Test Plan

Overview

Unit testing mainly relies on automation framework while smoking test which is embedded in the unit testing mainly relies on manual methods.

Integration testing should be executed both manually and automatedly.

System testing mainly relies on manual methods.

UAT testing and beta testing should be mainly executed manually.

Regression testing should be executed both automatically and manually.

Tips: All the scripts should be designed by experienced testing engineers.

Recommended Testing tools:

Visual studio 2015 internal tools

      DbUnit

DataFactory

1 Unit Testing

Dispatch one member from each testing team (5 persons in total) and embed them into the development team of the five functions to do the smoking test during the development process. Let the programmers get quicker feedback when ensuring the efficiency of the formal testing.

Use more data-driven scripts in testing the function of employee data and job data while using more key-word driven in testing the function of recruiting, employee benefits and payroll.

Each testing group should dispatch a representative to use external files like spreadsheets to drive the data-driven and key-word driven scripts. Meet once a week to communicate the defects having been found. The member embedded in the development team should report the results of smoking test every day to the team to which he belongs.

Tips: All of the five functions should be tested using internal automation testing tools in Visual Studio 2015. Considering the importance of unit testing in the maintenance process, make sure that at least 4 staff in each testing team participate in it (including the one who is embedded in the development team).

2 Integration Testing

2.1 Component Integration Testing

There are five different major functions in this application which are Recruiting, Employee data, Job function data, Employee benefits and Payroll.

With the data of these five aspects exchanged between the 5 units, what we do here in component testing is to check the handling of data passing. Basically, once the data is generated by the 5 units, its range and data types should be checked to ensure nothing wrong before it goes into exchange. Five functions mean five types of data.

The information about an employee’s name, when she/he is recruited and the employer’s name and location should be included in recruiting data. Employee data contains an employee’s basic information such as name, birth date, social security number, race, review history, etc. Job function data should contain detailed requirements and offerings of specific jobs, e.g. salaries. Employee benefits should include health benefits such as 401K as well as employees' compensation benefits. Payroll should be listed on a weekly/monthly/yearly basis with the income calculated after tax based on the according state’s regulations.

To verify the data correctness, a data-driven type of automation testing should be employed here to ensure each data correctness and ready to be called. Since we focus on data here, the test should be initialized when the unit testing is completed and the basic functions of each unit are tested and ready to go.

The method we employ here is to isolate the tested data from the rest of the system. In such case the missing functions can be replaced by stubs and drivers to simulate the interface between the software components in a simple manner. Thus, we are able to keep a separate log file of data items being passed. In this way, we can call data from each unit and see if its values are in right format and range. Tests can include checking the handling of some extreme data values while other interface variables are passed as normal values. Unusual data values in an interface can help explain unexpected performance in the next unit. External values like most popular jobs’ function data which benefits plan with most numbers of clauses, should all be tested to see if the data can be accessed correctly.

2.2 System Integration Testing

Use top-down testing strategy in system-integration testing to avoid useless work at the following testing. If there are defects, stop testing immediately and communicate with the team leader and the programmer first to find out the mistake.

Pay more attention to data communication in the interaction of job data and employee data.

Tips: Each team should dispatch one or two members to be in charge of the testing drivers and stubs preparation. In the integration testing process, the leader of each team should have a short meeting with each other every day to communicate the defects and report them to the development team. Use black box testing baseline usually. Once discovering the defects of the interaction of different function testing, the process should turn into white box testing.

3 System Testing

3.1 Migration Testing

Use different internet browsers, such as Chrome, Firefox and IE, in different operating systems including IOS, Android and Windows separately to the whole system to see whether these five functions could work properly.

3.2 Hardware Testing

Use different computers (ThinkPad, Apple Mac, Lenovo, etc.), different mobile phones (Samsung, apple, etc.), and various tablets (iPad, Surface Pro, etc.) to see whether the application is compatible with different hardware.

3.3 Functional Testing

Examine whether each sub-function in the five functions is consistent with the application requirement documents and whether there are any omissions.

Tips: All of the migration testing, hardware testing and functional testing should be tested manually and immediately communicated with the QA manager and development team once errors are found.

3.4 Load Testing

Simulate the normal flow rate of the clients in the cloud server to see whether the five functionalities could execute fluently.

3.5 Stress Testing

Simulate really high flow rate of the clients in the server to see what would happen.

3.6 Recovery Testing

After load testing and stress testing, we could know the stress boundary of the cloud server and the database. Then make it a failure and give it back to the development team to see how to solve this problem.

(1) While an application is running, suddenly restart the computer, and afterwards check the validness of the application's data integrity.

(2) While an application is receiving data from a [network](https://en.wikipedia.org/wiki/Computer_network), unplug the connecting cable. After some time, plug the cable back in and analyze the application's ability to continue receiving data from the point at which the network connection disappeared.

(3) Restart the system while a [browser](https://en.wikipedia.org/wiki/Web_browser) has a definite number of sessions. Afterwards, check that the [browser](https://en.wikipedia.org/wiki/Web_browser) is able to recover all of them.

This process should be carried out automatically.

Automation tools recommended: Load impact.

3.6 Data Backup

Each team should dispatch one member to be in charge of the testing data backup and use a specific confidential computer in case unexpected events happen especially when testing the database and cloud server.

3.7 Database Test

We need to test whether our database meets the functional requirements and the stress level by using four automation tools listed below:

(1) DbUnit

We can use DbUnit to test whether the input and output of our database (Employee data, Job data, etc.) are true. DbUnit is a JUnit extension targeted at database-driven projects that, among other things, put your database into a known state between test runs. This is an excellent way to avoid the myriad of problems that can occur when one test case corrupts the database and causes subsequent tests to fail or exacerbate the damage.

(2) QuickTest Professional (QTP)

We can simulate the user's operating procedures by QTP. We can use the verification method in QTP and the way of monitoring the performance of data to test our database. It is a gray box method.

(3) DataFactory

This is a black box method. This tool helps us create database with any structure. By creating a large amount of data which we want to fill in our database, we can test whether the function of our database meets the standards.

(4) Loadrunner

Loadrunner is used to test applications and to measure system behavior and performance under load. LoadRunner can simulate thousands of users concurrently using application software, recording and later analyzing the performance of key components of the application.

4. Alpha Testing

After all the above testing steps are done, we should focus on the eventual step before the software release to the public, which is of often called Alpha Testing. We are not going to use the same engineers during the previous step. Instead, a new small QA team (3 people) is needed. Finding plenty of bugs, crashes, missing docs and features are expected in Alpha testing. There are two important phases we will go through in the Alpha testing：

(1) By using any automation tools (either debugger software like Jdb or hardware-assisted debuggers), improve the quality of the product and ensure beta readiness. Find the underlying bugs and the omitted bugs that still alive after the above processes. Guarantee to catch bugs as soon as possible.

(2) Test the stability and adaptability of the software going through the intended environment. IOS, Android and Windows will be tested in this process. Three engineers in the QA team will separate to choose one of these environments and the Alpha Testing will take 2 weeks during the whole testing plan.

5. User Acceptance Testing

User Acceptance Testing is mostly focus on a validate type and concern about the requirement of the software. Testers imitate themselves as customers and are going to test the achievement of the Major functions of the application.

(1) To test the aesthetic measure about the software, whether the design is humanization and suitable for the most of the customers’ flavor.

(2) Main functions will be tested in these processes. Does the recruiting serve conveniently? Is the employee data complete or not? Are there clear classifications of the Job function and the Payroll? Is there a comprehensive exhibition of the Employees benefits? Does the software require a complex operation? Is it easy to use across all ages?

The UAT will also be implemented by the QA team and it will take 4 weeks to finish the all test objects.

6. Beta Testing

Beta Testing is a pre-release testing. We plan to use the actual users to test the products. Conduct the test in the real world with the real customs and make the feedback covering every element of the product. Testers should report the issues and the missing data of beta software. Does each of information display in the software? How long will it appear? Any underlying bugs during the software operating process? Does the connection work well between the server and the software so that the response of the requirement returns quickly?

Testers in the Beta Testing will be the real users. We may use 20 users to find different types of issues and make feedbacks return to the developers. The whole Beta Testing will be a 4 week process.

7. Regression Testing

We use automation tools in regression testing. Regression test cases need to be selected very carefully so that maximum functionality is covered in minimum set of test cases. These sets of test cases need continuous improvements for newly added functionality. In such cases, selective tests needs to be executed in order to save not only the cost but time of the testing. These selective test cases are picked based on the enhancements done to the system and parts where it can affect the most.

Efforts the team should do in regression testing: (1) Run automated test cases every day in the evening so that any regression side effects can be fixed in next days’ build. We can use QTP as our automated tools. (2) Compare current results with previously executed test results.

Recommendations on regression testing: The scope of our regression testing should cover each unit and each function of our system when programmers fix any bugs or add new codes for new functionality in our application. We need to focus more on the unit testing in that if the bugs need to be found and fixed at an early period, or we have to spend much more money and time on them. Data protection is also an important part we should focus in this part.