# CHAPTER SIX

# TESTING

## 6.1: Testing in Software Engineering

Testing is an activity that is used to discover errors and correct them, so that we are able to create a defect-free product for our customer or user. Testing is an important phase in the software development life cycle. The objective of testing is to evaluate if we have created the system correctly. During the earlier stages, the focus was to check what is being built but in testing when we have the end product ready, our focus shifts to validate whether the product that has been built has been built correctly or not. Hence, the focus shifts from building the product right to building the right product. [1]

There are two basic types of software testing, which are black box testing and white box testing. General testing process for large system development starts with the testing of individual program units such as functions, classes or objects. These are then integrated into sub-system and systems, and the interactions of these units were tested. Finally after delivery of the system, the customer may carry out a series of acceptance tests to check that the system performs as specified. [2]

Whereas, for smaller system or for system that are developed through scripting or reuse, there are often fewer distinct stages in the process.

The two fundamental testing activities are component testing, testing the parts of the system – and system testing, testing the system as a whole. [2]

## 6.2: Goals and Types of Testing

Basically, there are two distinct goals of the software testing process:

* To demonstrate to the developer and the customer that the software meets its requirements.
* To discover faults or defects in the software where the behavior of the software is incorrect, undesirable or does not conform to its specification.

The first goal, where you expect the system to perform correctly using a given set of test cases that reflect the systems expected use, leads to validation testing. The second goal leads to defect testing, where the test cases are designed to expose defects. The main types of testing approaches are defined below: [2]

### 6.2.1: System Testing

System testing involves integrating two or more components that implement system functions or features and then testing this integrated system. For most complex systems, there are two distinct phases to system testing – Integration Testing and Release Testing. As for the Web Based Payroll System, the system had to go through both Integration and Release testing.

#### 6.2.1.1: Integration Testing

Integration testing is mostly concerned with finding defects in the system, where the test team has access to the source code of the system. If the problem is discovered, the team goes through the source code to find the components that have to be debugged. [2]

Integration testing was done after every unit or feature being added to the system. For example, if the current has three features (show payroll chart, overtime chart, total hours worked chart), when the fourth feature, unit, or component if being attached or added to the system, integration testing had to be done throughout although there are still many units or components to be added to the system.

#### 6.2.1.2: Release Testing

In release testing that version of the system is tested that could be released to users or customers. The test team here validates if the system meets its requirements and also ensures system dependability. It is usually black-box testing where the test team is simply concerned with demonstrating the system does or does not work properly. If problems are discovered then they are reported to the development team whose job is to debug the program. Acceptance Testing is key aspect of release testing, where the customers or users are involved in release testing. If the release is good enough, the customer or user may then accept it for use.

After all the units were integrated and combined together to form a complete system, a release test was ran to make sure that system’s components are not affecting the other components after integrating them. Few feedbacks from the users were collected, according to the feedback the system has to go through the debugging and testing again. [2]

### 6.2.2: Component Testing

Also known as Unit Testing is the process of testing individual components in the system, to expose faults in these components, and the software developers are responsible for this testing. There are different types of component that may be tested at this stage:

1. Individual functions or methods within an object
2. Object classes that have several attributes and methods
3. Composite components made up of several different objects or function.

These composite components have a defined interface that is used to access their functionality.

The Web Based Payroll System was created in components and units. A new Rails project was created in such a way that each model, view, and controller are organized in separate folder. Below is the screenshot of the project structure.



Figure x: The main project folder



Figure x: The application folder

Figure above shows the units of the Web Based Payroll System which later were integrated as a whole and was tested using the system testing method.

In Rails, there are three different kinds of tests that can be written, which are unit, functional, and integration test.

* Unit testing tests the models.
* Functional testing tests the controllers.
* Integration testing tests at a high level through multiple controllers.

Table y: Five levels of testing



## 6.3: Black box testing

Black box testing, also known as functional testing and behavioral testing, focuses on determining whether or not a program does what it is supposed to do based on its functional requirements. Black box testing attempts to find errors in the external behavior of the code in the following categories (1) incorrect or missing functionality; (2) interface errors; (3) errors in data structures used by interfaces; (4) behavior or performance errors; and (5) initialization and termination errors. Through this testing, we can determine if the functions appear to work according to specifications. However, it is important to note that no amount of testing can unequivocally demonstrate the absence of errors and defects the code.



Figure x: Black Box testing

*http://2.bp.blogspot.com/\_fOOSCGT3XIw/Sa4cVsX7T7I/AAAAAAAAAAM/i\_YOWD6xHIQ/s320/BlackBoxTesting.gif*

## 6.4: Test Cases

A test case in software engineering is a set of conditions or variables under which a tester will determine whether an application or software system is working correctly or not.

The format of the test case design is very important. I will use a particular format for the test cases, as shown in Table.

Table y: Test Case Planning Format



Table y: Test cases for Web Based Payroll System

|  |  |  |  |
| --- | --- | --- | --- |
| Test ID | Description | Expected Results | Actual Results |
| 1 | Admin user entered the username and password.  Test case 1 has successfully completed |  | Admin page appeared.  **Test case passed** |
| 2 | User entered the username and password. Test case 1 has successfully completed | User should be redirected to the home page, from where he/she started (Test case 1) | User redirected back to the home page. **Test case passed** |
| 3 | User pressed the “I have signed in, take me to the home page” link. | User must be redirected to the user’s home page, also called Wall, and get the friend’s posts in a listbox. | User’s home page appeared with the friend’s, liked pages’, and groups’ posts in a listbox (Wall).  **Test case passed** |
| 4 | User has selected YouTube video post item, (using tab and arrow keys from the keyboard) from the wall’s listbox and pressed enter | User will be redirected to the post page where the video will be played and the information, links of the video will appear in the form of text and hyperlinks | Page with video and the associated information is appeared.  **Test case passed** |
| 5 | User navigated to the “Friends” webpage by pressing on the “Friends” link from the home page. | List of first hundred,  0 – 100, friends will appear on the “Friends” page | List of hundred friends appeared in less than four seconds time.  **Test case passed** |
| 6 | After the completion of test case 5, user clicked on the “Next 100 friends” link on the “Friends” page | List will reload the next 100 friends upon the page postback | List has successfully reloaded with 100 friends, range from 100 – 200.  **Test case passed** |
| 7 | User has continuously pressed the “Next 100 friends” button to the point where there is no more friends to be shown in the list, or a user does not have a friend added at all and he navigated to the “Friends” page | List of friends will not add any item in it and a label above the list will notify the user that he/she has “0” friends to be added in the list | List appears empty with the label “0 friends”, and JAWS read that correctly.  **Test case passed** |
| 8 | User has selected an item from the list of 100 friends and pressed enter. | Three link buttons will appear on the right side of the friends list, link to the profile of the selected user, link to the photo albums of the selected user, link to post the text on the selected friend’s wall | Links appeared in less than one second.  **Test case passed** |
| 9 | User has navigated to the to the “Photos” web page, selected an album from his/her own albums using keyboard keys, and pressed enter. | The information (caption, index, tags, comments etc) of the very first photo in the album will be added in the photos listbox as items, photo index as item 0, photo caption as item 1, photo link as item 2, and so on. Pressing the “next” button is required to go to the next photo. | The information  associated to the photo has been added to the photos listbox successfully  **Test case passed** |
| 10 | User has navigated to the “Inbox” page to check the inbox send emails. | List of twenty inbox emails will appear at first. List of sent emails will be added upon the press of “Next” button | “User does not have the permission to this feature” error occurred.  **Test case failed** |

X test cases were developed to accomplish the Black Box testing for the Web Based Payroll System.