Developing a Test Plan for Student Registration System for a University

1. Test Plan Identifier

Some numbers are generated to identify this test plan and the levels related to this.

Actually the number may identify if the test plan is a Master plan, a level plan or whichever plan it represents. This is to assist in coordinating software and test ware versions within configuration management.

We know that test plan is basically a software documentation, they are dynamic in nature and must be kept up to date. Therefore, they will have revision numbers.

2. REFERENCES

List all documents that support this test plan. Refer to the actual version/release number of the document as stored in the configuration management system. Try not to copy text from other documents.

Documents that could be referenced included below:

(a)Project Plan

(b)Requirements specifications

(c)Development and Test process standards

(d)Methodology guidelines and examples

3. INTRODUCTION

In this part the purpose of the plan should be defined, possibly identifying the level of the plan.

It is the executive summery part of the test plan. You may want to include any references to other plans, documents or items that contain information relevant to this project/process. If preferable, you can create a references section to contain all reference documents.

Scope of the plan should be identified related to the Software Project plan which is relates to. Other items may include, resource and budget constraints, scope of the testing effort, how testing relates to other evaluation activities (Analysis & Reviews), and possible the process to be used for change control and communication and coordination of key activities.

4. Test Items

Thing we are going to do within the scope of the test plan. Essentially, something you will test, a list of what is to be tested. This can be developed from the software application inventories.

This can be controlled by the management. This information includes version numbers, configuration requirements where needed. It may also include key delivery schedule issues for critical elements. Remember, what we are testing is what we intend to deliver to the Client.

5. SOFTWARE RISK ISSUES

For the first time it’s very important to find out which software is to be tasted and what are the main requirement, like:

1. Security: High security must be given in this software.

2. Several interface: It is need to make several interface

3. Legible for the user: Make more easier for client

4. Ability to catch the new update (if need or if software update in future)

5. Having the system to change the request: If client request system will be change

6. Varsity regulations and rules: This software maintain some rules to varsity

Sometimes it’s a big problem that is user can’t understand the real usability of the software .so isn’t very important to realize or understand the requirement of the software .Sometimes some bugs or defect are solved by unit testing but if the problem is big or so many problem r gather together it’s hard to fix it.

6. FEATURES TO BE TESTED

In this software it’s very necessary to test ever step carefully like log in is the first step to test that is its works well or not. Then it will be good to test every pages from first to last be it working properly.

After That it is essential to test if user give any order is it works or not then is it put the record or not .

Another side one of the important thing is that is the system fast or not or is there any bugs or not.

Lastly another important things is to test the log out option carefully if it not works properly then user can’t feel good to use it .So it’s important to test those part properly before deliver it to the client .

7. FEATURES NOT TO BE TESTED

Sometimes it is not necessary to test every steps .Like in this software it is not essential to look that is the software’s every spelling is right or not .Not so essential to check the software running time or rate . Nothing is important to test any disadvantage about this Software.

8. Approach:

Approach outlines the testing process to be applied and can be considered to have six steps as illustrated in the diagram below. The core four steps are: Develop Tests, Prepare to Test, Run Tests and Review Test Results. These four steps are controlled by Plan Testing and Change Management.

- **Analyze Requirements** - Means understanding them and looking for what is missing and inconsistent from what is actually requires.
- **Develop Scenarios** - Means the preparation of scenarios that use the system, using techniques such as Use Case.

- **Derive Acceptance Criteria** - Once the previous two activities are underway or completed then the set of questions to be asked about the system to see if it matches the capability needed are prepared.
- **Construct Test Cases** - Test Cases are the set of specific inputs and expected results which enable one or more Acceptance Criteria to be proved.
- **Write Test Scripts** - Test scripts are the operational instructions for running Test Cases including: what has to be done to the system, what has to be measured, and how to do the measurement.
- **Review Documents** - This is a key quality process of checking all documentation produced during the development of the system.

Specify if there are special requirements for the testing.

Only the full component will be tested.
A specified segment of grouping of features/components must be tested together.

Other information that may be useful in setting the approach are:

MTBF, Mean Time between Failures - if this is a valid measurement for the test involved and if the data is available.

SRE, Software Reliability Engineering - if this methodology is in use and if the information is available.

9. ITEM PASS/FAIL CRITERIA:
Item Pass/Fail Criteria" section deals with defining when an item has passed or failed. This is not the place to define the detailed pass criteria for each feature, but to describe the process and overall standards for evaluating the test results.
There are various approaches that are used in evaluating test items including the mechanical one of passing or failing a test item depending on whether a predetermined number of incidents of certain types have been found.

- All test cases completed.
- A specified percentage of cases completed with a percentage containing some number
- Of minor defects.
- Code coverage tool indicates all code covered. A defect is something that may cause a failure, and may be acceptable to leave in the application.
- A failure is the result of a defect as seen by the User, the system crashes, etc.

Although a common method it is deeply flawed, especially for UAT. When assessing the system for use in an organization you are trying to see if it can deliver value to that organization and not if it works perfectly. The reality is that all computer systems have faults. What you are evaluating is if you can live with those faults

10. TEST DELIVERABLES:

- Test Plan - This document deals with what needs to be done in UAT.
- Designs - The UAT Acceptance Criteria.
- Test Cases - The values input and results expected from tests.

- Test Item Transmittal Reports - Developers handover report.
- Test Logs - The results of running the tests.
- Incident Reports - Observations of unexpected results.
- Incident Report Logs - Summary of Incident Reports.
- Test Summary Report - Summary of testing.
- The test data.

11. Remaining test tasks:

If this is a multi-phase process or if the application is to be released in increments there may be parts of the application that this plan does not address. These areas need to be identified to avoid any confusion should defects be reported back on those future functions. This will also allow the users and testers to avoid incomplete functions and prevent waste of resources chasing non-defects.

- Create Acceptance Test Plan.
- Create System/Integration Test Plan.
- Define Unit Test rules and Procedures.
- Define Turnover procedures for each level.
- Verify prototypes of Screens & reports.