# 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 1: The main project folder



Figure 2: 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 (the command to run the unit test is **rake test:units**).



Figure 3: Running unit test in Rails

* Functional testing tests the controllers (the command to run the functional test is **rake test:functionals**).



Figure 4: Running functional test in Rails

* Integration testing tests at a high level through multiple controllers (the command to run the integration test is **ruby –Itest test/integration/test\_filename.rb**).

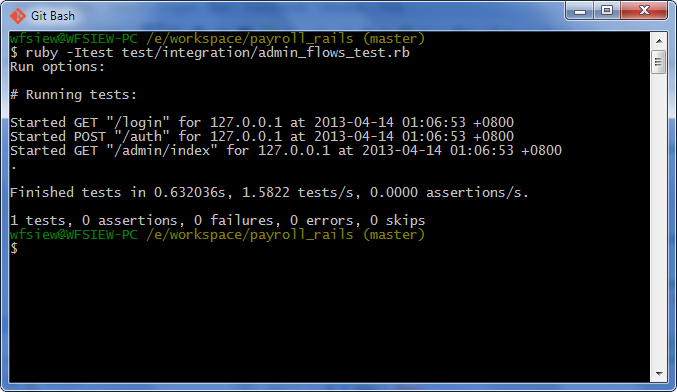


Figure 5: Running integration test in Rails

Table 1: 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 6: 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 2: Test Case Planning Format



Table 3: Test cases for Web Based Payroll System

|  |  |  |  |
| --- | --- | --- | --- |
| Test ID | Description | Expected Results | Actual Results |
| 1 | Admin user entered the username and password. | Admin user should be redirected to the admin home page. | Admin user redirected to the admin home page.  **Test case passed** |
| 2 | Admin user creates a user. | The newly created user should appear in the list. | The newly created user appears in the list.  **Test case passed** |
| 3 | Admin user edits a user. | The user should be successfully updated. | The user successfully updated.  **Test case passed** |
| 4 | Admin user selects user and delete. | The selected users should be successfully deleted. | The selected users successfully deleted.  **Test case passed** |
| 5 | Admin user search user. | The list of users should be displayed based on the search criteria. | The list of users displayed based on the search criteria.  **Test case passed** |
| 6 | Admin user creates a job title. | The newly created job title should appear in the list. | The newly created job title appears in the list.  **Test case passed** |
| 7 | Admin user edits a job title. | The job title should be successfully updated. | The job title successfully updated.  **Test case passed** |
| 8 | Admin user selects job title and delete. | The selected job titles should be successfully deleted. | The selected job titles successfully deleted.  **Test case passed** |
| 9 | Admin user search job title. | The list of job titles should be displayed based on the search criteria. | The list of job titles displayed based on the search criteria.  **Test case passed** |
| 10 | Admin user creates an employment status. | The newly created employment status should appear in the list. | The newly created employment status appears in the list.  **Test case passed** |
| 11 | Admin user edits an employment status. | The employment status should be successfully updated. | The employment status successfully updated.  **Test case passed** |
| 12 | Admin user selects employment status and delete. | The selected employment statuses should be successfully deleted. | The selected employment statuses successfully deleted.  **Test case passed** |
| 13 | Admin user search employment status. | The list of employment statuses should be displayed based on the search criteria. | The list of employment statuses displayed based on the search criteria.  **Test case passed** |
| 14 | Admin user creates a job category. | The newly created job category should appear in the list. | The newly created job category appears in the list.  **Test case passed** |
| 15 | Admin user edits a job category. | The job category should be successfully updated. | The job category successfully updated.  **Test case passed** |
| 16 | Admin user selects job category and delete. | The selected job categories should be successfully deleted. | The selected job categories successfully deleted.  **Test case passed** |
| 17 | Admin user search job category. | The list of job categories should be displayed based on the search criteria. | The list of job categories displayed based on the search criteria.  **Test case passed** |
| 18 | Admin user creates a department. | The newly created department should appear in the list. | The newly created department appears in the list.  **Test case passed** |
| 19 | Admin user edits a department. | The department should be successfully updated. | The department successfully updated.  **Test case passed** |
| 20 | Admin user selects department and delete. | The selected departments should be successfully deleted. | The selected departments successfully deleted.  **Test case passed** |
| 21 | Admin user search department. | The list of departments should be displayed based on the search criteria. | The list of departments displayed based on the search criteria.  **Test case passed** |
| 22 | Admin user creates a employee. | The newly created employee should appear in the list. | The newly created employee appears in the list.  **Test case passed** |
| 23 | Admin user edits an employee. | The employee should be successfully updated. | The employee successfully updated.  **Test case passed** |
| 24 | Admin user selects employee and delete. | The selected employees should be successfully deleted. | The selected employees successfully deleted.  **Test case passed** |
| 25 | Admin user search employee. | The list of employees should be displayed based on the search criteria. | The list of employees displayed based on the search criteria.  **Test case passed** |
| 26 | Admin user search employee’s attendance. | The list of attendance should be displayed based on the search criteria. | The list of attendance displayed based on the search criteria.  **Test case passed** |
| 27 | Admin user creates a pay rate. | The newly created pay rate should appear in the list. | The newly created pay rate appears in the list.  **Test case passed** |
| 28 | Admin user edits a pay rate. | The pay rate should be successfully updated. | The pay rate successfully updated.  **Test case passed** |
| 29 | Admin user selects pay rate and delete. | The selected pay rates should be successfully deleted. | The selected pay rates successfully deleted.  **Test case passed** |
| 30 | Admin user search pay rate. | The list of pay rates should be displayed based on the search criteria. | The list of pay rates displayed based on the search criteria.  **Test case passed** |
| 31 | Admin user creates an overtime pay rate. | The newly created overtime pay rate should appear in the list. | The newly created overtime pay rate appears in the list.  **Test case passed** |
| 32 | Admin user edits an overtime pay rate. | The overtime pay rate should be successfully updated. | The overtime pay rate successfully updated.  **Test case passed** |
| 33 | Admin user selects overtime pay rate and delete. | The selected overtime pay rates should be successfully deleted. | The selected overtime pay rates successfully deleted.  **Test case passed** |
| 34 | Admin user search overtime pay rate. | The list of overtime pay rates should be displayed based on the search criteria. | The list of overtime pay rates displayed based on the search criteria.  **Test case passed** |
| 35 | Admin user creates a salary adjustment. | The newly created salary adjustment should appear in the list. | The newly created salary adjustment appears in the list.  **Test case passed** |
| 36 | Admin user edits a salary adjustment. | The salary adjustment should be successfully updated. | The salary adjustment successfully updated.  **Test case passed** |
| 37 | Admin user selects salary adjustment and delete. | The selected salary adjustments should be successfully deleted. | The selected salary adjustments successfully deleted.  **Test case passed** |
| 38 | Admin user search salary adjustment. | The list of salary adjustments should be displayed based on the search criteria. | The list of salary adjustments displayed based on the search criteria.  **Test case passed** |
| 39 | Admin user view all employees overtime chart. | The chart should be displayed. | The chart is displayed.  **Test case passed** |
| 40 | Admin user generates filtered overtime chart. | The chart should be displayed based on the filtered criteria. | The chart is displayed based on the filtered criteria.  **Test case passed** |
| 41 | Admin user view all hourly paid employees total work hours chart. | The chart should be displayed. | The chart is displayed.  **Test case passed** |
| 42 | Admin user generates filtered total work hours chart. | The chart should be displayed based on the filtered criteria. | The chart is displayed based on the filtered criteria.  **Test case passed** |
| 43 | Admin user view all employees hourly payroll chart. | The chart should be displayed. | The chart is displayed.  **Test case passed** |
| 44 | Admin user generates filtered hourly payroll chart. | The chart should be displayed based on the filtered criteria. | The chart is displayed based on the filtered criteria.  **Test case passed** |
| 45 | Admin user view all employees payslip. | The payslip should be opened in a window. | The payslip is opened in a window.  **Test case passed** |
| 46 | Admin user search employee and view selected employee’s payslip. | The list of employees should be displayed based on the search criteria and the payslip should be opened in a window. | The list of employees displayed based on the search criteria and the payslip is opened in a window.  **Test case passed** |
| 47 | Admin user clicks Logout button. | Admin user should be redirected to the login page. | Admin user redirected to the login page.  **Test case passed** |
| 48 | Hourly paid employee entered the username and password. | Hourly paid employee should be redirected to the hourly paid employee home page. | Hourly paid employee redirected to the hourly paid employee home page.  **Test case passed** |
| 49 | Hourly paid employee edits personal details. | The details should be successfully updated. | The details successfully updated.  **Test case passed** |
| 50 | Hourly paid employee edits contact details. | The details should be successfully updated. | The details successfully updated.  **Test case passed** |
| 51 | Hourly paid employee view job details. | The job details should be displayed. | The job details displayed.  **Test case passed** |
| 52 | Hourly paid employee view salary details. | The salary details should be displayed. | The salary details displayed.  **Test case passed** |
| 53 | Hourly paid employee edits qualification details. | The details should be successfully updated. | The details successfully updated.  **Test case passed** |
| 54 | Hourly paid employee view total work hours chart. | The chart should be displayed. | The chart is displayed.  **Test case passed** |
| 55 | Hourly paid employee generates filtered total work hours chart. | The chart should be displayed based on the filtered criteria. | The chart is displayed based on the filtered criteria.  **Test case passed** |
| 56 | Hourly paid employee view hourly payroll chart. | The chart should be displayed. | The chart is displayed.  **Test case passed** |
| 57 | Hourly paid employee generates filtered hourly payroll chart. | The chart should be displayed based on the filtered criteria. | The chart is displayed based on the filtered criteria.  **Test case passed** |
| 58 | Hourly paid employee view payslip for selected month and year. | The payslip should be opened in a window. | The payslip is opened in a window.  **Test case passed** |
| 59 | Hourly paid employee clicks Logout button. | Hourly paid employee should be redirected to the login page. | Hourly paid employee redirected to the login page.  **Test case passed** |
| 60 | Monthly paid employee entered the username and password. | Monthly paid employee should be redirected to the monthly paid employee home page. | Monthly paid employee redirected to the monthly paid employee home page.  **Test case passed** |
| 61 | Monthly paid employee edits personal details. | The details should be successfully updated. | The details successfully updated.  **Test case passed** |
| 62 | Monthly paid employee edits contact details. | The details should be successfully updated. | The details successfully updated.  **Test case passed** |
| 63 | Monthly paid employee view job details. | The job details should be displayed. | The job details displayed.  **Test case passed** |
| 64 | Monthly paid employee view salary details. | The salary details should be displayed. | The salary details displayed.  **Test case passed** |
| 65 | Monthly paid employee edits qualification details. | The details should be successfully updated. | The details successfully updated.  **Test case passed** |
| 66 | Monthly paid employee view total work hours chart. | The chart should be displayed. | The chart is displayed.  **Test case passed** |
| 67 | Monthly paid employee generates filtered total work hours chart. | The chart should be displayed based on the filtered criteria. | The chart is displayed based on the filtered criteria.  **Test case passed** |
| 68 | Monthly paid employee view overtime chart. | The chart should be displayed. | The chart is displayed.  **Test case passed** |
| 69 | Monthly paid employee generates filtered overtime chart. | The chart should be displayed based on the filtered criteria. | The chart is displayed based on the filtered criteria.  **Test case passed** |
| 70 | Monthly paid employee view payslip for selected month and year. | The payslip should be opened in a window. | The payslip is opened in a window.  **Test case passed** |
| 71 | Monthly paid employee clicks Logout button. | Monthly paid employee should be redirected to the login page. | Monthly paid employee redirected to the login page.  **Test case passed** |

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

# References

|  |  |
| --- | --- |
| [1] | angad, "Testing In Software Engineering," 2 November 2012. [Online]. Available: http://techforum4u.com/content.php/417-Testing-In-Software-Engineering. [Accessed March 2013]. |
| [2] | A. Bertolino, "Software testing research and practice," 2003. |