automation QA do without QA concepts?

As far as automation is concerned – s(he) can manage with his/her programming skills. Agile anyways applies to all irrespective of your skillset.

But somewhere I feel it is important to know about the QA concepts. Testing is not just about requirement-cases-execution-defects. Your domain knowledge and business understanding play an important role. Additionally, your attitude towards 'quality' is of utmost importance.

For many people Agile means following Scrum?

Agile approach stands for evolving requirements and solutions through collaboration. It advocates adaptive planning, evolutionary development, early delivery, and continual improvement, and it encourages rapid and flexible response to change.

Any development method following this approach is 'agile' – and Scrum is just one of it.

Some other agile methods that I have heard of – Extreme programming, Kanban, Lean Software development, Scrumban, etc.

And then there are some development practices which support these agile methods – Daily scrum, TDD, BDD, Continuous Integration, Timeboxing, etc.

How will you handle if QA team finds 2 major defects just before the release date?

Unfortunate scenario. Why? Because critical/major defects should always be caught earlier in the test phase. But you never know with multiple deployments…

Now that the situation has come up,

- Document the defect in detail – environment, configs, build version, test data, user setup, steps, expected and actual results, any logs, screenshots, etc.
- The next step would be to identify if any workaround is available for that functional flow, or is it a blocker.
- Then it's time to inform the internal stakeholders – scrum master, agile team, managers, business analyst and product owner. As a QA, it is our responsibility to inform stakeholders about the build quality.

Whether to go ahead with the release OR delay it is entirely management decision, but our inputs will play a critical role in decision-making.

- Say there is a workaround available, then we can go-ahead with the release and deliver a hot-fix in the next deployment window.
- If there is no workaround, the impacted functionality/customers [business impact] need to be assessed. Can we take that risk? If Yes, go ahead. Else – delay the release.

The Go/No-Go decision will be taken after discussion between internal POCs and the Client.

Note: Such situation happens. It's good that at least issues were caught before go-live. But it does need a retrospective as to when it was introduced in the system and if it could have been caught earlier.

Ever done a mistake? What was it and what did you learn out of it?

The idea is to gauge your observant and learning attitude.

Everybody makes mistakes. If you can re-collect any one from your personal experience – that will be the best.

Example – Once there was a project where I neglected the important of regression testing. Just executed the cases without any exploration. Later on, business identified a major regression defect in the same module. It came back to me in the root-cause analysis meeting. That day, I accepted that regression testing is very important before going live/demo.

Do you follow Agile? What's the Sprint duration and different activities?

Yes, we do follow agile with a 2-week sprint model. Agile squads [Dev + QA + BAs + DBAs + Scrum Master + etc.] are distributed according to the products/modules/projects. We have daily scrum in the morning (or evening) to discuss on daily activities. Before the start of sprint, we have sprint planning meeting to discuss which user stories should be picked up for current sprint. Then at the sprint end we have sprint demo to the client/product owner for the done features.

Regarding retrospective, we usually don't have it after every sprint but every alternate sprint, i.e., monthly. Also, we have a reoccurring backlog grooming session as well with the Product Owner as & when it is scheduled.

What will be your top 5 to 10 scenarios to test ABC application?

One of the Q frequently asked in any interview. Either they will open the application like e-commerce/travel/etc. OR just ask you to give scenarios verbally.

Before you start giving the test scenarios, take a pause and think about the application and its modules/functionalities.

- Homepage rendering on different browsers – covering both web and mobile.
- Does it require registration? Sign-Up scenario.
- Most of the applications need a login, verify login with different role users [authorization + authentication] – sign-in and sign-out.
- Testing of 'Search' within the application, i.e., search results with different test data combinations.
- Test to check any broken links, i.e., different categories listed, header & footer notes, etc.
- Any integration with outside-apps like OTP/Email verification/etc.

Apart from these, interviewers are impressed if you cover other aspects as well apart from just functional cases,

- UI/UX design, i.e., easy-of-navigation, colour-scheme, image-rendering, scroll bars, font-size, neat-and-clean vs. cluttered, etc.
- Localization and Globalization testing, i.e., if the application is to be used in different geographies/languages.
- Page rendering performance measurement
- Cross-browser testing
- Security aspects like cookies, certificate, secure connection, etc.
- Tests on handheld devices, say a smartphone.

You are managing a team where there are people more experienced than you. How would you handle it?

It's all about defining the roles and responsibilities. With experience they must be having a lot of knowledge about domain-tech-products-process-etc. The task would be to identify how well can we leverage their experience and to balance it with their aspirations.

Agree there might be few conflicts but then which team doesn't have it. It is all about being transparent and having a clear discussion about the roles, responsibilities, problems and solutions. It is the lack of communication that poses a greater problem.

And for the unmanageable, one-off case – will have to play the authority card, i.e., look I am leading this team and these are your roles/responsibilities and goals.

A new member joins your team. How will you go about training them?

The training should focus on - what is it that a new team member needs to start contributing to a project?

- Organization structure, i.e., where does this project fits in the overall organizational structure.
- Project overview covering different applications and their role.
- Knowledge about the technology stack being used, i.e., test management tool, test automation tools, defect management tool, internal tools, etc.
- Individual application(s) sessions with hands-on practise, say let's start with sanity tests and then functional tests.
- And one very important aspect – domain training. Even more important for a tester.

We can plan for all these items and prepare a new joiner kit to be used in any subsequent future hiring. Once done, we are good to go to utilize the his/her expertise.

What if we give you performance testing? Or Security testing? i.e., some new work which you haven't worked on yet.

Awesome! That would be great. Just that hope you give me the time to prepare and train.

I see the need for full-stack QAs who can do functional testing, test automation, performance engineering, etc. Looking at the trend it would be a great opportunity for me to up-skill and grow in the career graph.

Iterating again, the only concern I have is about the timeline. I am no expert in these areas, so would need some time before I can start contributing. The expectation shouldn't be from Day-1. Rest I am open to new learning and aligning with project work.

What were some of the enhancements that you brought in? It can be technical, process or management perspective.

One of the popular interview Qs – helps to gauge your innovative mindset.

Start with any technical change that you might have brought,

- Introducing any **new tech/tool** – cucumber/code analysis/performance/Jenkins/etc.
- **Upgrading** the existing tech stack – Junit to TestNG | HTTPClient to Rest-Assured | Selenium upgrade | Java upgrade | CI-CD implementation | etc.
- Any **process** changes – shift-left testing, moving from UI to API testing, collaboration with Dev/BAs, setting the review process, aligning with the agile process, root-cause analysis sessions, domain learning sessions, etc.
- **Trainings** – any trainings you might have delivered OR started the process – technical/domain/process.
- **Management** – one-to-one discussions, team retrospectives, fun events, team re-organization, etc.

Think it through and the best would be to keep handy few enhancements that you might have brought in the project. That way it will be easy for you to answer!

How do you do Goal setting? How to measure goals?

Goal setting is an important activity. According to me, goals act as a bridge between organizational growth & vision and individual's aspiration & career growth.

The best way is to align individual and team's goal with company's vision. **How?**

1. Understand the Company/Business Unit goals – technology adoption, process changes, domain, re-organization, etc.
2. Define how your team can contribute to the above BU goals – adopting latest technology, driving sales, product development, test automation, domain expertise, etc.
3. Once you have the team goals, rest is just the mapping of these goals with individual's skills & strength.
4. Important point to note – Communication is very important. Have a discussion with each team member on what are the BU/Team goals for this year and how team member's individual skills can help achieve it + discussion on any personal career aspirations in line with BU goals.

How to measure?

Generally, it's a good practise to revisit goals progress on a quarterly basis [if not monthly]. If you have put definitive goals with clear criteria – you can easily track the progress, quarterly.

Any challenging conversation you have had?

Oh Yes, everybody has their share of challenging times 😊 and related conversations,

- Conversation with the manager regarding my aspirations and what I need to do to achieve those. And then the appraisal discussions.
- A team member not performing up-to-the-mark. Had a one-to-one conversation with him/her to understand what's the real issue – personal/professional – and then chalked out a plan for improvement.
- Interacting with a tough client who always used to bargain on the estimated efforts and then we need to provide a detailed analysis.
- Conversation with all internal stakeholders on a delivery miss or a production defect. The subsequent root-cause analysis and retrospective measures.
- Convincing the product owner/scrum master on why we cannot include a particular user story now in the running sprint. Or why this particular test approach is not feasible.

- Communicating that test automation needs proper planning, time & resources, maintenance, regular execution and tracking in order to be successful in the long-run.
- Not to forget the daily tough situations we get in – onshore-offshore miscommunication, extended work-hours, tight timelines, framework discussions, etc.

What was the conclusion? Conclusion is always a win-win situation, a middle ground where both the parties understand each other's point and agree on a plan that is beneficial to both!

Note: You just need to re-collect a few of your challenging times and the conversations you had around it.

Why is it that UAT team find defects most of the time even when you have completed System Testing & signed-off?

There can be multiple reasons to it,

- **Business outlook**: UAT team is closer to customer experience and how the system will actually be used by end-users. Though System test team should also derive cases based on real-world usage but still there are few gaps left which UAT team discovers.

- **Test Data**: One of the important factors for a successful test. Often UAT testers have access to production-like data which uncovers defects that are missed by System Testing team.

- **Timing**: The time of testing also plays an important role. Unit tests will focus on code-level checks, system tests focus on the modules, integration & functionalities. Once the system is deployed for UAT, it is already free from basic defects and the focus is to test end-to-end business flow.

And then there can always be regression defects introduced in the system in later phases of testing which are then caught in UAT.

Ever missed a deadline? How did you handle it?

Once in a while there are projects that miss the deadlines. There can be multiple reasons ranging from resourcing, technical, infrastructure, process, etc.

What's important is how did you handle a deadline miss.

The first step is to **identify the bottleneck**, i.e., identify the reason for delay. Say, once there was a project with frequently changing timelines along with too-many defects.

Next step is to **resolve the issue**. Say, building a requirement process to streamline changes and to focus more on build quality by implementing rigorous unit & API tests.

In parallel, the most important aspect is **communication**. Delay shouldn't come as a surprise (or shock) to stakeholders at last minute. Keep all the stakeholders informed if you foresee that the timelines are slipping and might be impacted. Keeping stakeholders informed avoid any escalations later on.

Last, it's time for retrospective to avoid timeline miss in future deliveries, i.e., to learn and correct the mistakes.

Ever helped a teammate, technically? What was it?

Important Q to gauge your team work. Don't stop at just one…

Yes, as a team it all works on give-and-take. Sometimes I help others in the team and vice-versa.

The first that I can think of is **training** all the new joiners with respect to project applications, tools & technologies used, etc.

Next there have been many incidents when people get stuck somewhere in **test automation**, be it scripting OR framework changes, etc. Few times I might have helped like suggesting workarounds to solve the problem [say click, actions class, robot class, javascript, etc.]. Other times it is not just one-sided help, we usually brainstorm to get to a conclusion.

Functionally, there have been many incidents where I took a lead in explaining the functionalities to team members.

Apart from that, I won't say help but **enhancements**. Whenever someone is stuck or comes with a problem – we try to solve it collectively – say rerun failed cases automatically, use rest services instead of UI wherever possible for test data setup, ad-hoc tests coverage, etc.

Think it through, and everybody have helped someone some time – you just need to re-collect!

What will you do if your team member misses a bug into production?

Unfortunate! But defects will leak. "Absence of defects is a fallacy".

Now that it has already happened, I would rather concentrate on next important pointers,

- What is the **issue and its impact**? I.e., priority and severity. What's the SLA for production fix - how to accommodate it – what should be the test coverage – etc.

- **Retrospective**: Why it was missed? Or at what stage it should have been caught? Identifying the root cause is the first step to rectifying it.

- **Corrective measures**: How can we increase the test coverage to ensure such defects are not missed again. Can we a production coverage list? Or alternate flows checklist? A process changes, say review? Etc.

- **Meeting**: It is unfortunate to get production issues. People shouldn't take it lightly. A team meeting might help here to convey the seriousness if an issue occurs in production – it impacts the reputation!

One should NOT pin-point someone but assess the situation and take corrective measures. That's how you improve as a team.

How do you deal with 'bad feedback' from the Client?

Feedback is a way to improvement, just that how you take it constructively.

First, don't take it personally. Think from the client's perspective and you might understand why such feedback.

Take down the pointers [from feedback] to work upon. It might be about the build quality or a timeline miss or budget overflow etc.

Build a plan to tackle all the above pointers.

Note: Sometimes it just might be a misunderstanding which needs to sorted out via proper communication/discussion between stakeholders.

Ups-and-downs are part-and-parcel of a project. What's important is how you handle it to either make it or break it. And yes, always keep the internal stakeholders aware of what's happening with the client side.

If you are leading a team, how do you know which team member is better? What are some of your parameters?

A tough one indeed. Since there are multiple parameters to relatively assess the team members. Few pointers,

First & foremost – **Testing skills**. When I say skills, it means the thought process a person has when testing an application.

- Some testers are able to identify tough defects by exploring different alternate flows while others stick to just test cases.
- Few understand the requirements from client's perspective and ask Qs which are logical.
- Some collaborate with development-business-DB-etc. teams very well to understand how the system works.

Independent: The next important parameter is independence, i.e., if the team member is able to take complete responsibility of the assigned task and understands the accountability. You need not follow up again & again. He/She delivers without question.

Team Work: How well he/she is within the team. A helping team member will always take the lime-light.

Pro-active: Few members wait for the tasks while others are proactive to ask about it or find ways to do it better.

Big Picture: Not everybody understands the bigger picture of business landscape-technology-applications-internal hierarchy-etc.

Stakeholder communication: Networking is an art not everybody is good at. There are only a few who gel up with most of the people. It is NOT important to know many people, what's more important is that how many people know about you!

Note: These is not an exhaustive list. All these traits work hand-in-hand to build a professional that climbs up the ladder faster than the others.

If I ask your manager, what would he/she say is your biggest strength and biggest improvement area?

Everybody has their own strengths and improvement areas. Just make sure it is something that won't impact your candidature. Improvement area [or weakness] should be like it can be turned in your favour 😊

Strengths [easy part],

- A team-player who is always ready to help.
- Quick-learner, understands quickly and won't ask it again.
- Responsible, give me a task and you need not follow it up.
- Technical, I am a tech-person who loves to explore new tools & technologies.

Improvement areas/weakness [tough part],

- Learn to say No, I am usually flexible enough to accommodate any requests which sometimes backfires in the sense that I have to stretch.
- Big-picture, my manager always used to tell me – "You are good but to be at the next level - understand the big picture".
- Networking, expand your presence among development-business-client teams.
- Personally, I feel I have missed few chances because I am non-political. But that is how I am!

What is your career inclination? Managerial track or the technical track? And why...

One of the important Q seeing the change in QA landscape with Agile.

a. I am a people's person. I like to work with a team, manage their aspirations in line with company goals, handle project deliveries, business communication and reporting. I feel my communication and people management skills are good enough to pursue the managerial track.

b. I am a techie. I love to explore new tools and technologies. I learn by trying different technical approaches. Also, with the advent of agile – QA landscape is quickly adopting the technology-first approach. I feel my skills are more aligned to pursue the technical track.

c. Both! My dream role would be to engage people for project deliveries and also, I don't want to leave the technical track. Would love to work in a role where have to manage a team and also get some hands-on exposure to technology as well.

The choice is completely personal in line with your career goals and aspirations!
