1. Introduction to software Testing
2. Software development process
3. Levels and types of testing

**Testing Techniques**

**Black Box testing**

**External interfaces, inputs and expected outputs**

1. **Equivalence partitioning**
2. **Boundary value analysis**
3. **Decision table testing**
4. **State transition testing**
5. **Error guessing**
6. **Use case based testing**

**System: 18 to 60 age will be considered as valid**

**Valid partition**

Age : 18 to 60

**Invalid partition**

* age less than 18
* Age greater than 60

**Test cases**

* Test with age 25 (valid)
* Test with age 16 ( invalid)
* Test with age 40( Valid)
* Test with age 65( Invalid)
* Test with age 18 (valid)
* Test with age 60 (valid)

**Email address validation**

[**user@example.com**](mailto:user@example.com)

**Invalid partition**

* Email address without @ symbol
* Email address without a domain user@
* Email address without top level domain usr@example

**Test cases**

1. Test with a valid email [user@example.com](mailto:user@example.com)
2. Test with email address missing @
3. Test with email address missing domain
4. Test with email address missing top-level domain
5. Test with empty email
6. Test with multiple domain ([user@sub.example.com](mailto:user@sub.example.com))

**Login crendetial**

User name password

**Valid :**

Valid user name and password

**Invalid**

* **Invalid username**
* **Invalid password**
* **Both are invalid**

**Test cases**

* Test with valid user name and password
* Test with invalid user name
* Test with invalid password
* Test with both invalid
* Test with empty username (invalid)
* Test with empty password (invalid)

**Numeric Range input**

**1 to 100**

**Valid partition:**

**Number between 1 to 100**

**Invalid partition**

**Number less than 1**

**Number greater than 100**

**Test cases:**

1. 50 ( valid)
2. 0 ( invalid)
3. 75 ( Valid)
4. 105( Invalid)

**Boundary Value analysis**

**Numeric Range input**

Lower 1

Max : 100

Boundary values

Lower 1

Upper 100

Test cases:

1. Test with the value as 1
2. Test with the value 0
3. Test with the value 2
4. Test with the value 100
5. Test with the value 99
6. Test with the value 101

N----------------------------M

0 1 2 99 , 100, 101

N-1, N , N+1 M-1, M, M+1

**Date range Input**

**Where user need to input a date , and it should be with in a range**

**Boundary values:**

* **Lower bound :** January 1, 2020 ---🡪 N
* **Upper bound :** December 31, 2022 --🡪 M

N----------------------------M

N-1, N , N+1 M-1, M, M+1

**Test cases:**

1. Date just above the lower bound ( January 2, 2020)
2. Date with the lower bound( January 1, 2020)
3. Date just below the lower bound ( December 31,2019)
4. Date just below upper bound ( December 30 , 2022)
5. Date with the upper bound ( 31 December, 2022)
6. Date just above the upper bound ( January 1, 2023)

**File Size Limit**

**A system has file limit on upload**

**Boundary values:**

* **Lower Bound: 1 MB**
* **Upper Bound: 10 MB**

**Test cases**

1. Test with the file size just above the lower bound ( 1.1/1.2/1.5 Mb)
2. Test with the lower bound ( 1Mb)
3. Test with file size just below the lower bound ( 0.9/0.5 Mb)
4. Test with the file size 9 mb
5. Test with the file size 10 MB
6. Test with the file size 11 Mb

**Password Length**

**Set password with a specific length**

**Boundary Values:**

* **Minimum length: 8 characters**
* **Maximum Length: 20 Characters**

**Test cases**

1. Test with 7
2. Test with 8
3. Test with 9
4. test with just below char 19
5. Test with maxi char 20
6. test with maxi char 21

**Check box selection**

**Boundary values:**

**Minimum Selection: 1 checkbox**

**Maximum Selection: 5 Checkboxes**

**Test cases:**

1. **Test with the minimum selection (1)**
2. **Test with the just below the minimum (0)**
3. **Test with the just above the min (2)**
4. **Test with the maximum value selection (5)**
5. **Test with the just below the maximum ( 4)**
6. **Test with the just above the max(6)**

**Decision Table testing**

Consider a system that decides whether to approve or reject a loan application based on the applicant’s credit score, income, employment status.

**Conditions**

1. Credit score: Low, medium, High
2. Income: Low, medium, high
3. Employment status: Unemployed, Employed, Self Employed

**Action:**

1. **Accept**
2. **Reject**

|  |  |  |  |
| --- | --- | --- | --- |
| **Credit Score** | **Income** | **Employment status** | **Action** |
| Low | Low | Unemployed | **Reject** |
| Low | Low | Employed | **Reject** |
| **Low** | **Low** | **Self employed** | **Reject** |
| Low | **Medium** | **Unemployed** | **Reject** |
| **Low** | **Medium** | **Employed** | **Reject** |
| **Low** | **Medium** | **Self Employed** | **Reject** |
| **Low** | **High** | **Unemployed** | **Reject** |
| **Low** | **High** | **Employed** | **Reject** |
| **Low** | **High** | **Self Employed** | **Reject** |
| Medium | Low | Unemployed | **Reject** |
| Medium | Low | Employed | **Reject** |
| Medium | **Low** | **Self employed** | **Reject** |
| Medium | **Medium** | **Unemployed** | **Reject** |
| Medium | **Medium** | **Employed** | **Approve** |
| Medium | **Medium** | **Self Employed** | **Approve** |
| Medium | **High** | **Unemployed** | **Reject** |
| Medium | **High** | **Employed** | **Approve** |
| Medium | **High** | **Self Employed** | **Approve** |
| **High** | Low | Unemployed | **Reject** |
| **High** | Low | Employed | **Approve** |
| **High** | **Low** | **Self employed** | **Approve** |
| **High** | **Medium** | **Unemployed** | **Reject** |
| **High** | **Medium** | **Employed** | **Approve** |
| **High** | **Medium** | **Self Employed** | **Approve** |
| **High** | **High** | **Unemployed** | **Reject** |
| **High** | **High** | **Employed** | **Approve** |
| **High** | **High** | **Self Employed** | **Approve** |

**Online Shopping Discount:**

**Suppose a system provides discount based on a customers membership level and the total purchase amount**

**Condition:**

1. **Membership level (Basic, Silver, Gold)**
2. **Total Purchase amount ( Low, Medium, High)**

**Action:**

* **Apply discount**
* **Do not apply discount**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Membership Level** | **Total purchase amount** | **Action** |
| 1 | Basic | **Low** | **Do not apply** |
| **2** | **Basic** | **Medium** | **Do not Apply** |
| **3** | **Basic** | **High** | **Apply discount** |
| **4** | **Silver** | **Low** | **Do not apply** |
| **5** | **Silver** | **Medium** | **Apply discount** |
| **6** | **Silver** | **High** | **Apply discount** |
| **7** | **Gold** | **Low** | **Apply discount** |
| **8** | **Gold** | **Medium** | **Apply Discount** |
| **9** | **Gold** | **High** | **Apply Discount** |

**State Transition Diagram**

**Traffic Light system**

**RED -----🡪 Green ----🡪 Yellow**

1. **If the light is red , it will transition to Green after a certain time.**
2. **If the light is green it will transition to yellow after certain time**
3. **If the light is yellow it will transition to Red after a certain time**
4. **There should not be a direct transition from Red to yellow or yellow to green.**

**State Transition Table**

|  |  |  |
| --- | --- | --- |
| **Current State** | **Event** | **Next State** |
| **Red** | **Timer expires** | **Green** |
| **Green** | **Timer Expires** | **Yellow** |
| **Yellow** | **Timer Expires** | **Red** |

**State Transition Test cases**

1. **Initial State test:**

**Start with the initial state red**

1. **Red to green transition:**

Transition from red to green and verify if it occurs after timer expires

1. Green to yellow:

Transition from green to yellow and verify if this occurs after timer expires

1. Yellow to Red:

Transition from yellow to red and verify if this occurs after timer expires

1. Invalid Transition test:

Attempt to transition directly from Red to yellow or yellow to green and ensure that it does not happen

ATM Machine

Input the card --🡪 Correct Password 🡪 Password accepted

Incorrect Passord -🡪 Reattempt

Incorrect Password -🡪 Reattemt

Correct password 🡪 Login

Incorrect password 🡪 Card bloccked

**Error Guessing**

Adhoc testing

Find defects by random checking

Done by experienced people

Experience – based intuition

Exploratory testing

Focused testing

Informal testing

Supplemental technique

Example

Input validation

Boundary conditions

Compatibility and integration

Stress and load testing

Security vulnerabilities

Usability and User experience

Edge cases

**Banking System**

**Use case: Transfer funds**

Description: The system allows users to transfer funds between accounts.

Main flow

1. User selects the transfer option
2. User enters the recipients account details and the amount to transfer
3. System verifies the availability of fund and execute the transfer
4. System displays confirmation message

Alternative flow: the recipients account details are incorrect and system displays an error message.

The enough fund is not available and system displays an error message.

Test Scenarios/ Test case1

1. Successful transfer

* User initiate a transfer with valid details
* Expected result: Funds are successfully transferred , and confirmation is displayed.

1. Insufficient funds

* User tries to transfer and amount higher than the available balance
* Expected Result: System displayed error messages regarding insufficient finds.

1. Incorrect Recipient details

* User tries to transfer and amount to incorrect recipient
* Expected Result: System displayed error messages regarding incorrect recipient

Ecommerce Platform

Use case: Add to cart

Description: Users can add items to their shopping cart for future purchase

Main flow

Alternative flow

Online shopping website:

Use case: User registration

Description: The system allow users to create accounts to make purchase

Main Flow:

1. User navigates to the registration page
2. User enters valid details (email, password etc.)
3. User submit the registration form
4. System registers the user and confirms successful registration

Alternative flow:

If the user enters invalid email format, the system displays an error message

Test case1: Valid user registration

1. Navigate to the registration page
2. Enter valid details (name, email, password)
3. Submit registration form

Expected result:

User successfully registered to system and user receives a confirmation email

Test case2: Invalid email format

1. Navigate to the registration page
2. Enter invalid email format ( e.g. missing ‘@’ symbol)
3. Submit registration form

Expected result:

System displays an error message indicating the invalid email format

User is prompted to correct the email format

Test case3: Password strength validation

1. Navigate to the registration page
2. Enter a weak password ( e.g. less than 8 characters)
3. Submit registration form

Expected result:

* System displays and error message indicating the password is weak
* User is prompt to enter a strong password

Test case: Mandatory field missing

Test case: duplicate email

Test case: Invalid data in form

Test case: Alternative phone no

Test case: Invalid captcha

**White box testing**

Statement coverage

Branch Coverage

Path coverage

**Statement coverage**

def add\_number(a, b):

    result = a+b

    return result

# Test case 1 :Positive numbers

addition = add\_number(5, 3)

# Test case 2: Negavite numbers

addition = add\_number(-3, -5)

# Test case 3: Zero and positive number

addition = add\_number(0, 7)

# ==================================

def check\_number(num):

    if num > 0:

        print("Positive Number")

    else:

        print("Non positive number")

# Test cases 1 : Positive number

check\_number(5)

# Test case 2 : zero

check\_number(0)

# else part and print non positive number

# Test case 3:

check\_number(-5)

# else part and print non positive number

# =======================================

def absolute\_difference(a,b):

    if a>b:

        result= a-b

    else:

        result = b-a

    return result

# Test case1 : Positive numbers

absolute\_difference(10,20)

# Else statement is executed

# TEst case 2: Negative number

absolute\_difference(-3,-5)

# If statement will be executed

# Test case3: Equal number

absolute\_difference(10,10)

# else part will be executed

# =======================

# DECISION COVERAGE

def calculate\_grade(score):

    if score>= 90:

        print("A")

    elif score >=80:

        print("B")

    elif score >= 70:

        print("c")

    else:

        print("invalid grade")

# Test case 1: Grade A

calculate\_grade(99)

# Test case 2: Grade B

calculate\_grade(85)

# Test case 3 : Grade C

calculate\_grade(75)

# Test case 4: invalid grade

calculate\_grade(10)

# =================================

def even\_or\_odd(num):

    if num%2== 0:

        print("Even")

    else:

        print("odd")

# Test case 1 : odd number

even\_or\_odd(21)

# Test case 2 : Even number

even\_or\_odd (100)

Coverage Matrices

**JaCOCO ( Java code coverage)**

Integrated with maven Gradle build tools

Coverage.py ( Python code coverage)

Emma ( Java code coverage)

LCov( Linux code coverage)- C and C++

Cobertura

Dot net / Java

Istanbul ( Java script)

Cyclomatic complexity

E-N+2P

E : Number of edges in the program control flow graph

N: Number of nodes in the control flow graph

P: Number of connected components

def check\_number(num):

    if num>0:

        print("positive number")

    else:

        print("Non Positive number")

#        Entry

#          |

#          |

#          |

#      [Decision]

#     /          \

# print          print

# positive       negative

#     \              /

#           Exit

# N= Node

# N = 1 entry + 1 decision + 2 outputs = 4

# E = 1 entry to decision + 2 decision + 1 = 4

# P= 1

# Complexity = E-N +2P = 4-4+2\*1 = 2

# ==========================

def square\_number(num):

    result = num\*\*2

    return result

#    (Entry)

#       |

#       |

#    [return]

# V = E-N+2P

# v= 1-2+2= 1

# ============================

def max\_number(a,b):

    if a>b:

        return a

    else:

        return b

#         (Entry)

#           |

#           |

#           |

#         [Decision]

#         /  \

#        /    \

#     Return  Return

#     a          b

# Node:

# Entry node (1)

# Decision node(1)

# Two Return Nodes( 2)

# Edges:

# Entry to decision

# Decision to returns(2)

# Decision Exit (1)

# V= E-N+2P

# V= 4- 4+2

# V= 2

**TEST STRATEGY DOCUMENT**

1. Introduction

* Purpose : Define the purpose and scope of the test strategy document
* Objective: Specify the goals and objective of the website testing
* Scope: Clarify the functionality browsers devices and environments to be covered.

1. **Testing approach**

* **Testing types:** Specify the testing types to be employed( functional, usability, performance , security etc.)
* **Methodology:** Define the methodology ( manual, automated, exploratory) to be used
* **Testing levels:** Outline the different testing levels( unit, feature testing, integration test, system test, user acceptance test)

1. **Test Environment:**

* **Browser and devices:** list the browsers (Chrome, firefox, safari, etc) and devices (desktop, mobile, tablets)
* **Operating system:** Windows, macOS , iOS, Android to be covered

1. **Test scenarios and Test cases**
2. **Testing tools**

Test management tools – JIRA, TestRail

Automation tool: Selenium, Jmeter, Playwright, cypress

1. Test Schedule:
2. Risk and Contingency plans:

Identify the risk: List potential risk in testing (time constraint, resource availability etc)

Mitigation plan: how to mitigate identified risk

1. Performance and load testing

Performance Metrics: Define performance metrics (response time, concurrent users)

Load Testing: load testing approach you are going to follow

1. Usability testing:

User experience: outline the usability testing approach to ensure the user-friendly user interface

User feedback: include the plan to collect the user feedback

1. Security testing:

Security measures: Details the approach for testing website security (Penetration testing, vulnerability testing)

1. Acceptance criteria

Exit criteria: Define the condition to be met before concluding testing.

Acceptance criteria: specify the criteria for acceptance or rejection the website after testing

Conclusion:

Summary: Summarize the key points covered in the test strategy document

Approval: include stakeholder to approve

Focus:

Path coverage ensure every possible path from start to end execu

|  |  |  |
| --- | --- | --- |
| Aspect | Path coverage | Branch coverage |
| Focus | Path from start to end is executed | On exercising all possible branches in the code |
| Objective | All execution path | Ensure both true and false outcome of a decision point is tested |
| Granularity | Deep dive in the loops and each statement | Decision points ( true and false is tested) |
|  |  |  |
|  |  |  |

ted

Br

Test Strategy

Test Plan

Into

Focus :

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Test Strategy** | **Test plan** |
| Focus | High level approach and objectives for testing | Detailed description of testing activities and schedule |
| Scope | Defines the overall testing approach for the project | Details specific testing tasks and timelines for a particular phase of level |
| Objective | Outlines the testing approach, methodologies and goals | Specific detailed testing activities, resource and schedule |
| Conent | Covers the testing approach, types , levels of testing, tools and environment | Test cases, test scenarios , timelines |
| Level of detail | Less detailed strategic overview | Highly detailed , specific, operation level document for execution |
| Timing | Created before the test plan and sets the direction | After test strategy, |
| Components | Objective, scope, methodologies high level timelines | Test cases, test scenarios , timelines |
| Change frequency | Changes infrequently , serve as a reference throughout the project | Subject to change , updated more frequently based on the project need |

**Test plan - Website**

**Introduction:**

**Purpose:** Define the purpose and objective of the test plan

Scope: Clarify the functionalities, browsers, devices and environment to be covered.

References:

**Test Environment:**

Browser and Devices: list supported browsers ( Chrome, ff, safari, edge) and devices ( Desktop, mobile , tablets)

Operating systems: Windows, macOS, iOS, Android

**Test Scenarios and Test Cases**

**Functional Testing scenario:** Detail scenario covering main functionality (navigation, forms, etc)

**Test Cases:** Provide detailed test cases for each scenario, including expected out comes and precondition.

**Testing Tools**

**Test Case management tool: Specify the tool used for test case management (JIRA, Test Rail, etc)**

**Bug Management:** which tool you will use for bug management (JIRA)

**Automation tools : Selenium, cypress, etc.**

**Test Execution**

**Test Execution Plan: define the sequence of test execution (smoke, regression ..)**

**Reporting:** Document /outline how the results will be documented, defect reporting

**Performance and Load Testing:**

Performance test metrics: page load time, server response time etc.

Load Test plan: Load test approach, and scenarios (simultaneous user)

**Compatibility testing:**

Cross browser testing: Ensure all compatible browser is tested

Device Testing: various device and screen size

**Security Testing**

Security measures: detailed approach for sec testing (SSL, Data protection)

Compliance Check:

**User Acceptance Testing:**

**UAT Plan:** Detail user acceptance testing approach

User feedback: surveys, feedback

**Risk Assessments and Contingency Plans**

Identify the risks

Mitigation plan

Effort estimation / schedule

1. Test Scenario Identification
2. Test Case creation
3. Test Data preparation

* Input values
* Expected outputs
* Any specific data condition required for testing

1. Test Coverage analysis
2. Test case review

Test scenario:

Mobile

Call

Scenario:

1. Make a call to a contact
2. Receive a call from unknown number
3. Make a call to a number
4. Reject a call

Write the test scenarios for Pen

**STLC**

1. Test strategy
2. Test plan
3. Test case development/Design
4. Test Execution
5. Test closure activity

**Test Design Phases**

1. Test scenario identification:
2. Test case creation
   1. Precondition
   2. Steps to execute
   3. Expected result
   4. Post cleanup
3. Test data preparation
   1. Input values
   2. Out put values
4. Test coverage analysis
   1. Ensure all important aspects of the software system are covered
   2. Identify the gap in the test coverage and do the necessary changes to achieve comprehensive testing.
5. Test case review:
   1. The created test cases are reviewed by peers or senior testers to ensure their accuracy, completeness and effectiveness.

**Software Test Execution**

1. **Test environment setup**
2. **Test case execution**
3. **Defect Reporting**
4. **Test Result Documentation**
5. **Test logs and Artifacts**

**Software test closure activities**

Test Result analysis:

1. Analyse the overall quality of software
2. Bug pattern

Defect Analysis and closure

1. Review the open bug

Test closure report

Documentation and achieving

Stakeholder communication

**Login Functionality**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test case Id** | **Test Objective** | **Pre condition** | **Steps** | **Expected result** | **Post condition** | **Actual result** | **Test result** |
| TC\_Login\_001 | To verify the successful login in to the website | Valid user credentials | Open the chrome browser, Type the website name and press enter | The home page is launched | Logout from the web account |  | Pass |
| Enter the valid user’s name | User name is entered |  |  | **Fail** |
| Enter the corresponding password | The password is in encrypted form (\*\*\*\*\*) |  |  |  |
| Click on signin button | The user is redirected to the home page of the website after successful login.  Personalized welcome message is displayed |  |  |  |

**Create a BUG**

**New**

**This is** initial stage when a bug created.

TP-046

TP-65

Open – bug will assigned to appropriate Developer or tester

In Progress

Developer will analyze the bug, and develop a solution

Fixed:

Developer will do the code changes and fix the problem and mark the status as fixed.

Retest

The bug is marked as retest while awaiting the verification

Verified:

Fix has been verified and he bug no longer exists

Bug

**Description**

Incorrect error message when we log in with an invalid password and correct user name

**Prerequisite:**

Username: admin

Password: admin1234

**Steps to Reproduce:**

1. Open the website xyz.com
2. Enter the login ID as admin
3. Enter the password as admin1234 or any invalid password
4. Click on the login button

**Expected Result:**

The error message “ Enter correct credentials “ should be displayed

**Actual Result:**

The error message “ Incorrect password “ is displayed

**Reproducibility**: 10/10

**Logs**: Attached as lssue.log

**Screenshot**: Attached as issue.jpeg

Reference: HLD, LLD, SRS

Severity:

The impact of bug on the product

Critical :

System crashes, data corruption , or loss of essential functionality

High Severity

Medium

Low:

Application ---- Aplication

Priority – all about scheduling and timings

High:

Need immediate fix

Medium:

Fix in next build

Low priority

High Severity and low priority bug

Banking application

Bank gives the interest quarter

4% interest rate

Developer made a mistake

Interest 40%

This code will be executed at the end of quarter

After 3 month

Priority medium

Low severity and high priority bug

Developed a website for google

Low severity

Priorityhigh

**Software Test Metrics**

Test coverage metrics:

Defect Metrics

Test execution Metrics

Test effectiveness metrics

Test efficiency metrics

Test environment metrics

**Software Requirement Traceability Matrices (RTM)**

**Software Requirement Traceability Matrices (RTM)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **BRS** | **SRS** | **HLD** | **LLD** | **Test case** |
| **BRS\_01** | **SRS\_01** | **HLD\_05** | **LLD\_05** | **TC\_01, TC\_05, TC\_08** |
| **BRS\_02** | **SRS\_05** | **HLD\_08** | **LLD\_10** | **TC\_011, TC\_015, TC\_018** |
| **BRS\_02** | **SRS\_05** | **HLD\_08** | **LLD\_10** | **TC\_011, TC\_015, TC\_018** |
|  |  |  |  |  |
|  |  |  |  |  |
| **BRS\_021** | **SRS\_05** | **HLD\_018** | **LLD\_10** |  |
|  |  |  |  |  |

**Web Testing**

**Functional testing- Link testing**

1. Link verification
2. Click ability
3. Target window/tab
4. Internal and external link
5. Broken link

**Functional testing- Testing Forms**

1. **Form submission**
2. **Input validation**
3. **Error handling**
4. **Accessibility**
5. **Autocomplete and suggestions**
6. **Form Reset**
7. **Cross browser and cross device compatibility**
8. **Data security**

Website business cycle

Web Testing : Compliance

W3C (Workd Wide web Consortium)

**HTML Validation**

**CSS Validation**

**JavaSCript Compliance**

**Accessibility testing (WCAG)**

**API Testing**

**Usability Testing**

**Interface testing**

**Database Testing**

**Non Functional testing:**

* **Performance**
* **Security Testing**

**Challenges**

**Test environment setup**

**Scalability and performance testing**

**Compatibility testing**

**Best Practices:**

**Comprehensive Test planning**

**Realisitc test environment**

**Performance testing**

**Compatibility testing**

**Usability testing- involve real user**

**Database testing**

**Objects in database**

**Table**

**Views**

**Triggers**

**Functions**

**Data integrity**

**Primrary key**

**Foreign key**

**Unique constraints**

**Check constraints**

**Cloud Testing**

**Infrastrucature as a service**

**On demand resource allocation**

**Scalability and Elasticity**

**Cost optimization**

**Service virtualization**