Checklist for Test Case preparation

Kishore Chavali

Introduction

The following is a quick checklist to verify that all possible test cases are identified during test plan/test case preparation.

The objective of this checklist is to provide a guideline in preparing test cases

Item for identifying more test cases
For each input to the system identify valid values • Identify file inputs • Identify user data inputs • Identify system inputs
For each input to the system identify invalid values
For each input to the system identify boundary values to be tests (equivalence call partitioning)
Get list of error messages that system will give from development team. Each error message should have at least one test case. • This keeps growing. If possible trace the error number with test cases
For each output define expected outcome – each output condition becomes a test case • Identify various file outputs user will notice • Success and Error Path outputs • Database outputs if any
Check for duplicate values for all inputs
Check for deletion conditions of objects created during tests and related effects on the system.
Check for update conditions and related effects on the system
User Interface related bugs (aesthetics/logical grouping of inputs, typos etc)
If file based system: small file, large files, corrupt files, invalid files as input
What condition can cause runaway or loop, overflow situations?
Perform abnormal actions or sequence of actions
Test with default values of the system
Change all default values used by system one by one and test changing behavior.
Write User Scenario cases: Administrator tasks, Designer tasks, Operator Tasks and what each user expects
Combinations of integration systems (third party version with which product integrates)
Combinations of database or file systems used (helps identify certification matrix)
Combinations of supported development environments (like Java version)
OS specific cases
Test with multiple user accounts and login as different users at different times
Concurrent usage scenarios if applicable
Performance boundaries -> Which variable effect performance of the product
How can we verify accuracy or consistency of the system? (Is client and server compatible, is repository data consistent etc.)
Are there explicit date conditions? Current date, future date, invalid dates, range of dates, expiry dates
What can cause corrupt inputs and how does system respond?

References:

1. Testing Computer Software: Cem Kaner, Jack Falk, Hung Nguyen