2. Functional Testing and Test Case Design

9. **How do you ensure comprehensive test coverage?**
    - Use traceability matrices to map requirements to test cases.
10. **What types of functional testing have you performed?**
    - Regression, smoke, system integration, and UAT testing.
11. **Can you describe your approach to test case design?**
    - Analyze requirements, write positive/negative test cases, and cover edge cases.
12. **How do you handle prioritizing test cases?**
    - Focus on critical user paths, high-risk areas, and core functionalities first.
13. **What is boundary value analysis?**
    - A technique where testing is done at the boundaries of input ranges (e.g., min-1, min, max, max+1).
14. **How do you test integrations, such as SAP or Allegro systems?**
    - Validate data flows, API requests/responses, and error handling across interfaces.
15. **What techniques do you use for negative testing?**
    - Provide invalid inputs, unexpected conditions, and verify system stability.

---

3. Domain-Specific Testing

Banking

16. **What are the key test areas for a banking application?**
    - Transaction validations, security compliance (e.g., PCI DSS), and performance under load.
17. **How do you test payment gateways?**
    - Validate multiple payment methods, encryption, error handling, and integration with banks.
18. **What challenges do you face in banking domain testing?**
    - Handling sensitive data securely and ensuring compliance with regulations like GDPR.

e-Commerce

19. **What test cases are critical for e-commerce platforms?**
    - Validating the search feature, checkout workflows, inventory updates, and payment integrations.
20. **How do you handle load testing in e-commerce?**
    - Use tools like JMeter to simulate user loads and validate application performance during peak hours.

Energy and Commodities

21. **What unique tests are needed for energy applications?**
    - Validate data integration from multiple sources, real-time updates, and compliance with energy regulations.
22. **How do you test trading systems, such as Allegro?**

- Focus on trade lifecycle workflows, risk management, and synchronization with external systems like SAP.

---

4. Tools and Technology

23. **How do you use JIRA for test management?**
    - Link test cases to user stories, track defects, and monitor progress through dashboards.
24. **What is your experience with Zephyr?**
    - Managing test cases, tracking execution, and generating test reports integrated within JIRA.
25. **How do you ensure configuration management in testing?**
    - Use tools like SVN or TFS to manage test artifacts and scripts.
26. **What is your approach to defect tracking?**
    - Log detailed defect reports in tools like JIRA, including reproduction steps, severity, and priority.
27. **How do you track metrics using JIRA?**
    - Use custom dashboards for defect status, sprint progress, and test execution rates.

---

5. Leadership and Collaboration

28. **How do you manage distributed teams in Global Delivery mode?**
    - By ensuring clear communication, regular sync-ups, and effective documentation.
29. **How do you handle conflicts within a QA team?**
    - Facilitate discussions to understand root causes and align on resolutions.
30. **What strategies do you use to mentor your team?**
    - Conduct regular knowledge-sharing sessions and provide feedback for improvement.
31. **How do you handle tight deadlines?**
    - Prioritize high-risk areas and collaborate closely with stakeholders to manage expectations.
32. **How do you ensure stakeholder satisfaction?**
    - Through regular updates, incorporating feedback, and delivering defect-free features.

T-Shirt Estimation Technique for QA

Definition:
T-Shirt estimation is a relative estimation technique used in Agile projects to assess the effort or complexity of tasks by categorizing them into predefined sizes, such as Small (S), Medium (M), Large (L), and Extra Large (XL). It is a simple, intuitive, and collaborative method often used for high-level effort estimation during planning sessions.

Relative estimate is a method to estimate items by comparing how similar they are to each other in terms of complexity

---

Steps to Perform T-Shirt Estimation

1. **Define the Categories:**
    - Define the T-Shirt sizes based on effort or complexity:
        - **Small (S):** Minimal effort, simple task, or low risk.
        - **Medium (M):** Moderate effort or complexity.
        - **Large (L):** Significant effort or complexity.
        - **Extra Large (XL):** Very complex or time-consuming task.
2. **Break Down the Work:**
    - Divide the work into smaller, manageable tasks, such as test case creation, execution, automation, or regression testing.
3. **Evaluate Each Task:**
    - Assess each task based on:
        - **Complexity**: Is the task straightforward or intricate?
        - **Risk**: Are there potential challenges or unknowns?
        - **Dependencies**: Does it rely on other teams or deliverables?
        - **Effort**: How much time will it take to complete?
4. **Assign a T-Shirt Size:**
    - Collaboratively assign sizes to tasks using team consensus during planning sessions.
5. **Validate and Refine:**
    - Review estimates with stakeholders to ensure alignment and refine them if necessary.

Real-Time QA Scenarios with Examples

Scenario 1: Manual Functional Testing

- **Task:** Testing a login functionality.
- **Estimation:**
    - **Small (S):** Validating login with valid credentials.
    - **Medium (M):** Covering edge cases like invalid credentials, special characters in inputs, and locked accounts.
    - **Large (L):** Validating multi-factor authentication (MFA) or integration with third-party systems.

Scenario 2: Regression Testing

- **Task:** Performing regression testing for a release.
- **Estimation:**
    - **Small (S):** Regression of a single module with no major changes.
    - **Medium (M):** Regression of multiple modules with minor enhancements.

    - **Large (L):** End-to-end regression involving several modules and integration testing.

Scenario 3: Automation Testing

- **Task:** Automating test cases for a new feature.
- **Estimation:**
    - **Small (S):** Automating 1-2 straightforward test cases.
    - **Medium (M):** Automating 5-10 medium-complexity test cases with data-driven testing.
    - **Large (L):** Developing a robust framework for new test cases, handling complex workflows, or integrating with CI/CD pipelines.

Scenario 4: API Testing

- **Task:** Testing APIs for a payment gateway.
- **Estimation:**
    - **Small (S):** Validating GET and POST methods with simple inputs.
    - **Medium (M):** Testing with authentication tokens, headers, and multiple input combinations.
    - **Large (L):** Validating API workflows, handling multiple endpoints, and ensuring data synchronization across systems.

Scenario 5: Bug Fix Validation

- **Task:** Retesting a defect fix.
- **Estimation:**
    - **Small (S):** Retesting a UI defect fix.
    - **Medium (M):** Retesting a backend logic issue with edge cases.
    - **Large (L):** Validating a critical defect fix with cascading effects across modules.

Advantages of T-Shirt Estimation

1. **Simplicity:**
    - Easy for all team members to understand and participate in.
2. **Flexibility:**
    - Works well for early-stage planning or when precise estimates are difficult.
3. **Encourages Collaboration:**
    - Involves the entire team, ensuring diverse perspectives on task complexity.
4. **Scalability:**
    - Can be scaled to fit smaller tasks (e.g., functional testing) or larger tasks (e.g., end-to-end testing).

Challenges in T-Shirt Estimation

1. **Subjectivity:**
    - Different team members may perceive complexity differently.
2. **Not Precise:**
    - Provides a relative estimate but not exact effort in hours or days.
3. **Needs Calibration:**
    - Requires alignment among team members on what each size represents.

Best Practices for T-Shirt Estimation

1. **Collaborate:**
    - Involve QA, developers, and stakeholders to gain diverse insights.
2. **Use Historical Data:**
    - Refer to previous similar tasks to calibrate sizes.
3. **Combine with Other Techniques:**
    - Use T-Shirt estimation in conjunction with story points for detailed planning.
4. **Iterate and Refine:**
    - Revisit estimates as more details about tasks become available.

T-Shirt estimation is a powerful yet simple tool for quick and collaborative effort estimation, especially in Agile teams. It helps ensure testing efforts are well-planned and aligned with the overall sprint or project goals.

Why T-Shirt Estimation is Called a Relative Estimation Technique

T-Shirt estimation is called a **relative estimation technique** because it focuses on **comparing tasks or user stories based on their relative complexity, effort, or size**, rather than assigning absolute values like hours or days. The estimation process categorizes tasks into predefined T-Shirt sizes (e.g., Small, Medium, Large, Extra Large) to convey their relative magnitude.

Key Reasons Why T-Shirt Estimation is Relative:

1. No Fixed Time Duration

- Unlike absolute estimation, T-Shirt estimation does not assign fixed hours or days to tasks. Instead, tasks are categorized based on how they **compare** to others.
- **Example:**
  If Task A takes twice the effort of Task B, Task A might be a "Medium," while Task B is a "Small." The exact time isn't specified, but their relative effort is clear.

2. Simplifies Comparison

- Teams compare tasks against each other rather than trying to calculate exact effort.

- **Example:**
  "Is Task C more complex than Task A? If so, it should be a larger size category."

3. Focus on Effort and Complexity

- It uses **effort** (time, resources) and **complexity** (dependencies, risks) as the basis for comparison.
- **Example:**
  A task with many edge cases or integrations may be categorized as "Large" compared to a straightforward task, which might be "Small."

4. Flexible Across Teams

- The meaning of "Small," "Medium," or "Large" is relative and specific to the team's experience and context.
- **Example:**
  What one team considers "Medium" might be "Small" for another, based on their skillsets or tools.

5. Iterative Refinement

- As the team progresses, they refine these relative estimates based on the actual complexity encountered during implementation.