SOFTWARE TESTING FUNDAMENTALS

About me

- 19 years of IT industry experience wearing multiple hats in leadership roles in Dev, QA and Delivery
- Passionate about teaching and imparting knowledge
- 3 years of training experience with over 1000 hours of programs conducted
- Areas of expertise include Manual, Automation Testing, API and Performance Testing
- Other interests include Python, Network Automation, AI/ML
- https://www.linkedin.com/in/nagaraj-ravinuthala/

AGENDA

- What and Why of Software Testing
- Different Types, Methods and Levels of Testing
- Test Plan
- Test Design
- RTM

WHAT IS SOFTWARE TESTING?

- Evaluating a software system or its components with an intent to find <u>defects</u>
- We do this by executing the system with an intent find <u>gaps</u> or <u>errors</u>
- Gaps / Errors — Defects / Bugs

WHAT IS SOFTWARE TESTING?

- What is a gap?
  - Expected — Home loan interest rate would vary based on the amount of loan (like for loan < 20 lakhs its 7%, for loan > 20 lakhs its 8%)
  - Actual — Interest rate is set to 8% irrespective of the amount of loan.
- What is an error?
  - The new interest as per the bank is 8.5%, but the user is being charged at 8% because he rounded off the value mistakenly

A process of analyzing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item.

– ANSI/IEEE 1059 Standard

WHY DO WE NEED TO TEST THE SOFTWARE?

- **Nissan** had to recall over 1 million cars from the market due to **software failure** in the airbag sensory detectors
- In April 2015, **Bloomberg** terminal in London crashed due to **software glitch** affected more than 300,000 traders on financial markets. It forced the government to postpone a 3bn pound debt sale.
- **Starbucks** recently experienced a massive glitch in its point-of-sale (POS) systems which disabled cash registers at thousands of stores. Starbucks was forced to give away products for free before it finally shut the affected stores down
- Some of the **Amazon's** third-party retailers saw their product price is reduced to 1p due to a **software glitch**. They were left with heavy losses.

Seven Principles of Software Testing

- Consider the following scenario
  - You are moving a file from folder A to Folder B
- What would be your test scenarios for this?
- Think of all the possible test scenario for this

Seven Principles of Software Testing

Apart from the usual scenarios, you can also test the following conditions

- Trying to move the file when it is Open
- You do not have the security rights to paste the file in Folder B
- Folder B is on a shared drive and storage capacity is full.
- Folder B already has a file with the same name, in fact, the list is endless
- Or suppose you have 15 input fields to test, each having 5 possible values, the number of combinations to be tested would be 5^15

Seven Principles of Software Testing

- Exhaustive testing is not possible
- Defect clustering
- Pesticide paradox
- Testing shows presence of defects
- Absence of error fallacy
- Early testing
- Testing is context dependent

Basic Software Testing Principles – Software Quality Assurance & Test Automation (qatestautomation.com)

QUALITY ASSURANCE VS QUALITY CONTROL

- QA usually refers to a set of principles and practices that organizations need to follow to Assure the Quality of their products
  - QA ensures we are involved throughout the development process
- QC usually refers to a set of activities to control the Quality by validating it against the original specification
  - QC ensures we act as gatekeepers before the product is rolled out

TYPES OF TESTING

- Types of testing can broadly be classified as:
  - Functional Testing
  - Non Functional Testing
- Both above types of testing can be done either manually or via automation

METHODS OF TESTING

- Manual Testing
  - Tester assumes the role of end user
  - Uses test plans, scenarios, cases etc.
- Automation Testing
  - Tester uses scripts or other software to validate the product
  - Helps in quick and repeated execution of test scenarios
  - Makes it ideal for regression testing
  - Automation scripts can also be used during performance testing

TESTING METHODS

- Black-box Testing
  - Treats the system as a black-box
  - All that matters is what goes in and what comes out
  - Oblivious to the internal architecture and code
- White-box Testing
  - Deals with detailed investigation of internal logic and code
  - Includes statement and branch coverage
  - Can also include condition/ path coverage etc.
  - Can be done at unit, integration or system levels
  - Tests for memory leaks, security loopholes via penetration testing

TESTING LEVELS

- Unit Testing
  - Developers test the components they are developing at the most basic unit level
  - Can be done either manually or via automation
- Integration Testing
  - Aims to test integrations between combined parts of application
  - Can be done manually or via automation
- Regression Testing
  - Aims to ensure existing components are not destabilised while adding new features
  - Preferably done via automation as this is a repetitive activity

TESTING LEVELS

- System Testing
  - Aims to verify the entire system as a whole
  - Aims to verify both functional and technical specifications
  - Tested in environment close to production environment
  - Verifies business requirements as well as system architecture
- User Acceptance Testing
  - Aims to validate the final product against the requirements as end user would use them

NON FUNCTIONAL TESTING

- Performance Testing
  - Identify bottlenecks in terms of Networks Delays, DB Transactions etc.
  - Has sub-types
    - Load Testing
      - Aims to test the system behaviour by applying maximum load
      - Can be done during normal and peak load conditions
    - Stress Testing
      - Aims to observe the behaviour of the system under abnormal conditions

NON-FUNCTIONAL TESTING

- Security Testing
  - Identify the vulnerabilities,
- Usability Testing
  - How easy and intuitive is the product to use
  - Efficiency of use, Learnability, Memorability, Errors/ Safety and overall satisfaction
- Localization Testing
  - Working of the product in different locales
- Interoperability Testing
  - Working of the product across browsers, environments

TEST PLAN

Why do we need to plan?

Anything done without proper plan in place is bound to fail

Describes Scope, Approach, Resources, Schedule of intended test activities

Identifies test items, features to be tested, who does what, test environment, entry and exit criteria etc.

ACTIVITY

- Prepare a plan for office outing

TYPES OF TEST PLANS

- Master Test Plan
  - A single high-level plan for a project or product that unifies all other plans
- Level wise Test Plan
  - Unit Test Plan
  - Integration Test Plan
  - System Test Plan
  - Acceptance Test Plan etc.
- Type wise Test Plan
  - Performance Test Plan
  - Security Test Plan etc.

TEST PLAN SECTIONS

- Scope
  - Features to be tested/ not to be tested
- Approach
  - Overall approach to testing
  - Testing Types, Levels and Methods to be used
    - Unit, Integration, System, Acceptance etc.
    - Manual vs Automated
    - Black box vs White box
    - Whether security and performance to be included

TEST PLAN SECTIONS

- Entry/ Exit criteria
- Test Deliverables
  - Test plan, Test cases, scripts, execution reports, defect reports
- Estimate and Schedule
- Staffing and Training Needs
- Assumptions/ Risks/ Mitigation Plan

TEST PLAN GUIDELINES

- Keep it concise, avoid redundancy and ambiguity
- If you borrow a template make sure to delete unwanted sections
- Be specific e.g. while talking about prerequisite software ensure to mention proper versions as well

TEST PLAN GUIDELINES

- Avoid lengthy paras and use lists, tables as needed
- Get it reviewed thoroughly, make it a quality plan
- Keep it dynamic

TEST CASE DESIGN TECHNIQUES

- Specification based or blackbox technique
  - Boundary Value Analysis
  - Equivalence partitioning
  - Decision table testing
  - State transition diagrams
  - Use case testing

TEST CASE DESIGN TECHNIQUES

- Structure based or whitebox technique
  - Statement testing and coverage
  - Decision testing and coverage
  - Condition testing
  - All path testing

TEST CASE DESIGN TECHNIQUES

- Experience based technique
  - Error guessing
  - Exploratory testing

TEST CASE DESIGN TECHNIQUES

- https://reqtest.com/testing-blog/test-case-design-techniques/

TEST DESIGN

- While Test Plan is external facing, Test Design is for internal consumption of the testing team
- Tasks about how we go about our actual work on day to day basis
- Listing various test conditions, thinking about test data, highlighting test scenarios, writing test cases for those scenarios

WRITING TEST SCENARIOS

- Read and understand thoroughly various documents like BRS, SRS, FRS
- For each requirement figure out all possible user actions and objectives
- After due analysis of requirements list out all possible scenarios to verify those requirements
- Once all the test scenarios are listed, create RTM
- Get the test scenarios reviewed by Test Lead and other stakeholders

WRITING TEST SCENARIOS

- Go through the specifications document meticulously and derive test cases
- If there are conditions applied, create a test case that makes every condition once true and once false
- If there are several branches of any process, make sure that test case cover every branch at least once
- Make sure you know the ‘Expected Result’ for each case
- Same action can not have two outputs
- Identify scenarios for cross-functional testing

USE CASE BASED TEST SCENARIOS

- A use case describes various actors and their interactions with the system
- Use cases cover the complete transactions from start to finish
- Depict the actual use of software by the end user
- Are significant for the acceptance of software by the users
- Develop at least one test case for a normal use
- Cover all use cases. Cover all users
- Identify any dependency between different use cases and cover those
- Develop a test cases for scenarios that have not been identified in the use case. Because that is the place where you are more likely to find loopholes

TEST SCENARIO VS TEST CASE

- Take eCommerce Application — Amazon.co.in
- Test Scenario: Check Login Functionality
- Test Cases
  - Check system behavior when valid email id and password is entered
  - Check system behavior when *invalid* email id and *valid* password is entered
  - Check system behavior when *valid* email id and *invalid* password is entered
  - Check system behavior when *invalid* email id and *invalid* password is entered
  - Check system behavior when email id and password are left blank and Sign in entered
  - Check Forgot your password is working as expected
  - Check system behavior when valid/invalid phone number and password is entered.
  - Check system behavior when "Keep me signed" is checked

TEST SCENARIO VS TEST CASE

- Other Test Scenarios
  - Check Search Product Functionality
  - Check Product Description Page
  - Check Payments Functionality
  - Check Order History

- So next time you are on a web page or an app keep thinking how you would test various things that you see there

CHECKLISTS

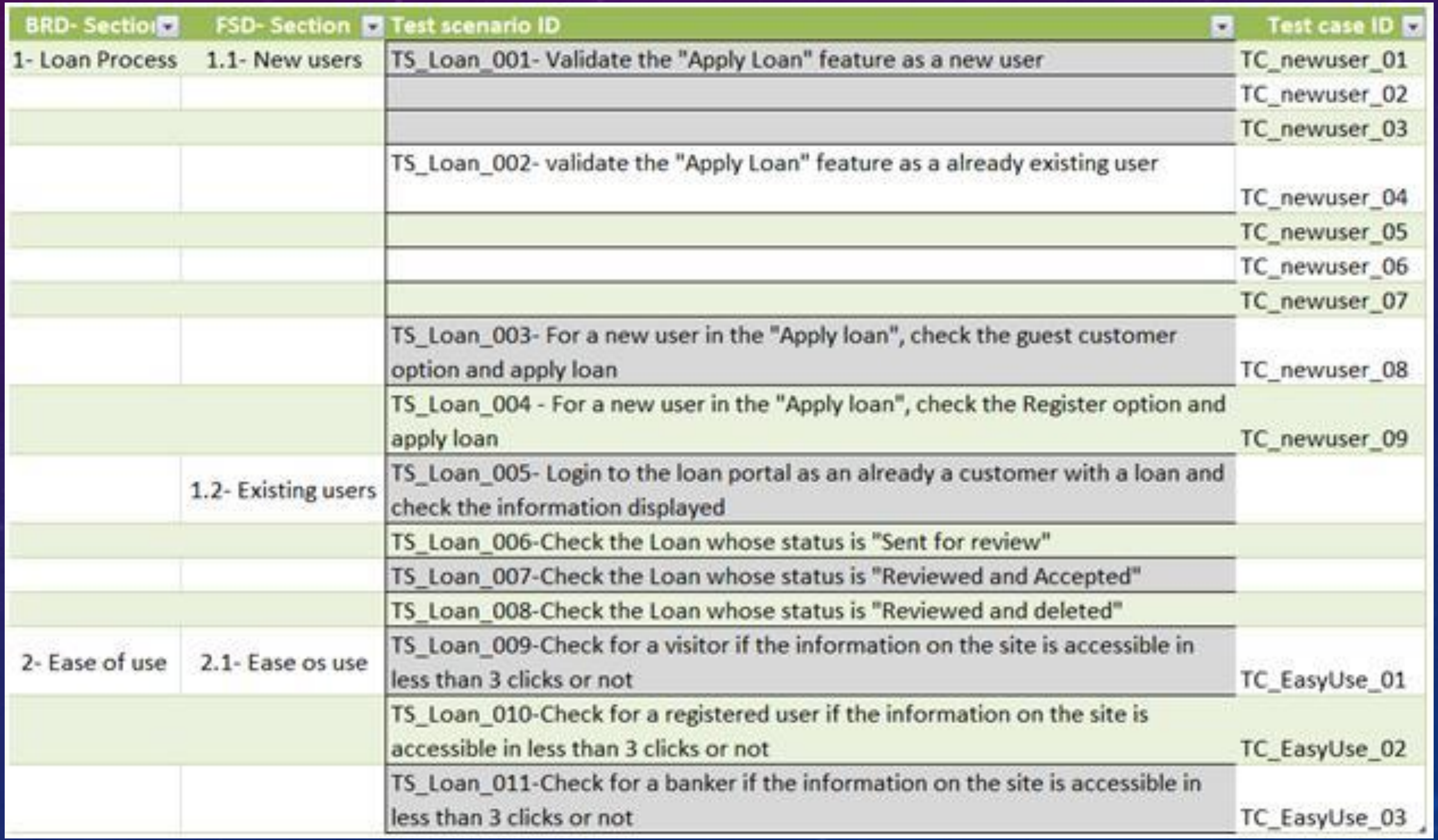
- What are checklists?
- Why do we need them?
- How can we use them in testing?

Test Readiness Review (TRR) Criteria	Status
All the Requirements finalized and analyzed	Done
Test plan created and reviewed	Done
Test case preparation done	
Test Case review and sign off	
Test data availability	
Is Sanity Testing done?	
Team aware of the roles and responsibilities	
Team aware of the deliverables expected of them	

CHECKLISTS

- Entry exit criteria checklist
  - Will help in decision making when to start or stop testing
- Automation checklist
  - Will help you to decide whether you can start investing time in automation or not?
  - Whether features are stable

REQUIREMENT TRACEABILITY MATRIX



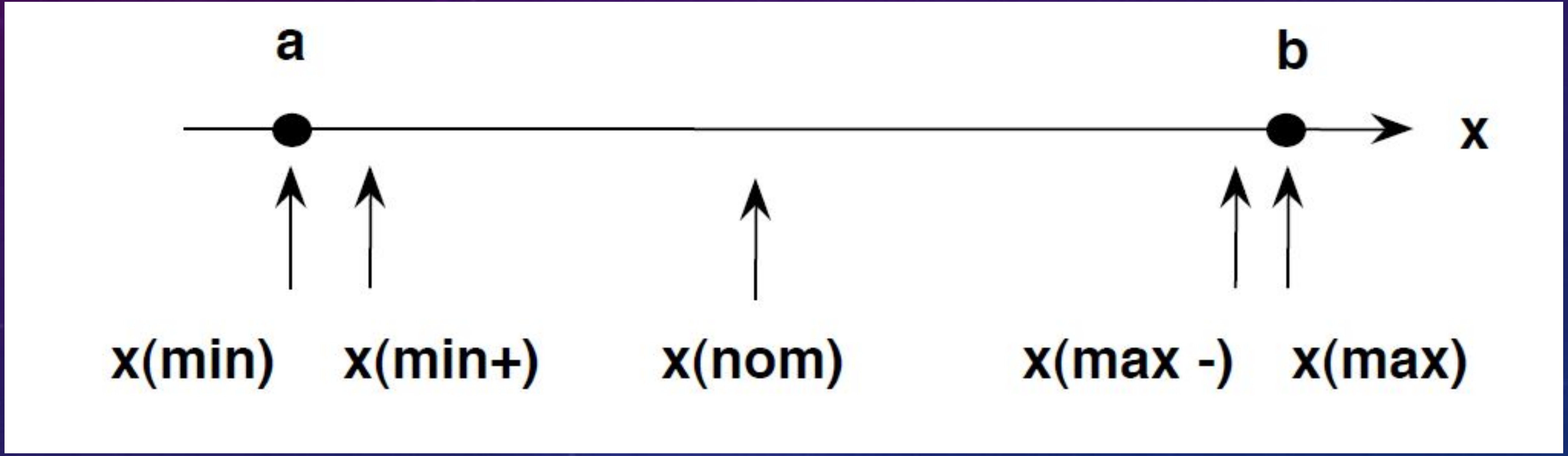
BRD- Sectio	FSD- Section	Test scenario ID	Test case ID
1- Loan Process	1.1- New users	TS_Loan_001- Validate the "Apply Loan" feature as a new user	TC_newuser_01
			TC_newuser_02
			TC_newuser_03
		TS_Loan_002- validate the "Apply Loan" feature as a already existing user	TC_newuser_04
			TC_newuser_05
			TC_newuser_06
			TC_newuser_07
		TS_Loan_003- For a new user in the "Apply loan", check the guest customer option and apply loan	TC_newuser_08
		TS_Loan_004 - For a new user in the "Apply loan", check the Register option and apply loan	TC_newuser_09
	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	
		TS_Loan_006-Check the Loan whose status is "Sent for review"	
		TS_Loan_007-Check the Loan whose status is "Reviewed and Accepted"	
		TS_Loan_008-Check the Loan whose status is "Reviewed and deleted"	
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
		TS_Loan_010-Check for a registered user if the information on the site is accessible in less than 3 clicks or not	TC_EasyUse_02
		TS_Loan_011-Check for a banker if the information on the site is accessible in less than 3 clicks or not	TC_EasyUse_03

BOUNDARY VALUE ANALYSIS

- There could be situations, due to time and budget constraints, we may not be able to carry out exhaustive testing for different sets of test data
- In such cases we resort to BVA
- And come up with Boundary Testing
- We also split our test data into partitions called Equivalence Partitions

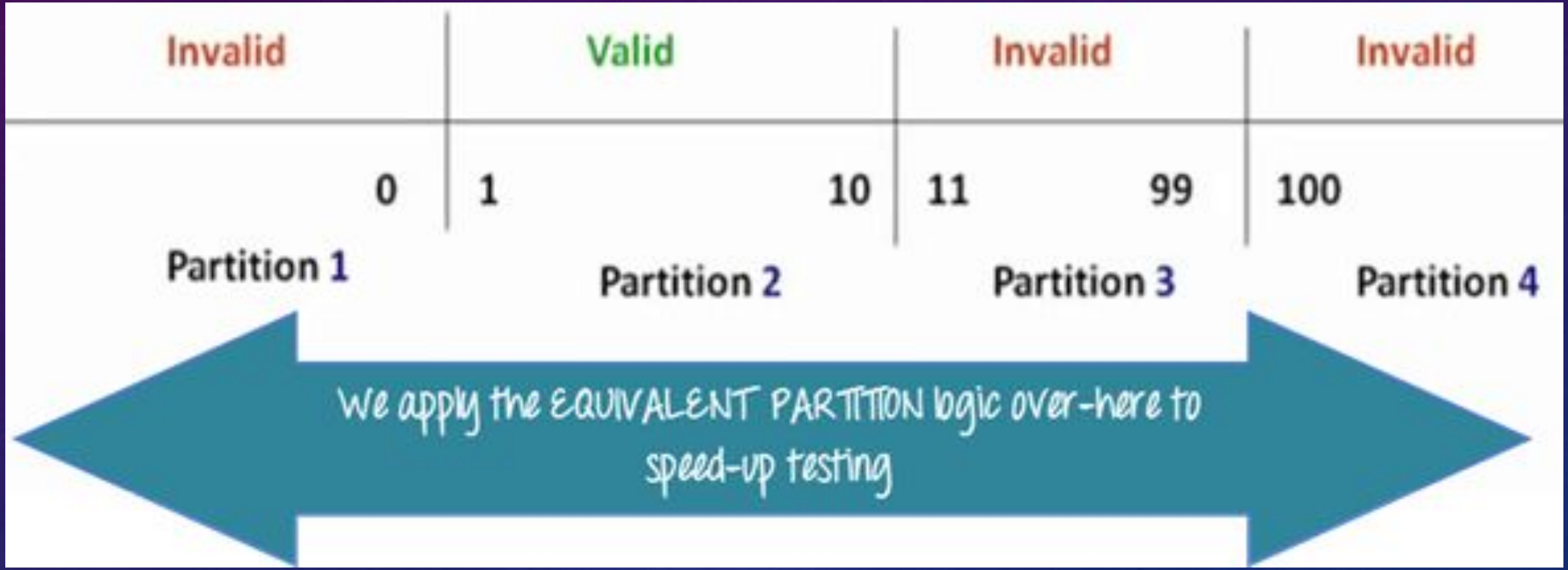
BOUNDARY TESTING

- Boundary testing is the process of testing between extreme ends or boundaries between partitions of the input values.
- The basic idea in boundary value testing is to select input variable values at their:
    - Minimum
    - Just above the minimum
    - A nominal value
    - Just below the maximum
    - Maximum


a
b
x
x(min)
x(min+)
x(nom)
x(max -)
x(max)

EQUIVALENCE PARTITIONING

- Closely related to Boundary Value Analysis or Boundary Testing
- Let's say Dominos allows a person to order only 10 pizzas at a time
- Pizza values 1 to 10 is considered valid. A success message is shown.
- While value 11 to 99 are considered invalid for order and an error message will appear


Invalid
Valid
Invalid
Invalid
0
1
10
11
99
100
Partition 1
Partition 2
Partition 3
Partition 4
We apply the EQUIVALENT PARTITION logic over-here to speed-up testing

EQUIVALENCE PARTITIONING

- Example: Equivalence Partitions and Boundary Value Analysis
- Say a password field accepts minimum 6 characters and maximum 10 characters
- That means results for values in partitions 0-5, 6-10, 11-14 should be equivalent

Test Scenario #	Test Scenario Description	Expected Outcome
1	Enter 0 to 5 characters in password field	System should not accept
2	Enter 6 to 10 characters in password field	System should accept
3	Enter 11 to 14 character in password field	System should not accept

DECISION TABLE TESTING

- Used to test the system behaviour when multiple inputs are available and various combinations of those inputs have to be considered
- These combinations and the corresponding behaviour (output) are captured in a tabular form
- This is called Cause-Effect table as this table captures the cause and the effect they produce for a better coverage

DECISION TABLE TESTING

- Used to test the system behaviour when multiple inputs are available and various combinations of those inputs have to be considered
- These combinations and the corresponding behaviour (output) are captured in a tabular form
- This is called Cause-Effect table as this table captures the cause and the effect they produce for a better coverage

DECISION TABLE FOR LOGIN SCENARIO

- There is a login page with input fields username and password
- If the user is indeed authorised and password is correct, it should allow the user to login
- Note the cause and effect here.
- Cause is, username and password being correct.
- Effect is, user is allowed to login
- So this is also referred to as Cause-Effect Table

DECISION TABLE FOR LOGIN SCENARIO

CONDITIONS	RULE 1	RULE 2	RULE 3	RULE 4
USERNAME (T/F)	F	T	F	T
PASSWORD (T/F)	F	F	T	T
OUTPUT (E/H)	E	E	E	H

- Legend:
  - T — Username / Password being correct
  - F — Username / Password being wrong
  - E — Error message displayed
  - H — User is taken to home page

DECISION TABLE FOR LOGIN SCENARIO

- Interpretation:
  - Case 1 – Username and password both were wrong. The user is shown an error message.
  - Case 2 – Username was correct, but the password was wrong. The user is shown an error message.
  - Case 3 – Username was wrong, but the password was correct. The user is shown an error message.
  - Case 4 – Username and password both were correct, and the user navigated to homepage

DECISION TABLE FOR LOGIN SCENARIO

- What is its advantage?
- Helps us to see if we have covered all the possible conditions easily
- When there is time constraint we can reduce the number of test cases

DECISION TABLE FOR LOGIN SCENARIO

- Consider another scenario: File upload
- Conditions for uploading a file successfully are:
  - You can upload only '.jpg' format image
  - File size less than 32kb
  - Resolution 137*177
- Imagine having to write these test cases directly in text format

Conditions	**Case 1**	**Case 2**	**Case 3**	**Case 4**	**Case 5**	**Case 6**	**Case 7**	**Case 8**
Format	.jpg	.jpg	.jpg	.jpg	Not .jpg	Not .jpg	Not .jpg	Not .jpg
Size	Less than 32kb	Less than 32kb	>= 32kb	>= 32kb	Less than 32kb	Less than 32kb	>= 32kb	>= 32kb
resolution	137*177	Not 137*177	137*177	Not 137*177	137*177	Not 137*177	137*177	Not 137*177
Output	Photo uploaded	Error message resolution mismatch	Error message size mismatch	Error message size and resolution mismatch	Error message for format mismatch	Error message format and resolution mismatch		

DECISION TABLE FOR LOGIN SCENARIO

- When the system behavior is different for different input and not same for a range of inputs, both equivalence partitioning, and boundary value analysis won't help, but decision table can be used

DECISION TABLE FOR LOGIN SCENARIO

- Decision table can be used for doing all-pairs testing or pair-wise testing
- Let's assume we are in the Wild West… and there is a web page with the following elements

ELEMENT	ALLOWED VALUES
List Box	0,1,2,3,4,5,6,7,8,9
Check Box	Checked, Unchecked
Radio Button	On, Off
Text Box	Any value between 1 to 100

DECISION TABLE FOR LOGIN SCENARIO

- List box — 10
- Check box — 2
- Radio box — 2
- Text box — 100
- Total combinations possible — 10*2*2*100 = 4000 (positive)
- Number is much more if we consider negative test cases

DECISION TABLE FOR LOGIN SCENARIO

- Can you imagine the amount of efforts needed to document all these cases, execute them and log results?
- Let's see if we have a workaround or shortcut here.
- Luckily we have a shortcut. Its called pair-wise testing or all-pairs testing

DECISION TABLE FOR LOGIN SCENARIO

- We start by looking at how many possible values can be taken by each control.
- List box box input values can be considered as 2 (i.e. 0 and others — 1 to 9)
- Radio box and check box can either be selected or unselected. So they both amount to 2 input values each.
- The Text box values can be reduced into three inputs (Valid Integer, Invalid Integer, Alpha-Special Character)
- So the number of possible test cases is 2*2*2*3 = 24

DECISION TABLE FOR LOGIN SCENARIO

- Can we further reduce this?
- Yes - using all-pairs testing technique
- **Step 1 :** Order the values such that one with most number of values is the first and the least is placed as the last variable.
- **Step 2 :** Now, start filling the table column by column. List box can take 2 values.
- **Step 3 :** The next column under discussion would be check box. Again, Check box can take 2 values.
- **Step 4 :** Now, we need to ensure that we cover all combinations between list box and Check box.
- **Step 5 :** Now, we will use the same strategy for checking the Radio Button. It can take 2 values.
- **Step 6 :** Verify if all the pair values are covered as shown in the table below.

Text Box	List Box	Check Box	Radio Button
Valid Int	0	check	ON
Valid Int	others	uncheck	OFF
Invalid Int	0	check	ON
Invalid Int	others	uncheck	OFF
AlphaSpecialCharacter	0	check	ON
AlphaSpecialCharacter	others	uncheck	OFF

TEST ENVIRONMENT SETUP

- https://www.tutorialspoint.com/stlc/stlc_test_environment_setup.htm

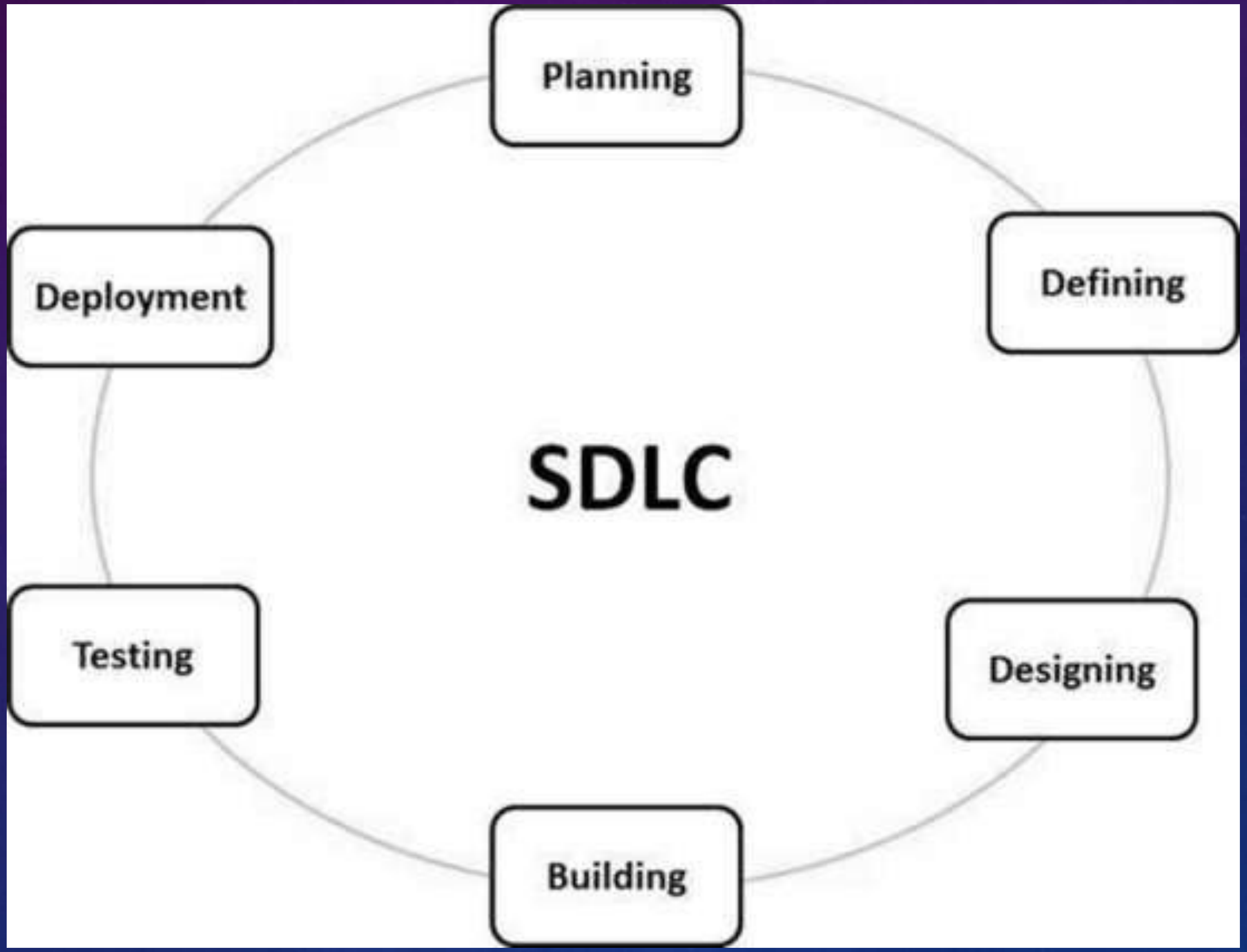
JIRA

- JIRA is an incident management tool
- Can be used for Project Management, Bug and Issue tracking
- Main concepts revolve around Project, Issue and Workflow
- Its platform independent, multi-lingual and supports multiple DBs at the backend

USES OF JIRA

- Requirement management and Task tracking
- Workflow and Process Management
- As a Help Desk for Support and Customer service related tasks to create tickets and track them to resolution
- For Bug and Issue tracking
- For change request management

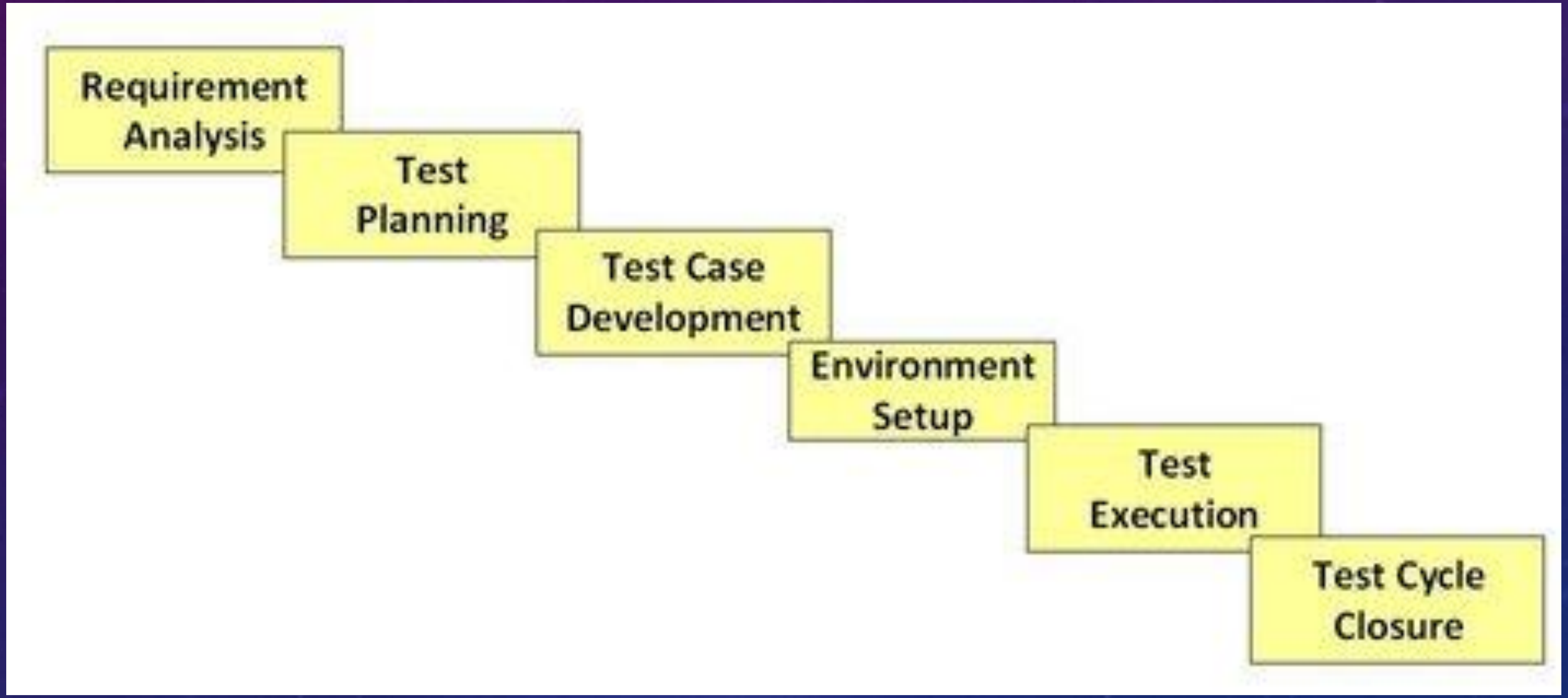
SDLC
SOFTWARE DEVELOPMENT LIFE CYCLE



SDLC
SOFTWARE DEVELOPMENT LIFE CYCLE

- Stage 1: Planning and Requirement Analysis
- Stage 2: Defining Requirements
- Stage 3: Designing the Product Architecture
- Stage 4: Building or Developing the Product
- Stage 5: Testing the Product
- Stage 6: Deployment in the Market and Maintenance

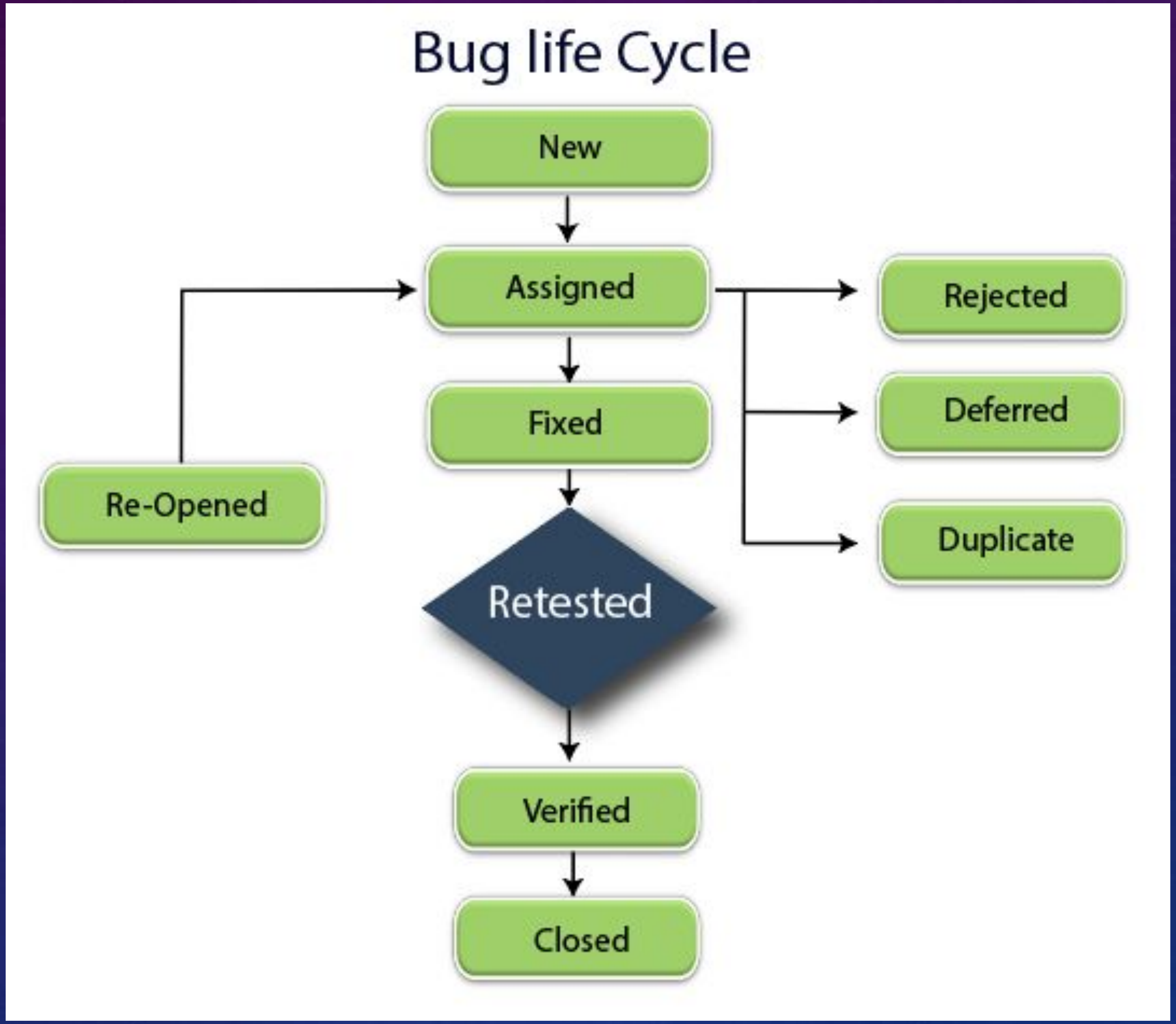
STLC
SOFTWARE TESTING LIFE CYCLE



STLC
SOFTWARE TESTING LIFE CYCLE

- Stage 1: Requirement Analysis
- Stage 2: Test Planning
- Stage 3: Test Case Development
- Stage 4: Test Environment Setup
- Stage 5: Test Execution
- Stage 6: Test Closure

DEFECT LIFE CYCLE



DEFECT RAISING GUIDELINES

- Keep the defect summary concise
- Mention the steps to reproduce clearly
- Describe the test env in detail
- Do not be too verbose, stick to the facts
- Attach snapshots and error logs
- Avoid grouping too many issues under single defect
- Try to avoid duplicate defects
- Choose appropriate severity and priority for the defect

SEVERITY & PRIORITY OF DEFECTS

- Severity
  - Defines the impact of the defect to the system
  - 1 – Critical
  - 2 – High
  - 3 – Medium
  - 4 – Low

SEVERITY & PRIORITY OF DEFECTS

- Priority
  - Defines how quickly a defect should be fixed (irrespective of the severity)
  - P1 – Fix defect immediately
  - P2 – Give high attention
  - P3 – Normal queue
  - P4 – Low priority
  - P5 – Can be deferred

SEVERITY & PRIORITY OF DEFECTS

- Defect with less severity but high priority
  - Spelling mistake on the homepage
- Defect with high severity but less priority
  - Corner case (like repetitive clicking on some button crashes the application)

Manual Testing Hands-On

- Read the given use case
- Prepare the following test artifacts
  - Test scenarios
  - Test cases
  - Defect tracker
  - RTM (Requirement traceability matrix)