**Tell me about yourself?**

Thank you for the opportunity. I’m Thippeshanaik L, a software test engineer with over 3.5 years of experience in manual testing and ETL/data testing.

I specialize in validating end-to-end functionality, data accuracy, and quality of software applications.

I have worked in domains such as banking, where I was responsible for test planning, creating detailed test cases, executing them, and reporting defects using tools like JIRA and ALM.

I also have solid experience in ETL testing, where I validate data flows from source to target, including data transformation rules and SQL validations across various databases like Oracle and IBM Data base.

I'm familiar with Agile methodology, and I’ve closely worked with developers, BAs, and business teams to ensure delivery of high-quality products.

I’m always focused on improving quality and user satisfaction. Right now, I’m looking for an opportunity where I can contribute my skills while also learning and growing in a challenging QA environment.

**As a software test engineer, my day-to-day activities include:**

* Understanding and reviewing data mapping documents shared by the development team
* Writing and reviewing test cases
* Uploading test cases into ALM after review completion
* Executing test cases and logging defects
* Performing regression testing and retesting
* Coordinating with developers and business analysts for issue resolution
* Updating test reports and status in JIRA
* For ETL testing – writing SQL queries to validate data from source to target
* Attending daily stand-up and team meetings

**Q 1. Difference between Verification and Validation:**

**Verification:**

Process of checking whether the product is being built correctly, according to requirements and design.

|  |
| --- |
|  |

Are we building the product right?

**Type**: Static testing (no code execution)

**Example:**

Reviewing a data mapping document to ensure it matches the requirement.

**Validation:**

|  |
| --- |
|  |

Process of checking whether the right product has been built, and it meets the user's needs.

|  |
| --- |
|  |

Are we building the right product?

**Type**: Dynamic testing (code execution)

**Example**:   
Running the ETL job and checking if the data in the target matches the expected output

**Q 2. Different Levels of Testing**

**Unit Testing**

Testing individual components or modules of the application in isolation.

Done by developers to ensure each small part works as expected.

**Example**: Testing a single function that calculates interest in a banking app.

**Integration Testing**

Testing how different modules **work together** after integration.

Ensures data and control flow correctly between components.

**Example:** Testing how the “login module” connects with the “user profile” module.

**System Testing**

Testing the **entire application as a whole** against requirements.

Performed by **testers** to verify functional and non-functional aspects.

**Example:** Testing the full e-commerce site — login, search, add to cart, payment, order confirmation.

**User Acceptance Testing (UAT)**

Testing done by **end users or clients** to confirm the system meets business needs.

Usually the final phase before release.

**Example:** A business user checks if an ETL-generated sales report matches their expectations.

**Q 3. What is the difference between functional and non-functional testing?**

**Functional Testing**

Purpose: Verifies that the application’s features **work as expected** according to requirements.

Checks **what** the software should do.

Examples: Login works correctly, payment is processed.

Testing methods: Manual testing, automation of functional test cases, API testing.

**Non-Functional Testing**

Purpose: Checks **how well** the application works in terms of performance, usability, reliability, etc.

Checks **how well** the software works.

Examples: Application loads within 3 seconds, can handle 1,000 users, UI is user-friendly.

Testing methods: Load testing, stress testing, security testing, usability testing.

**Q 4. Explain the SDLC and STLC.**

**SDLC (Software Development Life Cycle):**

It’s the **process** used to plan, build, test, and deliver software in a structured way. or

SDLC is the step-by-step process for developing software from requirements gathering to maintenance.

**Phases of STLC:**

**Requirement Gathering (Requirement Analysis)**

* Understanding what the client or business needs.
* Example: Client wants an e-commerce website with payment integration.

**System** **Design**

* Creating architecture, database design, and UI mock-ups.
* Example: Designing how the website pages and database will look and work.

**Development**

* Writing code to build the application.
* Example: Developers code the product listing, cart, and payment modules.

**Testing**

* Verifying the application works as expected and is bug-free.
* Example: Testers check login, payment, and product search functions.

**Deployment**

* Releasing the application to users.
* Example: Making the website live for customers.

**Maintenance**

* Fixing issues and adding improvements after release.
* Example: Adding new features like coupons or offers.

### ****STLC (Software Testing Life Cycle)****

### STLC is the **step-by-step process followed during testing** to ensure software quality. Phases of STLC:

**Requirement Analysis**

* Understand testing requirements from the business documents (BRD, SRS, user stories).
* Identify what needs to be tested.

**Test Planning**

* Define the testing strategy, scope, tools, resources, timelines, and responsibilities.

**Test Case Design & Development**

* Write detailed test cases, test scripts, and prepare test data.

**Test Environment Setup**

* Prepare hardware, software, databases, and test environments needed to execute tests.

**Test Execution**

* Run test cases, compare actual vs expected results, and log defects in tools like JIRA or ALM.

**Defect Reporting & Tracking**

* Report bugs to developers and retest after fixes.

**Test Closure**

* Prepare final reports, summarize test coverage, defect details, and lessons learned.

**In one line for interviews:**  
"STLC is the process of planning, preparing, executing, and closing testing activities to ensure the software meets quality standards."

**Q 5. What is a test case? What are the key components of a test case?**

A **test case** is a set of conditions or steps written by a tester to check whether a specific feature or functionality of the software works as expected.

Example:  
To test the “Login” feature, a test case will describe steps like: open the login page, enter valid credentials, click login, and verify the home page is displayed.

**Key Components of a Test Case:**

**Test Case ID** – Unique identifier for the test case

**Test Case Description** – What is being tested

**Preconditions** – Conditions that must be met before executing the test

**Test Steps** – Detailed step-by-step instructions to perform the test

**Test Data** – Data required for the test

**Expected Result** – What should happen if the software works correctly

**Actual Result** – What actually happened during execution

**Status (Pass/Fail)** – Outcome of the test case

**Remarks/Comments** – Any additional notes or observations

**In one line:**  
"A test case is a written set of steps to check a specific function, including what to test, how to test it, and what result to expect."

**What is software testing and why is it important?**

Software Testing is the process of evaluating a software application to ensure that it works as expected. It involves executing the software, finding defects/bugs, and checking whether the software meets the specified requirements and delivers the expected quality.

or

Software Testing is the process of checking a software to make sure it works correctly, is free of errors, and meets the requirements of the user or client.

### Why is Software Testing Important?

### Ensures Quality

### Helps deliver reliable, user-friendly, and high-performing software.

### Detects Bugs Early

### Identifies defects at an early stage, reducing fixing costs and effort.

### Prevents Failures

### Avoids software crashes, data loss, or security issues that can harm businesses.

### Saves Time & Money

### Fixing issues after release is costly. Testing early reduces rework and expenses.

### Improves User Satisfaction

### Users expect smooth, error-free software. Testing ensures better customer experience.

### Ensures Security

### Detects vulnerabilities to protect sensitive data from hackers.

### Meets Requirements

### Confirms that the product behaves as per client/business expectations.

**Q 6. What is the difference between a test plan and a test strategy?**

**Test Plan:**

A document that describes how testing will be done for a specific project.

Prepared by: Test Lead or QA Manager

Focus: Scope, schedule, resources, and activities for that project’s testing

**Test Strategy:**

A high-level document that defines the overall testing approach for the organization or multiple projects.

Prepared by: Usually QA Manager or higher management

Focus: Overall testing approach, methods, and quality standards

**Q 7. What is a Test Plan Document?**

A Test Plan is a formal document that describes the scope, objectives, approach, resources, and schedule of testing activities for a specific project. It acts as a blueprint for how testing will be carried out.

What does a Test Plan contain? (Key Contents)

* Test Plan ID – Unique identifier
* Introduction / Objective – Purpose of testing
* Scope of Testing – What will be tested (in-scope) and what won’t (out-of-scope)
* Test Strategy / Approach – How testing will be done (manual, automation, ETL etc.)
* Test Environment – Hardware, software, database, tools needed
* Test Deliverables – Documents to be provided (test cases, test data, reports, etc.)
* Roles and Responsibilities – Who will perform which testing tasks
* Schedule / Timeline – When testing activities will happen
* Entry and Exit Criteria – When to start and stop testing
* Risks and Mitigation – Possible risks and how to handle them

**Q 8. What are severity and priority? Give examples.**

**Severity vs Priority:**

Severity – Defines how serious the defect is in terms of the system’s functionality. (Impact on the application)  
Priority – Defines how quickly the defect should be fixed. (Business urgency)

**Examples:**

**High Severity, Low Priority**

* Example: In a banking app, “Interest calculation is wrong” – serious functional issue, but if release is tomorrow and the feature is not part of the release, it may be fixed later.

**Low Severity, High Priority**

* Example: The company logo is misspelled on the homepage – doesn’t affect functionality, but must be fixed immediately because it impacts business image.

**High Severity, High Priority**

* Example: “Login button not working” – blocks all users, must be fixed urgently.

**Low Severity, Low Priority**

* Example: A minor alignment issue in a rarely used page – not urgent and doesn’t affect functionality.

💡 Easy way to remember:

* Severity = Impact on the system 🛠️
* Priority = Urgency for the business ⏳

**Q 9. What is regression testing, and when do you perform it?**

Regression testing is the process of retesting the existing functionality of an application to ensure that new changes (like bug fixes, enhancements, or new features) have not broken or impacted the old functionality.

**When do we perform it?**

After bug fixes

After new feature implementation

📌 **Example:**

If a new “Discount Coupon” feature is added to an e-commerce site, regression testing ensures older functions like login, add-to-cart, and payment are still working correctly.

💡 **One-line version for quick answer:**

"Regression testing is done after changes in the code to make sure the existing features still work as expected."

**Q 10. What is Retesting?**

Retesting is the process of testing a specific defect again after it has been fixed to confirm that the fix works correctly.

It is done on the same test cases where the defect was originally found.

📌 **Example:**

If the “Login button not working” bug was fixed by developers, testers re-run the same login test cases to confirm it now works.

**Difference between Regression Testing and Retesting**

**Purpose**:

Regression Testing: To check if new changes have affected existing functionality.

Retesting: To check if a particular defect is fixed.

💡 **Easy one-liner:**

Regression = Check old features still work after changes.

Retesting = Check the bug is really fixed.

**Q 11. Defect Life Cycle (Bug Life Cycle)**

The Defect Life Cycle is the step-by-step process a defect (bug) goes through from the time it is found until it is fixed and closed.

**Typical Stages of a Defect Life Cycle:**

* New – Tester finds a defect and logs it.
* Assigned – The defect is assigned to a developer for fixing.
* Open – Developer starts working on the defect.
* Fixed/Resolved – Developer fixes the defect and marks it as resolved.
* Retest – Tester retests the defect with the same test case.
* Closed – If the defect is working fine after retest, tester closes it.
* Reopened – If the defect is still not fixed, tester reopens it.
* Deferred/Rejected/Not a Bug – If the defect is invalid, not important right now, or as per design.

📌 **Example:**

If the “Login button” doesn’t work, the tester logs it → Developer fixes it → Tester retests → If it works, it’s **Closed**. If not, it’s **Reopened**.

**💡 One-line definition for interview:**  
"Defect Life Cycle is the journey of a bug from discovery to closure, including states like New, Open, Fixed, Retested, and Closed."

**Q 12. What is exploratory testing? When would you use it?**

Exploratory testing is unscripted testing where testers explore the application on the fly, usually used when time is short or requirements are unclear.

📌 **Example:**

If a new feature (like “Apply Discount Coupon” in an e-commerce app) is released suddenly, and you don’t have enough time to write test cases, you directly explore the feature by trying different inputs, combinations, and flows.

**When do we use Exploratory Testing?**

* When **requirements are unclear or incomplete**
* When there is **limited time** to test (short deadlines)

**Exploratory Testing vs Ad-hoc Testing**

Exploratory Testing:

* A structured approach where testers explore, learn, and design tests while executing.

Ad-hoc Testing

* An informal way of testing without planning or documentation.

**Purpose:**

Exploratory Testing: To discover defects by exploring functionality systematically.

Ad-hoc Testing: To quickly find defects without following any process.

**💡 Easy way to remember:**

* Exploratory = Structured + Experience-based
* Ad-hoc = Random + No structure

**Q 13. What are positive and negative test cases? Give examples.**

**Positive Test Cases ✅**

These are test cases written to verify the system works as expected with valid input.

Aim: To confirm the software behaves correctly for valid scenarios.

📌 **Example:**

* Login page → Enter **valid username and valid password** → User should be logged in successfully.

**Negative Test Cases** ❌

These are test cases written to check the system’s behavior with invalid input or unexpected actions.

Aim: *To ensure the software handles errors gracefully without breaking.*

📌 **Example:**

* Login page → Enter **valid username and wrong password** → System should show an error message and not allow login.

💡 **One-liner to remember:**

* Positive test case = Check expected behavior with valid input.
* Negative test case = Check error handling with invalid input.