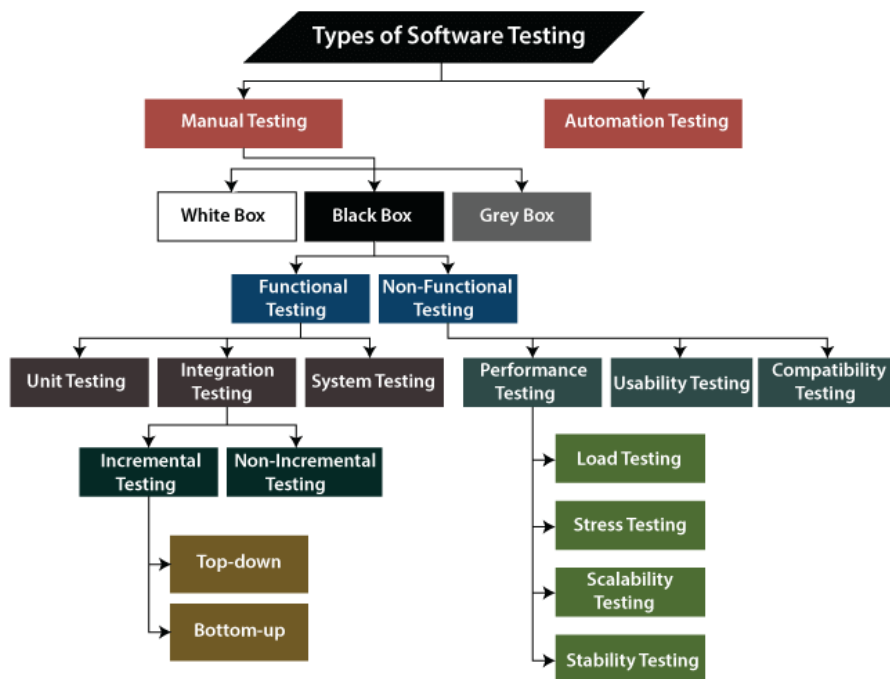
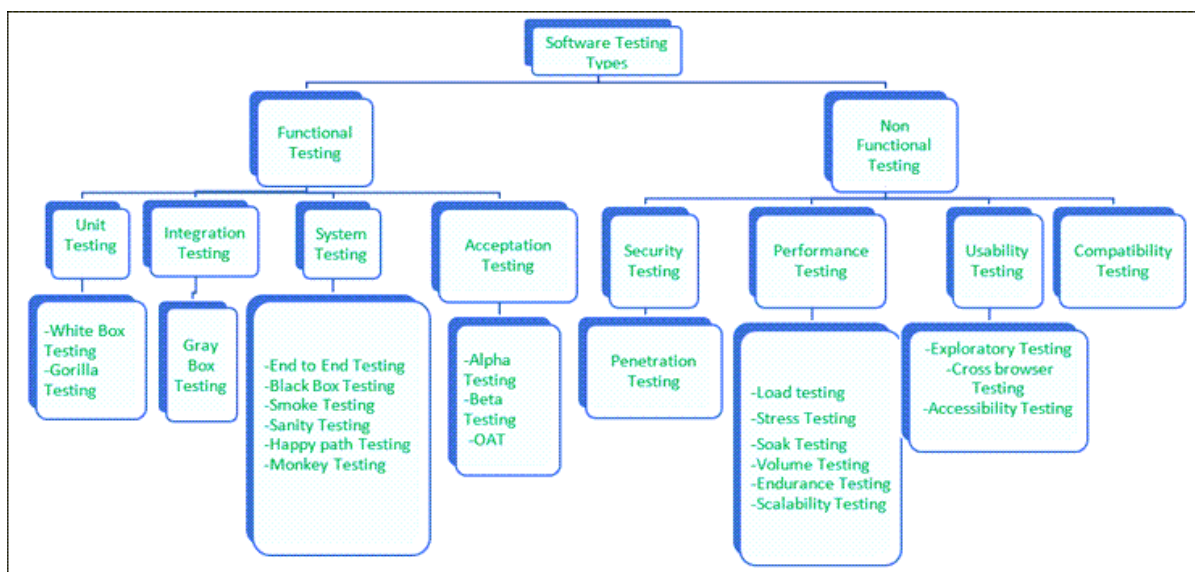
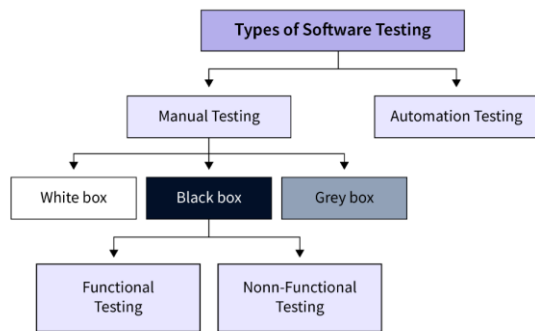


Types of Software Testing: Different Testing Types with Details



Testing is performed in the below order:

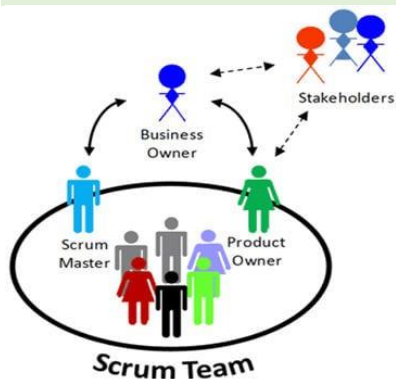


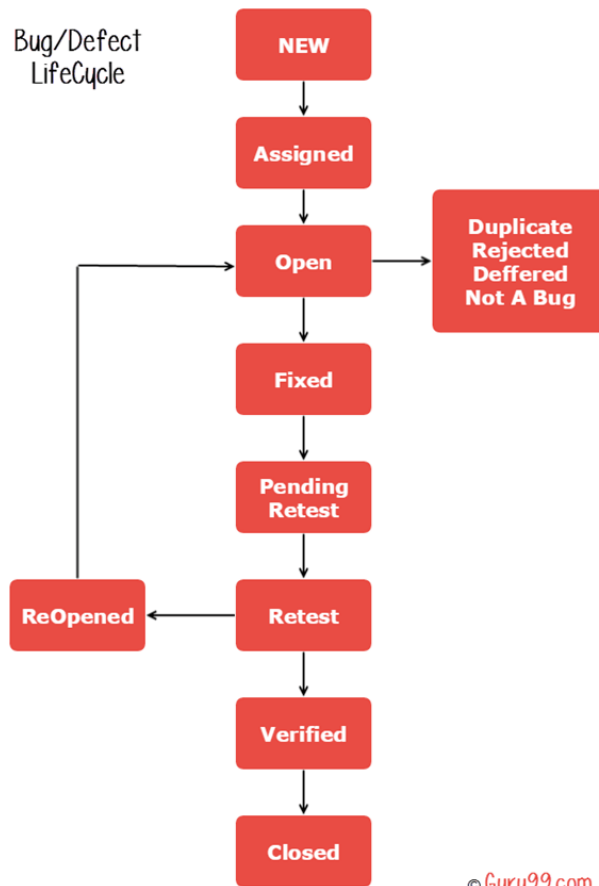
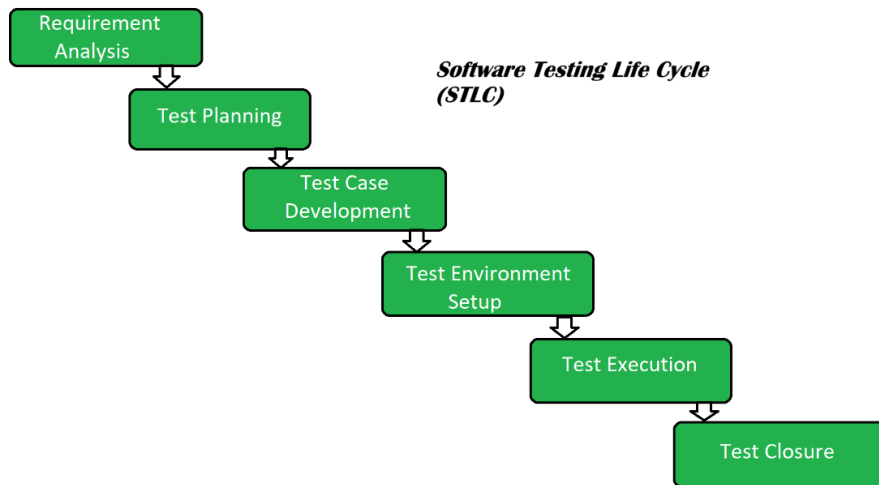
Integrated system testing

SIT Techniques

Mainly, there are 4 approaches for doing SIT:

1. Top-Down Approach
2. Bottom-up Approach
3. Sandwich Approach
4. Big Bang Approach





What is Agile?

- ✓ It's the **way of thinking** used for project management in various organizations
- ✓ This method **divides all the tasks into short phases** called iterations
- ✓ Agile means **being focused on delivering value** to the customer faster



4 Core Principles of Agile Methodology

- 1 Individuals and interactions over processes and tools
- 2 Working software over comprehensive documentation
- 3 Customer collaboration over contract negotiation
- 4 Responding to change over following a plan



Types of Agile Methodologies

- **Scrum**
Uses time-based iterations called sprints to deliver working software at regular intervals.
- **Kanban**
Visualizes work in progress through a board with columns representing different stages of work.
- **Lean**
Focuses on reducing waste and maximizing efficiency through continuous improvement.
- **Extreme Programming**
Emphasizes pair programming and code review to improve code quality.
- **Crystal**
Offers a flexible approach that can be tailored to the specific needs of a project or team.



What is Scrum?

Scrum is an Agile framework for managing complex software project work.



Scrum is a project management framework that helps teams work together to achieve a common goal:



What it is

Scrum is a set of practices, tools, and roles that help teams structure and manage their work. It's often used in software development, but its principles can be applied to any kind of teamwork.

How it works

Scrum encourages teams to:

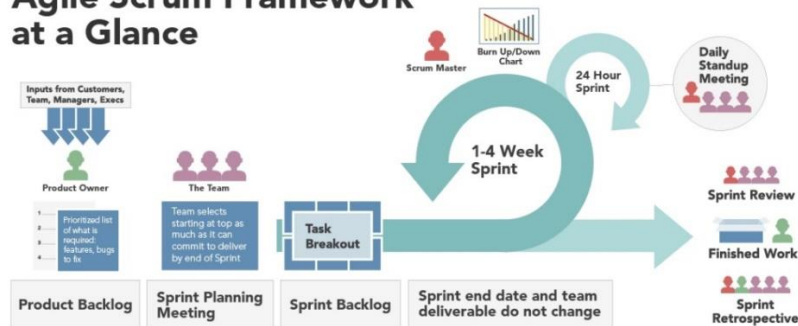
- Self-organize and learn from experience
- Adapt to change
- Reflect on their wins and losses
- Inspect and adapt at regular intervals
- Work in an open environment



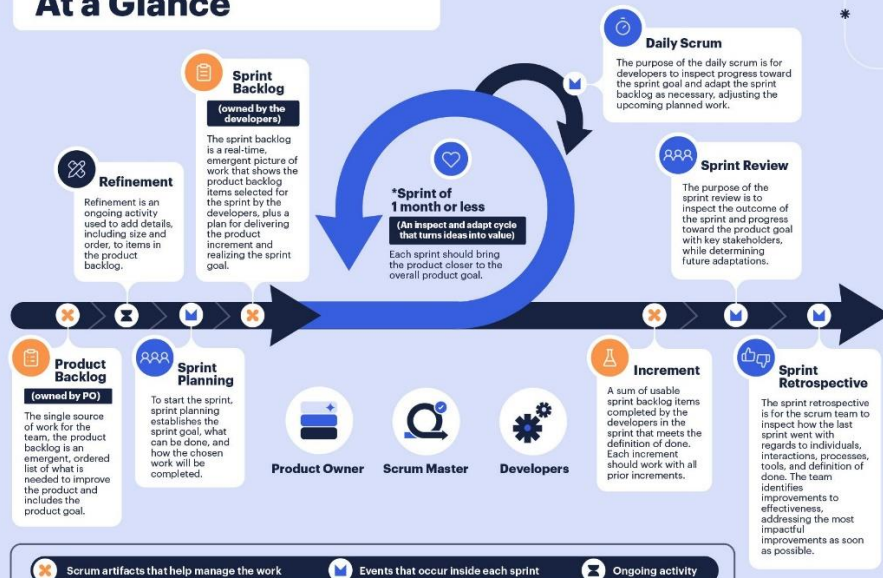
Difference Between Agile and Scrum:

Parameter	Agile	Scrum
Methodology	Agile is an approach to project management.	It is a framework of Agile methodology.
Prioritisation	In Agile, prioritisation means deciding the order in which the Agile team will work.	In Scrum, prioritisation takes place based on value.
Alternative	Waterfall is a good alternative to Agile.	Kanban is a good alternative to Scrum.
Delivery	Agile ensures continuous delivery.	Delivery is made after the completion of each sprint.

Agile Scrum Framework at a Glance



The Scrum Framework At a Glance



Who should participate and for how long?

SPRINT PLANNING

Who: The entire scrum team.
Timebox: Maximum of 8 hours for a month-long sprint. Shorter timebox for shorter sprints.

DAILY SCRUM

Who: The developers, but if the product owner or scrum master are actively working on items in the sprint backlog, they participate as developers.
Timebox: 15 minutes.

SPRINT REVIEW

Who: The entire scrum team. Stakeholders are invited to provide feedback on the increment.
Timebox: Maximum of 4 hours for a month-long sprint. Shorter timebox for shorter sprints.

SPRINT RETROSPECTIVE

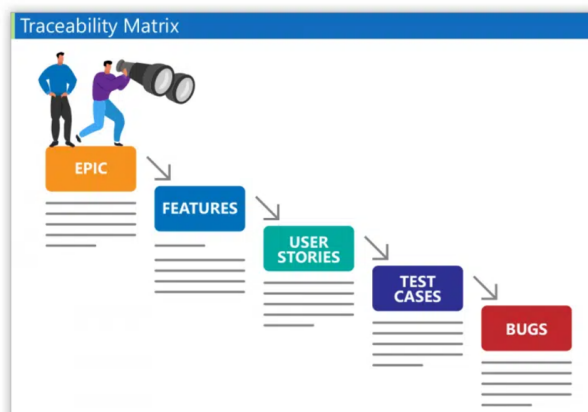
Who: The entire scrum team.
Timebox: Maximum of 3 hours meeting for a month-long sprint. Shorter timebox for shorter sprints.

Scrum Alliance®

Copyright © 2024 Scrum Alliance®, Inc. All Rights Reserved.

	A	B	C	D	E	F	G	H	I	J
1	Test Requirements Traceability Matrix (TRTM)									
2					Create Date					
3	Project Name				Last Update On					
4	Project ID				Client Name					
5	Project Manager									
6	Project Type									
7	S. no.	Req. ID	Req Desc	Source Of Requirement	Scenario	TC Id	TC Desc	Test Result	Defect Id	Defect Statu
8	1	BR 1.0	1. INBOX	New Requirement	1. Mails	TC_gmail_mail_01	Check the mail in inbox	Pass		
9				TC_gmail_mail_02		Verify the open mail	Fail	Def_01	Open	
10				TC_gmail_mail_03		Verify the unread mail	Pass			
11				TC_gmail_mail_04		Verify the Downloads and attachments	Fail	Def_02	Open	
12				TC_gmail_mail_05		Verify to navigate to different pages	Pass			
13				2. Reply	TC_gmail_Rply_01	Verify to reply a mail	Pass			
14					TC_gmail_Rply_02	Verify to reply all mails	Fail	Def_03	Closed	
15				3. FWD	*****	*****	*****			
16					*****	*****	*****			
17					*****	*****	*****			
18	2	BR 1.1	2. Sent Email	Change Request						

1	Sample Requirements Traceability Matrix									
2	BRD- Sectio	FSD- Section	Test scenario ID				Test case ID	Statu	Defects	
3	1- Loan Process	1.1- New users	TS_Loan_001- Validate the "Apply Loan" feature as a new user				TC_newuser_01	Passed		
4							TC_newuser_02	Passed		
5							TC_newuser_03	Failed	Defect_01, Defect_02	
6			TS_Loan_002- validate the "Apply Loan" feature as a already existing user				TC_newuser_04	Passed		
7							TC_newuser_05	Blocked	Defect_01	
8							TC_newuser_06	Failed	Defect_03	
9							TC_newuser_07	Passed		
10			TS_Loan_003- For a new user in the "Apply loan", check the guest customer option and apply loan				TC_newuser_08	Passed		
11			TS_Loan_004 - For a new user in the "Apply loan", check the Register option and apply loan				TC_newuser_09	Passed		
12		1.2- Existing users	TS_Loan_005- Login to the loan portal as an already a customer with a loan and check the information displayed				TC_Exist_User_01	Passed		
13			TS_Loan_006-Check the Loan whose status is "Sent for review"				TC_Exist_User_02	Passed		
14			TS_Loan_007-Check the Loan whose status is "Reviewed and Accepted"				TC_Exist_User_03	Passed		
15			TS_Loan_008-Check the Loan whose status is "Reviewed and deleted"				TC_Exist_User_04	Passed		
16										
17	2- Ease of use	2.1- Ease os use	TS_Loan_009-Check for a visitor if the information on the site is accessible in less than 3 clicks or not				TC_EasyUse_01	Passed		

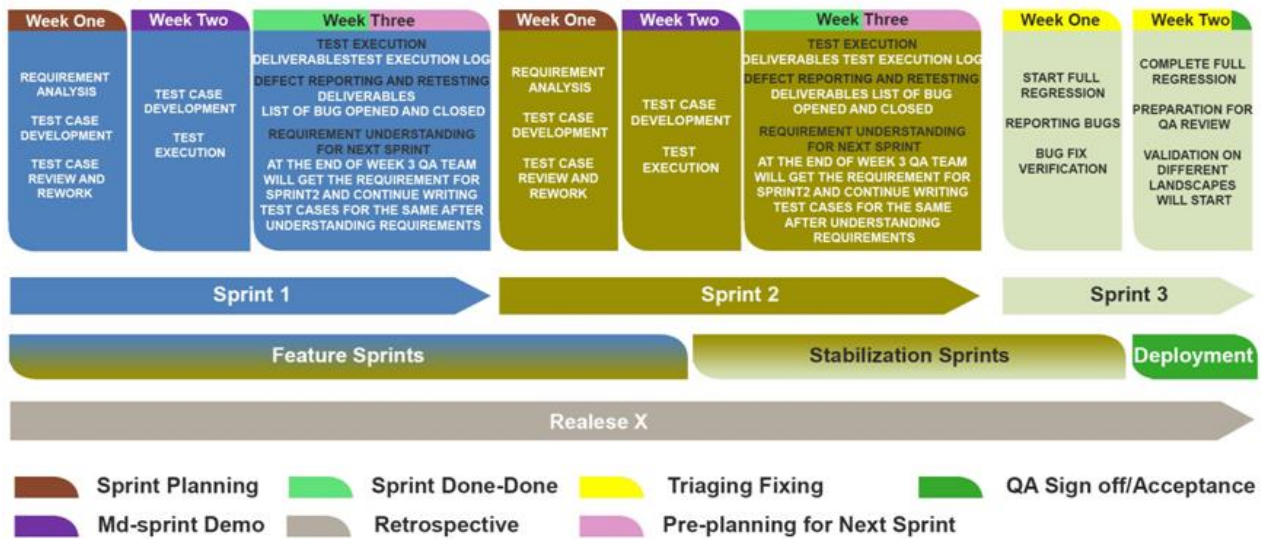


What is a Requirements Traceability Matrix (RTM)?

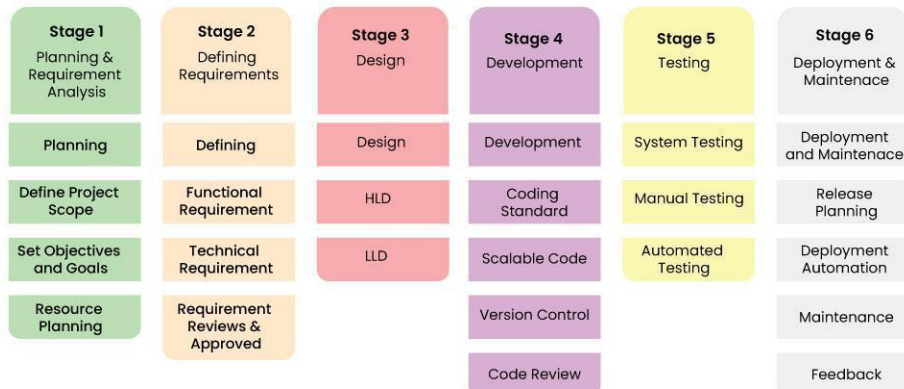
A Requirements Traceability Matrix is a tool that provides teams with the ability to easily trace requirements from end-to-end.

This means you are able to trace how higher level requirements (like Epics) trace all the way down to your lowest level requirements (like bugs). Teams can use our two matrices covered in these tutorials to build end-end traceability and also to create Test Case coverage, discover orphaned requirements, and manage the relationships between requirements!

Agile Scrum Process



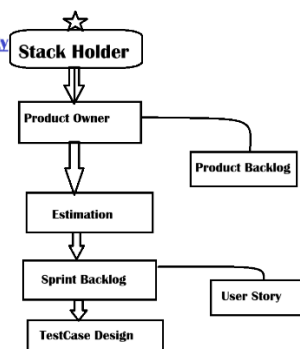
Software Development Life Cycle (SDLC)



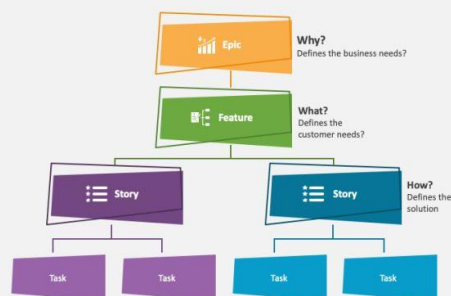
6 Stages of Software Development Life Cycle

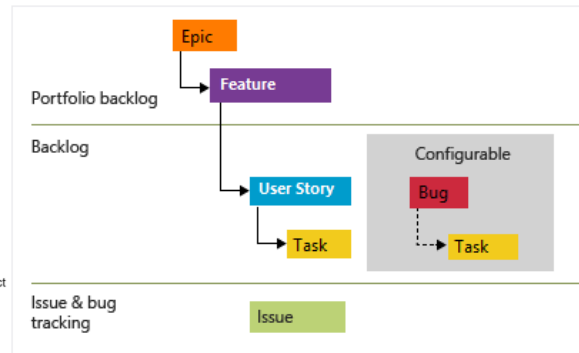
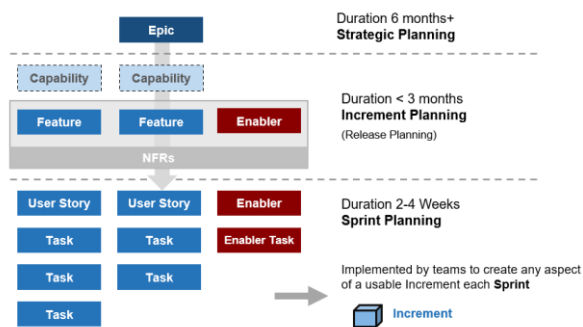


Architecture of agile Methodology



EPIC FEATURES AND USER STORIES





Sample User Story

1. Title

118 4. Case transfer

2. State

State: In Analysis
Area: Agile Dynamics 365 Template
Reason: Moved to stat...
Iteration: Agile Dynamics 365 Template\Iteration 2

3. Description

Description
As a case manager, I want to transfer the case to another team, so that other teams can continue to work on the case record.

4. Acceptance Criteria

Acceptance Criteria
Scenario 1: Transfer to Another campus
Given a case record is assigned to my team
When we transfer the case record to a team of another campus
Then the destination team gets read and update rights to the case record and gets read and update rights to the case's related records, and the originating team must keep a copy (snapshot) of the current case record, including related (or child) records, for reference and audit purposes.
Scenario 2: Transfer to Team within the same campus
Given a case record is assigned to my team
When we transfer the case record to a team of my campus
Then the destination team gets read and update rights to the case record and gets read and update rights to the case's related records, and the originating team doesn't need a copy (snapshot) of the case record and related (or child) records.

5. Discussion

Discussion
@Anne Kahil @Ben Questions for the workshop

- Why are we talking about making a copy? Can you explain the current process?
- Will teams from different campuses not have access to D365?
- Can teams from different campuses see all student records? Are there any confidential data we need to hide?

6. Dev effort

Planning
Story Points: 5
Priority: 2

7. Priority

Documents
SharePoint Location

8. Additional documentation

Related Work
+ Add link
Parent: 62 Case Management
Updated 27/11/2021. Active

9. Parent Features or Epics

Related (2)
92 1. Case statuses & stages
Updated Wednesday. New
72 2. Case assignment and routing
Updated Wednesday. New

10. Related User Stories

Key Points of GUI Functional Testing:

Key Point	Description
1. Button Functionality	Verifies that all buttons (e.g., Submit, Cancel, Reset) perform the intended actions when clicked.

2. Link Navigation	Ensures that links (hyperlinks, navigation links) open the correct pages or trigger the correct actions.
3. Input Field Validation	Checks that all input fields (text boxes, drop-downs, checkboxes) accept the correct input and validate user data.
4. Form Submission	Tests that form fields are correctly submitted and that the data entered is properly processed and stored.
5. Field Error Handling	Verifies that the system shows appropriate error messages for invalid input, and prevents submission of incorrect data.
6. Drop-Down Menus	Ensures that drop-down menus (select boxes) are working as expected, displaying correct options and accepting valid choices.
7. UI Workflow	Validates that the user can navigate through the application as intended, without errors or unexpected behavior.
8. Text and Labels	Checks that all UI text (labels, headers, buttons) is clear, concise, and matches the specifications.
9. Data Display	Ensures that data displayed on the UI (tables, lists, forms) is correct and updates as expected.
10. Modal Dialogs and Popups	Verifies that modals, alerts, popups, and confirmation dialogs appear and function correctly when triggered.
11. Date/Time Picker Functionality	Ensures that date/time picker controls are working correctly, allowing valid selections and handling edge cases.
12. File Upload/Download	Tests the file upload and download functionality, ensuring files are processed and displayed correctly.
13. Localization/Language Support	Verifies that the application works correctly across different languages and regional settings (if applicable).
14. Radio Buttons and Checkboxes	Ensures that radio buttons and checkboxes are functioning, including mutual exclusivity for radio buttons.
15. Tab Navigation	Ensures that tabbed interfaces are functioning correctly, and keyboard navigation is supported (Tab key).



Here's a clear comparison between Sanity Testing and Smoke Testing :		
Aspect	Smoke Testing	Sanity Testing
Purpose	Verifies the basic functionality of the application after a new build or major changes.	Ensures that specific functionalities work as expected after changes or bug fixes.
Scope	Broad and shallow testing, covering the application's critical paths.	Narrow and focused testing, validating specific features or areas impacted by recent changes.
Test Depth	Shallow, does not go into detail, just confirms core features work.	Deeper testing of specific functionalities or bug fixes.
Test Focus	Focuses on whether the application is stable enough to proceed with further testing.	Focuses on verifying the correctness of specific features or bug fixes.
Frequency	Performed on every new build or major release.	Performed after fixes, updates, or after smoke testing passes.
Test Execution Time	Quick, often takes less time as it covers only the most critical functionalities.	Quick, but might take slightly longer than smoke testing, as it is more focused on specific areas.
Outcome of Failure	If it fails, the build is rejected and not further tested.	If it fails, the feature or fix is rejected, and further testing cannot proceed.
Test Type	It's a shallow, high-level test that confirms basic functionality.	It's a deep, focused test on specific areas or features.
When Performed	Performed first, usually after a new build or release is deployed.	Performed after smoke testing, often after bug fixes or minor updates.
Automated or Manual	Can be automated due to its broad scope and basic checks.	Typically manual, as it focuses on specific features or fixes.



Aspect	Smoke Testing	Sanity Testing
Purpose	Ensures build stability.	Ensures correctness of specific changes.
Focus	Core functionality of the system.	Targeted areas impacted by changes.
Scope	Broad and shallow.	Narrow and deep.
Timing	Performed on every new build.	Performed after changes or fixes.
Automation	Often automated.	Typically manual.
Goal	Accept or reject the build.	Validate specific updates or fixes.

	A	B
1	SMOKE TESTING	SANITY TESTING
2	Smoke testing is nothing but we are going to check whether the build is ready for testing or not.	Sanity testing is subset of regression testing in that we are going to focus is on particular part of our application.
3	Perform by dev and test	Perform by tester
4	Its part of build acceptance testing	its part of regression testing
5	Usually Documented	Usually not documented
6	Check stability	focus on particular functionality
7		

Here are the 10 key points of Regression Testing , with unnecessary details removed		
#	Key Point	Details
1	Purpose	Ensures new code changes do not affect existing functionality.
2	When to Perform	After bug fixes, new features, or before major releases.
3	Scope	Retest impacted modules, interconnected functionalities, and core features.
4	Test Case Selection	Prioritize high-risk areas, recently modified code, and critical functionalities.
5	Automation	Automate stable, repetitive test cases to save time and resources.
6	Continuous Testing	Integrate into CI/CD pipelines for early detection of issues.
7	Risk-Based Approach	Focus on high-risk or frequently modified areas to optimize testing effort.
8	Challenges	Handling test maintenance, flaky tests, and time constraints.
9	Test Suite Maintenance	Regularly update test cases to reflect changes and remove obsolete ones.
10	Monitoring and Reporting	Continuously monitor test results and report trends, failures, or bottlenecks.

Here's the revised version of the Webpage Testing table without the Tools column:		
Testing Type	Purpose	Key Points
Tab Validation	Ensure the proper functioning of tabs within the webpage.	- Verify correct tab selection and switching.
		- Test active/inactive states.
		- Ensure the tab content loads correctly.
		- Verify tab functionality across devices and browsers.
		- Check for overflow handling when there are too many tabs.

		<ul style="list-style-type: none">- Ensure accessibility with keyboard navigation (Tab key).- Test for smooth transition between tabs (no flickering).
Link Validation	Ensure all hyperlinks (internal and external) are working properly.	<ul style="list-style-type: none">- Check for broken links (404 errors).- Ensure correct redirection.- Test anchor links.- Verify secure protocol (HTTPS) for links.- Ensure links open in the correct window/tab (if specified).- Validate links with special characters or query strings.- Check for external links to ensure they load correctly.
		<ul style="list-style-type: none">- Verify content is loading as expected.- Test page rendering across devices and browsers.- Validate page performance (load time).- Check SEO elements (meta tags, headers).- Verify images and multimedia (videos, audio) load correctly.- Ensure no content is cut off or hidden, especially on responsive views.- Validate forms and interactive elements on the page.- Test for proper error handling (e.g., form submission errors).
		<ul style="list-style-type: none">- Verify UI consistency with design.- Test responsiveness (layout adapts to different screen sizes).- Ensure interactive elements (buttons, sliders, etc.) work.- Test accessibility features (focus, ARIA).- Check for proper font sizes, colors, and alignment.- Verify visual appearance on different screen resolutions and orientations.



Key Components of a Test Plan	
Section	Description
1. Test Plan Identifier	A unique identifier for the test plan document. This helps in version control and tracking.

2. Introduction	Overview of the project, including its goals and objectives, and a summary of what the test plan covers.
3. Test Objectives	Clear goals of what the testing aims to achieve, such as verifying functionality, performance, security, etc.
4. Test Scope	Defines the boundaries of testing—what will and will not be tested, and any exclusions or limitations.
5. Test Strategy	The approach or methodology that will be used to conduct testing. This could be functional, performance, security testing, etc.
6. Test Criteria	Specifies the entry and exit criteria for testing. These criteria define when testing can begin and when it is considered complete.
7. Test Environment	Describes the hardware, software, network configurations, and other tools needed for testing.
8. Test Deliverables	Lists the deliverables for the testing process, such as test cases, test scripts, defect reports, and final test results.
9. Test Schedule	The timeline for testing, including milestones and deadlines for each testing phase (e.g., unit testing, integration testing).
10. Resource Requirements	Specifies the required personnel, tools, and infrastructure for testing, including the roles and responsibilities of the testing team.
11. Test Types	Specifies the types of testing to be performed, such as functional, regression, smoke, sanity, security, load, etc.
12. Test Approach	Describes the specific techniques and methods to be used in testing, including manual testing or automated testing strategies.
13. Risk Assessment	Identifies potential risks that could affect the testing process (e.g., resource limitations, tight deadlines) and how to mitigate them.
14. Defect Management	Describes how defects will be logged, tracked, and resolved, including the severity levels and priority for fixing them.
15. Test Suspension Criteria	Defines when testing should be paused or halted due to issues such as critical defects or resource unavailability.

16. Approval and Sign-Off

The process for getting the test plan approved by stakeholders or management and any necessary sign-offs for each phase.

Test Scenario vs Test Case:

Aspect	Test Scenario	Test Case
Definition	A high-level description of a feature or functionality to be tested.	A detailed step-by-step guide to test a specific condition.
Scope	Broad; defines what to test.	Specific; defines how to test.
Purpose	Ensures coverage of key functionalities.	Verifies detailed behavior of the system.
Level of Detail	High-level, usually in simple language.	Detailed, including inputs, expected results, and execution steps.
Example	Verify login functionality.	Verify that the user can log in with a valid username and password.

A typical **test case** includes the following components:

Test Case Component	Description
Test Case ID	A unique identifier for the test case (e.g., TC001, TC002).
Test Case Name	A brief name or description of the test case.
Test Objective/Description	A brief description of what the test case is intended to validate (e.g., verify login functionality).
Pre-Conditions	The necessary setup or conditions required before executing the test (e.g., user must be registered).
Test Steps	A sequence of actions or steps to be followed during the execution of the test case.
Test Data	The input data to be used in the test (e.g., username, password, etc.).

Expected Result	The expected behavior or output after executing the test steps.
Actual Result	The actual behavior or output observed during the test execution.
Status (Pass/Fail)	The result of the test case based on the comparison of expected and actual results.
Priority	The importance or priority of the test case (e.g., High, Medium, Low).
Test Case Type	The type of test (e.g., Functional, Regression, Usability).
Assigned To	The tester responsible for executing the test case.
Comments/Notes	Additional comments or notes relevant to the test case (e.g., known issues, assumptions).
Example Test Case	
Test Case Component	Example
Test Case ID	TC_001
Test Case Name	Verify login functionality with valid credentials.
Test Objective/Description	To verify that a user can log in successfully with valid credentials (username and password).
Pre-Conditions	User should be registered in the system.
Test Steps	1. Open the application login page.
	2. Enter valid username and password.
	3. Click the login button.
Test Data	Username: testuser
	Password: Test123!
Expected Result	User should be successfully logged in and redirected to the dashboard page.
Actual Result	User is logged in and redirected to the dashboard.
Status (Pass/Fail)	Pass
Priority	High
Test Case Type	Functional
Assigned To	Tester A
Comments/Notes	N/A

Test Scenario 1:	Verify User Login Functionality			
-------------------------	--	--	--	--

Test Case ID	Test Case Description	Test Steps	Expected Result	Priority
TC_001	Verify login with valid username and password.	1. Open the login page.	User should be successfully logged in and redirected to the dashboard.	High
		2. Enter valid username and valid password.		
		3. Click the "Login" button.		
TC_002	Verify login with invalid username and valid password.	1. Open the login page.	Error message: "Invalid credentials" should be displayed.	High
		2. Enter invalid username and valid password.		
		3. Click the "Login" button.		
TC_003	Verify login with valid username and invalid password.	1. Open the login page.	Error message: "Invalid credentials" should be displayed.	High
		2. Enter valid username and invalid password.		
		3. Click the "Login" button.		
TC_004	Verify login with both invalid username and password.	1. Open the login page.	Error message: "Invalid credentials" should be displayed.	High
		2. Enter invalid username and invalid password.		
		3. Click the "Login" button.		
TC_005	Verify the "Forgot Password" link works during login.	1. Open the login page.	User should receive a password reset link via email.	Medium
		2. Click on "Forgot Password" link.		
		3. Enter registered email address.		
Test Scenario 2:	Verify User Registration			
Test Case ID	Test Case Description	Test Steps	Expected Result	Priority
TC_006		1. Open the registration page.	User should be registered	High

	Verify successful user registration with valid data.	2. Enter valid username, email, and password. 3. Click on "Register".	successfully and redirected to the login page.	
TC_007	Verify user registration with missing required fields.	1. Open the registration page. 2. Leave one or more required fields empty. 3. Click on "Register".	Error message: "All fields are required" should be displayed.	High
TC_008	Verify user registration with invalid email format.	1. Open the registration page. 2. Enter an invalid email (e.g., "user@domain"). 3. Click on "Register".	Error message: "Invalid email format" should be displayed.	Medium
TC_009	Verify user registration with existing username.	1. Open the registration page. 2. Enter a username already used by another user. 3. Click on "Register".	Error message: "Username already exists" should be displayed.	High
TC_010	Verify the password strength requirement during registration.	1. Open the registration page. 2. Enter a password that doesn't meet strength requirements (e.g., too short). 3. Click on "Register".	Error message: "Password must be at least 8 characters long" should be displayed.	High
Test Scenario 3:	Verify Search Functionality			
Test Case ID	Test Case Description	Test Steps	Expected Result	Priority
TC_011	Verify search results are displayed for a valid search term.	1. Enter a valid search term (e.g., "Laptop") in the search bar. 2. Click the "Search" button.	Relevant search results should be displayed.	High

TC_012	Verify no results message for invalid search term.	1. Enter an invalid search term (e.g., "xyz123") in the search bar. 2. Click the "Search" button.	Message: "No results found" should be displayed.	Medium
TC_013	Verify search results can be sorted by price (low to high).	1. Enter a valid search term in the search bar. 2. Apply "Sort by Price: Low to High".	Search results should be sorted by price in ascending order.	Medium
TC_014	Verify search results can be filtered by category.	1. Enter a valid search term in the search bar. 2. Apply category filter (e.g., "Electronics").	Search results should display only products from the selected category.	Medium
TC_015	Verify the "Clear Search" functionality.	1. Enter a search term. 2. Click on the "Clear Search" button.	The search bar should be cleared and the results reset.	Low
Test Scenario 4:	Verify Add to Cart Functionality			
Test Case ID	Test Case Description	Test Steps	Expected Result	Priority
TC_016	Verify adding a product to the cart.	1. Browse through available products. 2. Select a product and click "Add to Cart".	The selected product should be added to the cart.	High
TC_017	Verify cart updates with multiple products.	1. Add multiple products to the cart. 2. Check cart icon for updated count.	The cart should display the correct number of products.	High
TC_018	Verify adding out-of-stock product to the cart.	1. Select an out-of-stock product. 2. Try to add it to the cart.	A message "Out of Stock" should appear and the product should not be added to the cart.	Medium
TC_019	Verify cart displays correct price and quantity for each item.	1. Add multiple items to the cart. 2. Verify item prices and quantities.	The cart should display the correct total price for each item.	High
TC_020	Verify removal of a product from the cart.	1. Add a product to the cart.	The product should be removed from	Medium

		2. Click the "Remove" button next to the product.	the cart and the cart should update accordingly.	
Test Scenario 5:	Verify Logout Functionality			
Test Case ID	Test Case Description	Test Steps	Expected Result	Priority
TC_021	Verify user can log out successfully.	1. Log in to the application. 2. Click on the "Logout" button.	The user should be logged out and redirected to the login page.	High
TC_022	Verify that after logout, the user is redirected to the login page.	1. Log out of the application. 2. Check if the user is redirected to the login page.	The user should be redirected to the login page.	High
TC_023	Verify session expiration after logout.	1. Log out of the application. 2. Try to access a restricted page.	User should be redirected to the login page with an error message.	Medium

--

Key Deliverables in Test Closure	
Deliverable	Description
Test Closure Report	A summary of the testing effort, including test execution status and defect information.
Defect Logs	Final list of defects, including their status (open/closed/deferred).
Test Artifacts	All test-related documents (test cases, scripts, data, results).
Test Metrics	Metrics and analysis regarding test effectiveness and coverage.
Lessons Learned Document	A document capturing insights for process improvement.
Sign-off	Formal approval from stakeholders indicating completion of testing.

Release Notes	Details for end users about known issues, limitations, and fixes.
Test Environment Cleanup	Confirmation of environment reset for future use.

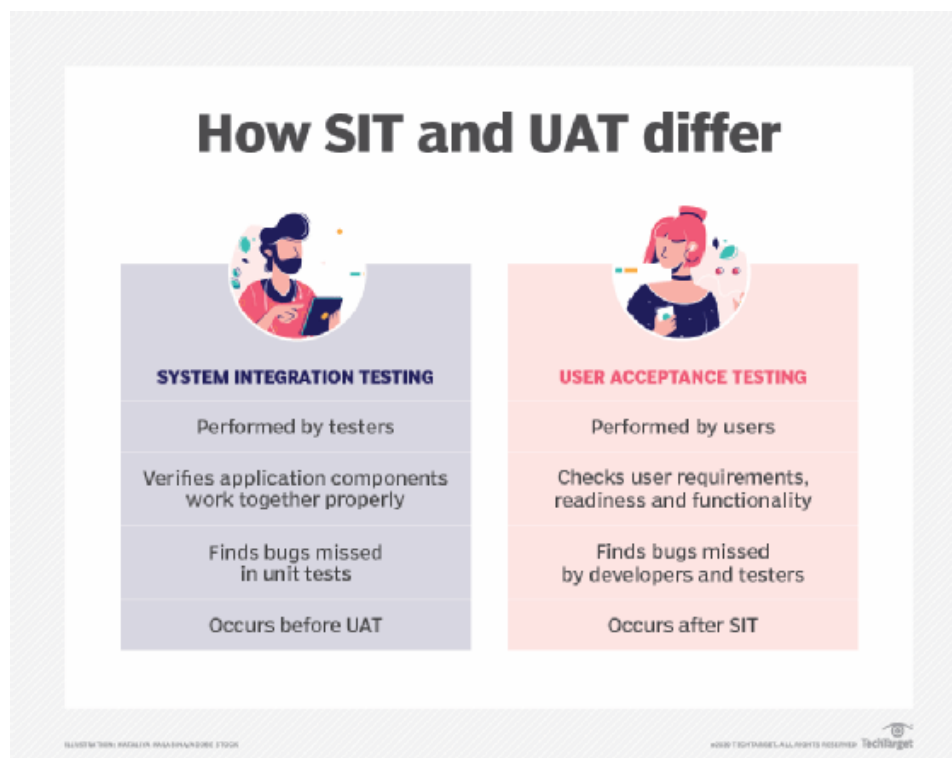
Key Steps in a Build Process:	
Step	Description
Source Code Management	Developers write and commit code to a version control system (e.g., Git, SVN). This is the starting point of the build process.
Compilation	The source code is compiled into machine-readable code or an executable (e.g., .exe, .apk, .jar, etc.).
Unit Tests	Automated unit tests are often run to verify that individual parts of the code (e.g., functions or methods) work as expected.
Linking	The compiled code is linked with external libraries, resources, and dependencies to form a complete application.
Packaging	The build is packaged into a deployable unit (e.g., a .zip file, .apk for Android, .exe for Windows).
Deployment	The built and tested version is deployed to a staging or production environment for further validation or release.

Major QA Points in the Build Process	
Key Point	Description
Build Verification Testing (BVT) / Smoke Testing	QA performs Smoke Testing on each new build to ensure core functionalities are working, validating that the build is stable enough for further testing.
Regression Testing	QA runs regression tests on every build to ensure new changes haven't broken existing functionality.
Automated Testing in Build Pipeline	QA integrates automated tests in the build pipeline, enabling quick feedback on each new build for faster defect detection.

Cross-platform Testing

QA ensures the build works across all intended platforms (e.g., web, mobile) to maintain consistency in functionality and performance.

Level of Functional Testing	Key Point
Unit Testing	Verifies individual components or functions to ensure correct behavior.
Integration Testing	Checks interactions between components or systems to ensure they work together as expected.
System Testing	Validates the complete system's functionality to ensure it meets the specified requirements.
User Acceptance Testing (UAT)	Performed by end-users to verify the system meets business needs and is ready for release.



Key Points of SIT Testing:

Key Point	Description
Purpose	Validates that different subsystems or external systems interact correctly with each other.
Scope	Covers end-to-end interactions between various systems and their integrations, including third-party systems.
Focus Areas	Focuses on verifying communication, data exchange, protocols, and interactions between integrated systems.
Performed After	SIT is typically performed after Unit Testing and Integration Testing but before System Testing .
Test Environment	Requires a realistic environment where all systems and components to be integrated are available and can interact.
Test Types	Includes testing of interfaces, data flow, network communication, error handling, and overall system interactions.
Test Cases	Test cases in SIT are designed based on the interactions between systems, focusing on data integrity, consistency, and communication.
Tools	SIT can be done manually or with the help of automated testing tools, depending on the complexity of integration points.
Defects	Identifying integration issues such as incorrect data mapping, protocol mismatches, and improper error handling between systems.
End Goal	Ensures that all systems work as expected when integrated, ensuring the end-to-end functionality of the application or system.



Key Points of UAT Testing:	
Key Point	Description
Purpose	To validate that the software meets the business requirements and is ready for production deployment.
Who Performs UAT	Performed by end-users or stakeholders (typically non-technical users) to confirm the software's functionality.

When it Occurs	UAT occurs after System Testing and before the system is deployed to production. It's typically the last phase of testing.
Focus	Focuses on business requirements and user experience rather than technical aspects of the system.
Test Environment	Conducted in a staging or pre-production environment that mirrors the actual production environment.
Test Scenarios	UAT test cases are based on real-world scenarios , derived from business processes, user needs, and requirements.
Types of UAT	- Alpha Testing : Conducted by internal users within the organization.
	- Beta Testing : Conducted by actual users or a small group of external customers.
Success Criteria	The system passes UAT if the users confirm that the software meets the business requirements and functions as expected in real-world usage.
Tools	UAT may be done manually or with the aid of automated tools, but it is often manual testing due to the business-oriented focus.



key points for Regression Testing:	
Key Point	Description
Purpose	Ensures new changes (bug fixes, new features, updates) do not break existing functionality.
After Bug Fixes	Performed after defects are fixed to ensure the fix doesn't cause other issues.
After New Features	Validates that newly added features don't disrupt existing parts of the system.
After Software Updates	Ensures that software updates or patches do not negatively affect the overall system.
After System Maintenance	Verifies that maintenance tasks (e.g., code refactoring) haven't caused new issues.
During Continuous Integration	Performed with each build or code integration in continuous development processes.
After Performance Improvements	Confirms that performance optimizations don't impact existing functionality.
After Platform/Environment Changes	Ensures that the software works correctly when moved to a different environment.

After UAT Feedback	Validates that changes made based on user feedback don't introduce new defects.
--------------------	---

key points for when testing starts in software development:	
Key Point	Description
During Requirement Gathering	Testing starts early by reviewing and validating requirements to ensure they are clear and testable.
During Development (Unit Testing)	Testing begins as developers write code, with unit testing to verify individual units or components.
After Code Completion (Integration Testing)	Once components are integrated, integration testing starts to check if they work together correctly.
After the Build is Ready (Smoke Testing)	Smoke testing is done to verify basic functionality and stability before moving to detailed testing.
When the System is Stable (System Testing)	System testing begins to validate the overall application and ensure it meets the specified requirements.
After User Feedback (UAT)	User Acceptance Testing (UAT) starts once the system is ready, to ensure it meets the end-user needs.
After New Changes (Regression Testing)	Regression testing starts whenever there are code changes to ensure nothing else is broken.
During Continuous Integration (CI)	In CI/CD environments, testing is continuou

Testing Review Process	
Step	Description
1. Preparation	Collect the necessary artifacts for review, such as requirements documents, test plans, or test cases.
2. Review Initiation	Schedule review meetings with relevant stakeholders, including testers, developers, and business analysts.
3. Individual Review	Team members independently analyse the artifacts to identify potential issues or areas for improvement.

4. Group Review Meeting	Discuss findings, consolidate feedback, and agree on corrective actions or improvements.
5. Report Feedback	Document the review outcomes, including identified issues, decisions, and next steps.
6. Action Implementation	Address feedback by updating test plans, cases, or processes as agreed during the review meeting.
7. Follow-Up	Verify that the recommended changes have been implemented correctly.



Types of Integration Testing			
Type	Description	Advantages	Challenges
1. Big Bang Testing	All modules are integrated and tested simultaneously after development is complete.	- Simple to implement.	- Difficult to isolate and debug defects.
		- No need for a phased approach.	- High risk of major failures.
2. Incremental Testing	Modules are integrated and tested step-by-step in a sequence.	- Easier defect isolation.	- Requires a well-planned integration strategy.
		- Problems are identified early.	- Time-consuming.
3. Top-Down Testing	High-level modules are tested first, and lower-level modules are integrated progressively.	- Helps identify design flaws early.	- Stubs may be required for incomplete lower modules.
		- Major functionality is tested first.	- Some modules may be under-tested.
4. Bottom-Up Testing	Lower-level modules are tested first, followed by higher-level modules.	- Critical lower-level modules are tested early.	- Drivers may be needed for testing higher-level modules.
		- No stubs are needed.	- User interface may be tested late.
5. Sandwich Testing	Combines Top-Down and Bottom-Up testing, testing both high-level and low-level modules simultaneously.	- Faster integration of modules.	- Requires skilled planning.
		- Covers critical high- and low-level modules early.	- May be complex to manage.
6. Hybrid Testing	A customized approach combining multiple integration	- Flexible and adaptable to	- Requires careful planning and expertise.

	strategies based on project requirements.	complex systems.	
		- Allows prioritization of critical modules.	- May involve extra coordina

Testing Methodologies

Testing Methodologies	
Testing Methodology	Description
1. Agile Testing	Testing is integrated into Agile development cycles, ensuring continuous testing alongside development.
2. Waterfall Testing	Follows the sequential Waterfall model, where testing occurs after the development phase.
3. V-Model Testing	Testing activities run in parallel with corresponding development phases (verification and validation).

Software Testing Techniques

Software Testing Techniques		
Technique	Type	Description
Boundary Value Analysis	Functional	Tests boundaries of input values (e.g., minimum, maximum) to check system behaviour at edges.
Equivalence Partitioning	Functional	Divides input data into partitions to reduce test cases while maintaining coverage.
Decision Table Testing	Functional	Tests combinations of inputs and corresponding outputs using a tabular format.
State Transition Testing	Functional	Verifies system behaviour under different states and transitions (e.g., login states).

Coverage Areas in Software Testing

Coverage Areas in Software Testing		
Coverage Area	Description	Examples
1. Behavioural Coverage	Tests the application's functional behaviour to ensure it performs as intended based on requirements.	- Does the login button redirect to the correct page?
		- Does the search functionality return accurate results?

2. Input Domain Coverage	Verifies the application's response to a wide range of inputs, including valid, invalid, and edge cases.	- Testing text fields for valid/invalid characters.
		- Validating numeric inputs within allowed ranges.
3. Error Handling Coverage	Ensures that the application properly handles errors and exceptions, and provides appropriate feedback.	- Handling invalid user inputs (e.g., wrong password).
		- Graceful handling of system failures (e.g., database issues).
4. Back-End Coverage	Validates the integrity of data, transactions, and system interactions on the server or database.	- Ensuring data is correctly saved in the database after form submission.
		- Verifying API responses are accurate and consistent.
5. Service Coverage	Ensures that external and internal services (APIs, microservices) work correctly and reliably.	- Testing third-party API integrations (e.g., payment gateways).
		- Ensuring microservices communicate effectively.
6. Calculation Coverage	Verifies that calculations (whether financial, mathematical, or logical) are correct and consistent.	- Testing tax calculation based on varying conditions.
		- Verifying that interest rates are applied correctly.

--