Photon
Phresco

Phresco Framework

Newest version of the Social Media Framework

Version 1.2.1
August 2012

This document applies to Phresco Framework v 1.2.1 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

TABLE OF CONTENTS

LIST OF FIGURES

1 About This Guide

The purpose of this guide is to assist developers, testers, release engineers, managers and others who aim to use Phresco Framework user interface for maintaining and controlling the critical phases in the entire life cycle of a project.

1.1 Document Conventions

Table 1: Conventions

Convention	Description
Blue	Identifies elements on a screen
Brown	Identifies tabs on a screen
	Note
Green	Phresco UI Navigation (Breadcrumbs)
Italic	Code structure
"Yellow"	Important
*	This is mandatory symbol

2 Introduction

Phresco, a product of Photon InfoTech Pvt Ltd, is designed to work seamlessly with the powerful and more popular technologies; Phresco assists users in creating projects that boast persistent uniformity in quality across platforms throughout the lifecycle of a project. Aiming to be the go-to tool for next generation multichannel development, Phresco allows the user to develop projects simultaneously with faster cycles across multiple channels. Phresco is endowed with latest tools and capabilities along with expanding libraries of pre-built templates, standardized codes and manifold archetypes which aid the development team to increase their capability by limiting the occurrences of errors and reducing the development time. Projects created using Phresco Framework adheres to the best practices in the industry making them resourceful, effective and reusable.

2.1 What Is Phresco Framework?

Phresco is a next-generation development framework of frameworks. It is a platform for creating next generation web, mobile and Multichannel presences leveraging existing investments combined with accepted industry best practices. Phresco publishes new artifacts (components) in the centralized maven repository or Customer specific repository under appropriate technologies. Once published in the centralized repository, subscribed users can view and deploy those artifacts in their projects.

Phresco encapsulates the best practices of a project structure, re-usable components, standards, documentation, quality assurance and release process. This ensures the quality of a project deliverables, which in turn will increase the productivity of a project. Though Phresco guarantees projects adhering to best practices, it permits every organization to follow their respective quality measures and standards.

3 Phresco – Powered With Facets

3.1 Lifecycle Management

Maintaining end to end lifecycle of a project has never been easier.

From providing a stepwise method in creating a template project to incorporate tools for code validation, build and deployment and continuous improvement, Phresco makes sure that every aspect of a project lifecycle is taken care of. The integration of Phresco with multiple open source projects allows the user community to reap the full benefits of the Framework.

3.2 Pre - Built Architecture

Phresco provides a pre-built architecture model with certified template architectures like Multichannel widget, SharePoint web parts, NodeJS Service gateway, etc.; it contains standard build structures with supported versions.

3.3 Quality Assurance

Phresco has a series of powerful automated testing instruments designed to analyze and test the work at the code level through various QA schemes.

The testing methods deployed in Phresco include:

- Unit Testing
- Functional Testing
- Performance Testing
- Load Testing

By using static code analysis tools to pick up and compare the metrics of different technologies with the source codes, Phresco makes sure that your project is of quality standards. Phresco Archetypes, created by following the best practices in the respective domains, help the developers in creating model projects by providing basic templates for any desired technology.

3.4 Web Development

In Phresco, web development supports for ASP.NET, SharePoint, PHP (raw), PHP (Drupal), and WordPress, HTML5 YUI Mobile Widget, HTML5 JQuery Mobile Widget, Html5 Multichannel YUI Widget, HTML5 Multichannel JQuery Widget and Java standalone. It also supports web applications using Responsive web design and Java / NodeJS based web services. (Stacks)

3.5 Mobile/Web Application Development

Mobile/ web application can be developed using HTML5 YUI Mobile iPhone Native, iPhone Hybrid, Android Native, Android Hybrid and HTML5 jQuery widgets as Responsive web design. Windows 8 support for mobile application is the new feature introduced in phresco. Applications can be created, build can be generated and it can be deployed using phresco.

3.5.1 iPhone Features

- **IOS Platforms**

  While testing IOS application, Phresco framework provides an option of selecting the list of the devices and its versions. Earlier, the lists of devices were hardcoded in the user interface but in the latest version, the system finds out the installed versions of the devices and shortlists only the versions installed.

3.5.2 Android Features

- **Proguard Enablement**

  The source code can be decompiled and viewed from the installed application. To keep it in encrypted format, Proguard enablement option is used which shuffles the classes of java and gives id to each class which is not related to the classes.
  That is class will be shuffled and will be renamed as a, b, c, d which will not be in understandable format.

- **Application Signing**

  The Android system requires that all installed applications be digitally signed with a certificate whose private key is held by the application's developer. In order to make application available in Android store (Market),

the application has to be certified. Phresco allows user to select certificate through the Phresco UI while build the app for final release.

3.6 Standalone Application Development

Standalone application does not require any software other than the operating system to run. This support is provided in the latest version of Phresco, where java standalone applications can be created.

3.7 Multichannel Presence

The Burgeoning user base of Internet has forced organizations to provide customers a seamless environment for buying or utilizing their services in a manner that the channels complement each other.

Having this in mind, Phresco has enabled Widgetized Responsive web design to be accessed across various channels.

3.8 Responsive Web Design

When creating a web page, it's been a common tactic to serve up two different sets of pages one for desktop browsers, another for mobile devices. With such a wide variety of mobile devices, screen sizes, and hardware features, designing a separate version of a site for each quickly becomes impractical.

Phresco deploys the Responsive Web Design (RWD) concept that intelligently adjusts the layout and features of a website based on how it's being viewed. Phresco with the aid of RWD helps developers in exposing the myriad functionalities developed using Phresco across multiple channels i.e., web, mobile, tablet and kiosk with minimum resizing, panning and scrolling efforts.

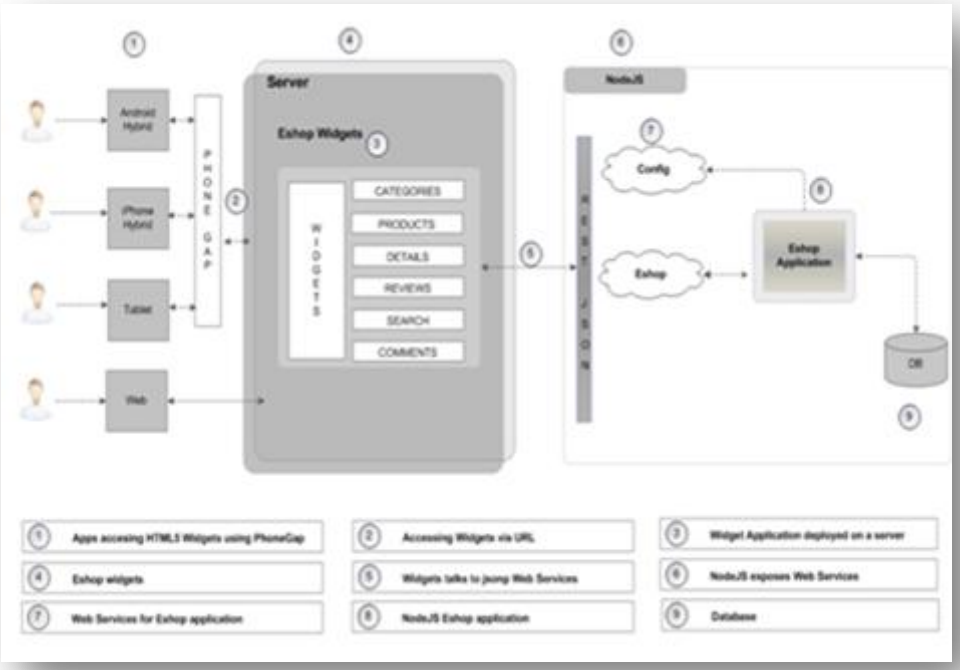


Figure 3-1: Multichannel Architecture in Phresco

3.9 Reuse and Continuous Improvement

For any organization, promoting reusability of the components is important. The components (artifacts) such as libraries, features, Web parts, archetypes, pilot projects and other artifacts can be reused in numerous projects with minimal changes.

3.10 Continuous Integration

One of the cherished aspects of Phresco is the Continuous Integration feature made available as part of the Framework which generates the build automatically deploys it and tests the build in periodic manner. With this feature, it is trouble-free to keep track of the latest build updates and make sure that the builds adapt seamlessly to the existing project which is otherwise a tedious process. Added advantage in the Phresco is multiple job creation. Multiple jobs can be created to a single project and it can be linked with each other to maintain the order of the automation.

4 Getting Started With Phresco Framework

Once JAVA_HOME path is set and jdk 1.6.0_18 version (or above versions which is yet to be validated) is installed, it's time to extract, install and start creating projects using Phresco.

4.1 Supported Platform

- Windows: xp, 2003, 2008, windows 7 (32 and 64 bits are supported)
- Mac: Mountain lion, Snow Leopard
- Linux: RedHat, SUSE Linux, Ubuntu

4.2 Supported Browsers

- Firefox: Version 6 and above
- Google Chrome: Version 11 and above(except for PHP , Drupal web driver migration)
- Internet Explorer: Version 9 and above
- Opera: Version 10 and above
- Safari: Version 5 and above (as of now Selenium RC is only supported for functional tests)

4.3 Steps To Extract and Execute Phresco Framework

- Download Phresco Framework-<version>.zip.
- Run Start framework server.bat for windows/ sh Start framework server.sh for Mac and Linux.
- This will automatically open the Phresco login page in the browser. (Default Browser)

4.4 Logging On To Phresco Framework

- In the Web Browser, enter the URL where Phresco Framework is hosted.
- At the log in screen, enter valid Insight (Photon) credentials.
- Now click Login.

4.5 Using the Navigation Controls in User Interface

Phresco's interactive navigation controls aid users in moving seamlessly through the Framework guiding them in achieving the optimum result.

After logging in, users will find a simple and easy way to navigate to the Home, Applications, and Settings or Help menus.

- Click on Go to Phresco Home in order to reach Phresco HOME page.
- Click on Applications, Settings or Help in order to reach the respective pages.

Figure 4-1: Welcome page

4.6 Phresco Videos

The videos provide a gist of what Phresco is all about and also gives a step by step audio-visual representation of how to browse through Phresco. It helps the user with a clear picture as how to work on Phresco framework. For Example: Creating an application, configurations of applications, testing a project etc.

Phresco video page is readily obtainable when Go to Phresco home is clicked. The playlists mostly has the complex working experience of any project.

Figure 4-2: Video library in home page

4.7 Logging Out Of Phresco Framework

- To log out of Phresco Framework, click Sign Out (located in the drop down list on the right top corner of the page).
- If Phresco is left inactive for 1 day, the session will automatically expire.

Figure 4-3: Change skin, Help, About Phresco and Sign Out

4.8 To Change the Skin Color of Your Account

There are two different skins (Red and Blue) available in Phresco Framework. In Phresco User Interface, click skins (this link resides in the right side at the very top of the page).

4.9 To Update the Available Version

Phresco releases periodic updates for the Framework and these new versions enables the Update Available button in the About Phresco link. Users can update to latest available releases by clicking the Update Available button.

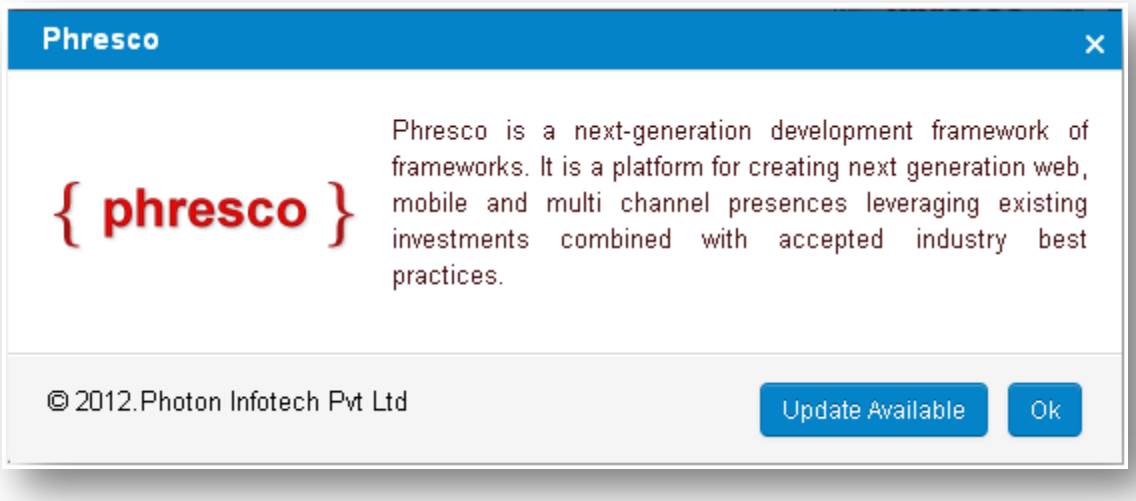


Figure 4-4: About Phresco

5 Phresco Project Life Cycle

Phresco is used to perform the following tasks effectively.

5.1 Project Creation

Phresco primarily helps the developers, testers, release engineers, managers and others to create projects with flexibility and affordability. Projects that are created using Phresco, invariably have high quality standards and best practiced structure. Once a project is created, Phresco also manages the entire lifecycle of a project and replenish the best productivity.

Creating an Application

Projects can be created with any type of technology and features (create hyperlink). Features and technologies are provided out of the box and can be used for creating a project. Phresco is user friendly and helps the software engineers to create projects with more ease. The applications can be created from the Applications -> Add Application. Applications tab has the App info and features from which the libraries can be selected for project creation in the wizard.

App Info

This encompasses the information about an application of a project. Also this holds the information about the Server, Database, Web Service and Email components which a project consumes.

Fields present in the App info are as follows

Name

It denotes the Name in which a project is created. Maximum text input of the field is restricted to 30 alphanumeric characters and is a mandatory field.

Code

It denotes the code in which a project is created. Maximum text input of the field is restricted to 12 alphanumeric characters and is not a mandatory field.

Description

It denotes the description of a project created. Upto 150 characters of the field can be entered and is not a mandatory field.

Version

It denotes the Version of a project. 20 alphanumeric characters with a special character "." Can be entered in this field.

Applications

Application type can be chosen by selecting the radio button against the desired application type. There are three types of applications administered by Phresco
They are

- Web Application
- Mobile Application
- Web Services

Technology (Archetype)

Type of the technology should be selected in this field and it is a mandatory field. Mentioned below are the dependent technologies for each application type.

Table 2: The technologies under web application

Technologies	Version
PHP	**5.0.x,5.1.x,5.2.x,5.3.x, 5.4.x**
Drupal6	**6.3, 6.25, 6.19, 6.14**
Drupal7	**7.8, 7.12, 7.14**
SharePoint	**3.5, 3.0, 2.0**
HTML5 Multichannel YUI Widget	**1.6, 1.5**
HTML5 Multichannel jQuery Widget	**1.6, 1.5**
HTML5 jQuery Mobile Widget	**1.6, 1.5**
HTML5 YUI Mobile Widget	**1.6, 1.5**
ASP.NET	**3.5, 3.0, 2.0**

WordPress	**3.3.1**
Java stand alone	**1.6, 1.5**

Table 3: The technologies under web services

Web Services	Version
NodeJS Web Service	**0.6x, 0.7x, 0.8x**
Java Web Service	**1.6, 1.5**

The technologies under mobile application are

- iPhone Native
- iPhone Hybrid
- Android Native
- Android Hybrid

Pilot Project

Phresco has included Pilot projects for most of the technology it support. Pilot project is an actual project built with the best practices of project development, libraries, components and validated code structures. Any project can be made a pilot project provided it addresses the important criteria- being developer specific and tester specific.

The pilot project selected from the drop down list incorporates some libraries, components and validated code structures to the newly created project and henceforth can be modified based on the requirement. If none project is selected from the drop down list, all the components and libraries should be configured.

Phresco by publishing pilot projects in the repository server,

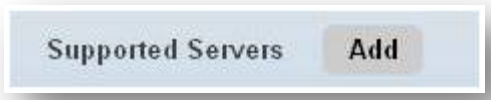
- Allows components/libraries deployed to be re-used in multiple other projects.
- Authorizes re-branding of the pilot projects.

Advantages

- Less effort needed in creating new projects.
- Ease in adopting the validated code structures and verified test cases ensure quality and saves time.
- Perks up the application performance.

Supported Servers

Every organization has an affinity for adopting a projects created to their in-house servers, saving time and allaying the adaptability concerns. Understanding this need, Phresco has been configured in such a way that a project developed or adopted run on a wide range of servers available in the market.



The desired server can be selected using the "Add" option against Supported Servers in the "**App info**" page

For the same project, Phresco allows more than one server to be shortlisted. This has been implemented to answer the adaptability concerns that afflict projects. In the App info page, the developers can choose their preferred server from the list.

Table 4: Supported servers

Supported Servers	Versions
Apache Tomcat	7.0.x, 6.0.x, 5.5.x
JBoss	7.1.x, 7.0.x, 6.1.x, 6.0.x, 5.1.x, 5.0.x, 4.2.x, 4.0.x
IIS	7.5, 7.0, 6.0, 5.1, 5.0
Web Logic	12c(12.1.1), 11gR(10.3.6), 11g(10.3.1), 10.3, 10.0, 9.2, 9.1
Apache	2.3, 2.2, 2.0, 1.3
NodeJS	0.6.x, 0.7.x, 0.8.x
Jetty	8.x, 7.x, 6.x, 5.x, 4.x

Supported Databases

As with the servers, Phresco incorporates popular databases that are widely preferred by organizations.

The desired database can be selected using the "Add" option against Supported Databases in the "**App info**" page
For the same project, Phresco allows more than one database to be shortlisted. In the App info page, the developers can choose their preferred database from the list.

Table 5: Supported databases

Supported Databases	Versions
MySQL	5.5.1, 5.5, 5.1, 5.0, 4.1, 4.0
Oracle	11gR2, 11gR1, 10gR1, 10gR2 9iR2, 9iR1, 8iR3, 8iR2, 8iR1, 8i
MongoDB	2.0.4, 1.8.5
DB2	10, 9.7, 9.5, 9
MSSQL	2012, 2008R2, 2008, 2005

Note

Oracle database requires a dependency jar file. This jar should be used with a proper license and hence the users should manually add the dependency tag in the pom file which points the file path present in the local machine. Below is the example code snippet

```
<dependency>
   <groupId>com.oracle</groupId>
   <artifactId>ojdbc14</artifactId>
   <version>10.2.0.</version>
    <scope>system</scope>
    <systemPath>DRIVER NAME:/oracle/webdriver/ojdbc14--10.2.0.jar</systemPath>
</dependency>
```

Consumes

Some of the established Web Services are supported in Phresco Framework. As like Servers and the databases, Phresco allows more than one Web Service to be selected from the "App Info" page. From the shortlisted Web Services displayed in the App Info page, the desired option can be chosen.

The supported Web Services in Phresco Framework includes,

- REST/JSON 1.0
- REST/XML 1.0
- SOAP 1.1
- SOAP 1.2

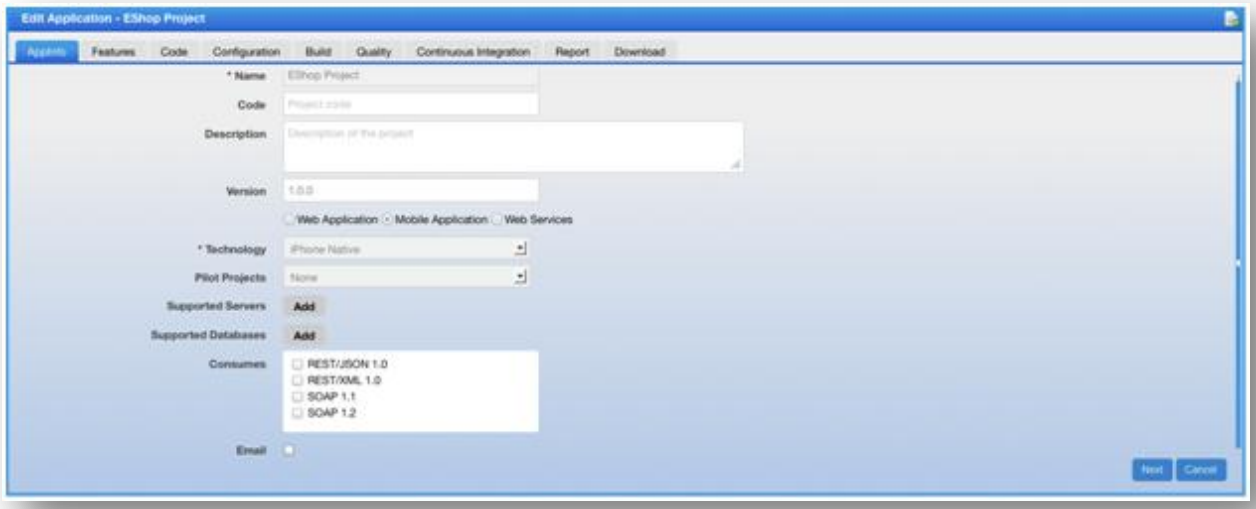
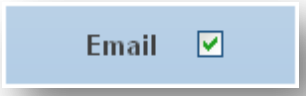


Figure 5-1: App info page

Email

Email checkbox can be selected if a project requires email configuration. There are some projects which uses the option of sending email notification to the mail id furnished by the user.

Only if the email checkbox is selected during the creation of project, email type will be displayed in the drop down shown in the configuration page.



When the App info and the features are selected a project creation is completed.

Features

Phresco also renders exiguous features to enhance a project. This includes External features, JS libraries and custom features.

External features:

External features are those features provided out of the box by Phresco for the users to enhance their project. These features can be chosen in multiple based on the requirement of a project.

JS Libraries:

JS Libraries are the features provided out of the box by Phresco for the users in implementing the code. These libraries can also be chosen in multiple based on a project's requirement.

Custom Features:

These features are the customer specific features which can be incorporated in Phresco on a request by the user based on a project's requirement.

Features considered in Phresco are

- Libraries: jar, dll, etc
- Modules: Drupal module, NodeJS module, etc
- Features: Web parts

Features are the libraries of a project and the developers make use of these features to have superlative creation of project. For example Facebook application is a feature which can be added to a project. This allows the user to access the Facebook application by contacting the web browser.

Latest version of phresco contains the Features page which is now enabled with a feature icon and feature specific description. On clicking the feature a Description pop up appears.

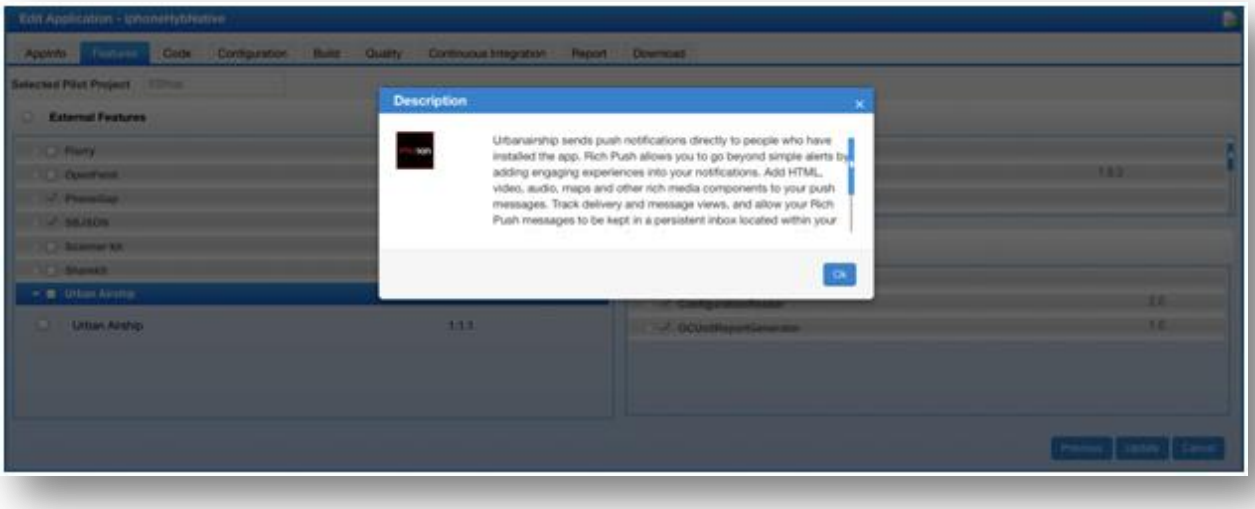


Figure 5-2: Example of the feature icon

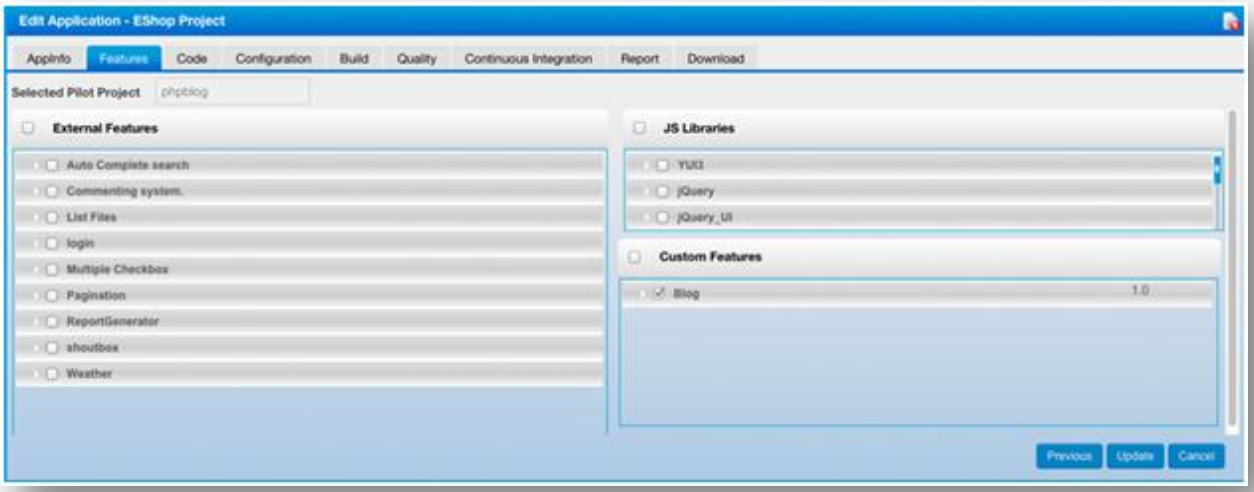


Figure 5-3: Feature selection page

5.2 Code Validation

Code validation is a method in which the source code of a project is tested in compliance with the standards. Phresco renders code validation by manipulating sonar tool which picks up standards for contra distinct technologies and examines in contrast with codes of a project. When Phresco starts, the code validation tool also starts automatically. Code validation process can be triggered by clicking on Validate button only when the sonar tool is in active status.

Code Validation is the tool which aggregates the standards to verify 100% validation of the code and code coverage for all the technologies. The errors are extricated depending on how the errors affect a project. Result is furnished in the form of graph.

New version of Phresco supports code validation of test cases. Code validation for test cases can be done against source code and also functional test cases.

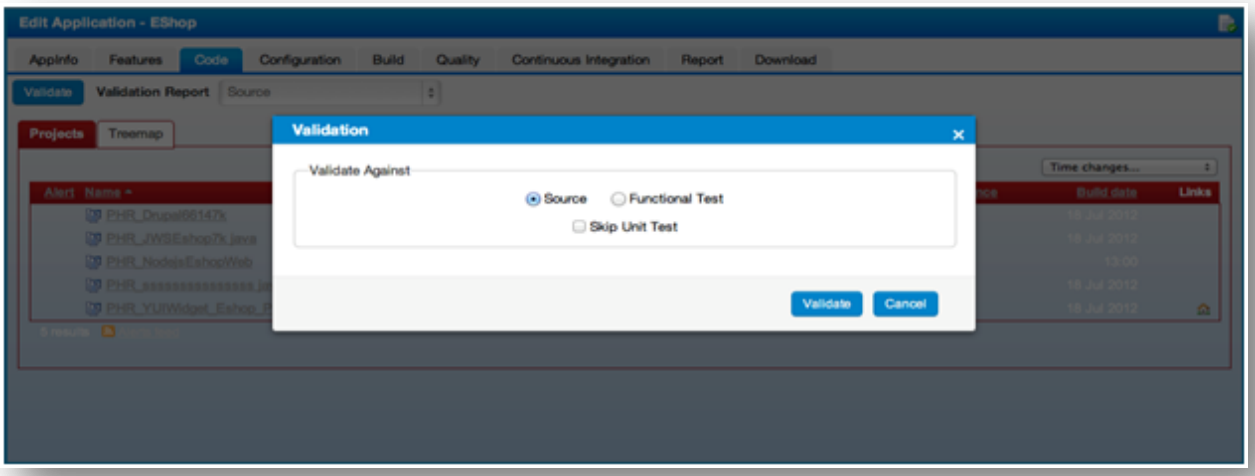


Figure 5-4: Code Validation Pop Up

Skip Unit test

When the check box is enabled, unit tests will be skipped and code validation runs against source or functional test when the radio button is clicked.

Violations and Rules Compliance

- Blocker
- Critical
- Major
- Minor
- info

Prerequisites for code validation

A prior installation of PEAR software is essential to run code validation for PHP, Drupal and WordPress projects.

- PHP Path should be set in environment variable
- Maven path should be set before the installation of Pear
- Pear has to be installed

To install PEAR in Phresco,

1. Set pear path- {pear installation path}/<APACHE>/bin/php in the environment variables for users
2. Set PHP path- {PHP installation path}/<APACHE>/bin/php/php 5.3.5 in the environment variables for system.
3. To set Maven path in the environment:

- Open a command prompt
- Navigate to Phresco Framework-> bin
- Run the env.bat (windows)/env.sh (Mac and LINUX) for this will set Maven temporarily in the command prompt

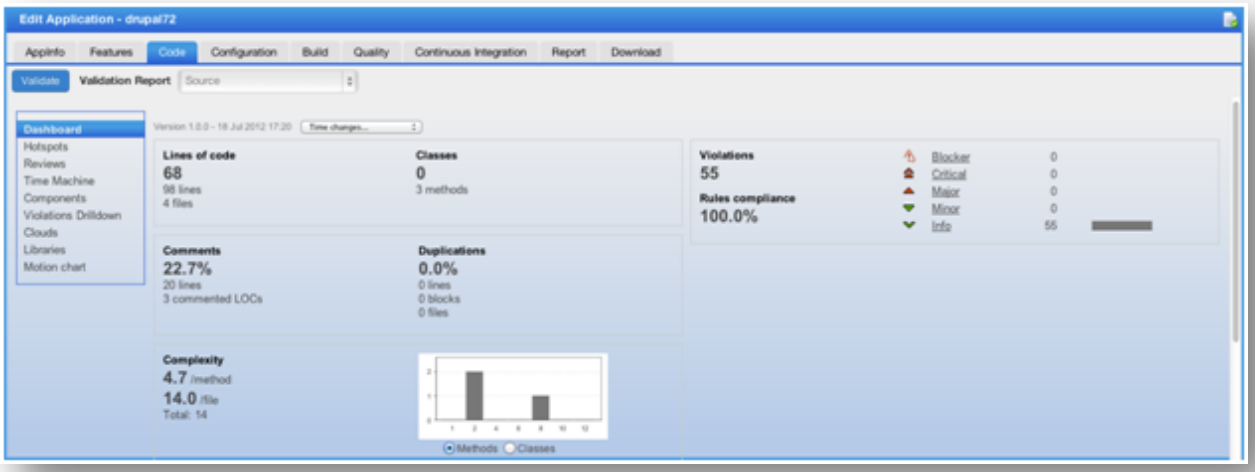


Figure 5-5: Code validation report

- Following this the steps for Pear installation should be carried over in the same command prompt

---

Note:

- This is carried over only when maven is not available.
- Setup for wamp and xamp is available in the downloads tab of Phresco.

---

4. Download <Php_Drupal_code_validation_setup> from Phresco
5. Extract the zip file.
6. Edit Php_Drupal_code_validation_setup-> PearInstall-> setup.bat.
7. Configure
8. Php path should be set in setup.bat by opening it with edit tool. Below mentioned is the syntax to set the php path.
9. Call pear.bat <php path>
10. E.g.: call pear.bat <driver_name>:\<APACHE>\bin\php\php5.3.5
11. Execute setup.bat in Command Prompt / Terminal
12. On the execution of the command there are few steps which require User's input.

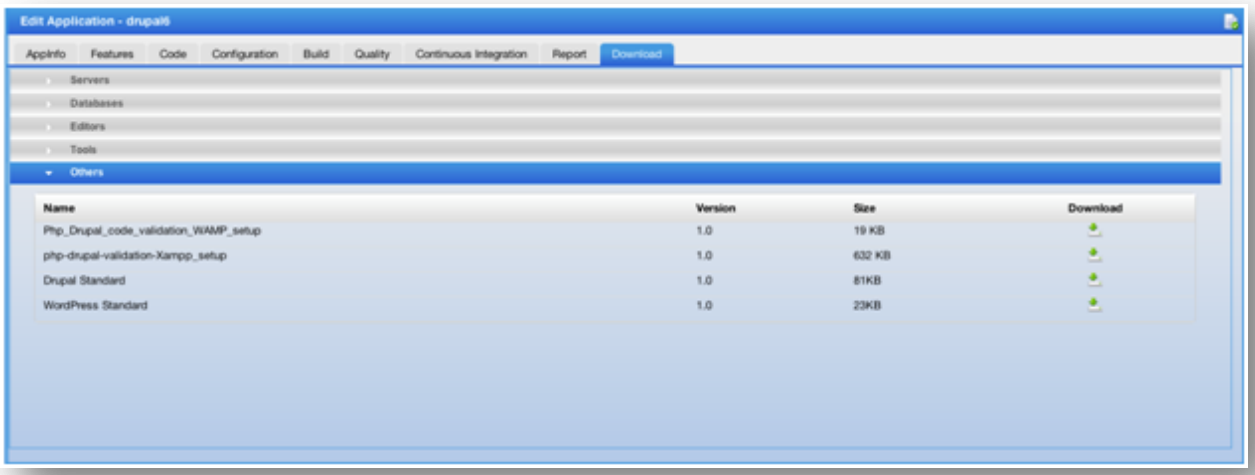


Figure 5-6: ***Downloading Php_Drupal_code_validation_setup for pear installation***

Steps which requires user input

a. *Are you installing a system-wide PEAR or local copy?*
Select an option or if it takes time to respond, a default value will be chosen by the system itself.
The default value would be *<system:local> [system] :*
Press Enter.

b. *1-12, 'all' or Enter to continue: _*
Press enter.

c. *Would you like to alter php.ini <driver_name>:\<APACHE>\bin\php\php5.3.5\php.ini>? [Y/n] :*
Type y and press enter.

d. *Press Enter to continue:*
Press enter.

e. *Press any key to continue*
Press enter.

f. *<driver_name>:\<APACHE>\bin\php\php 5.3.5>call pear upgrade pear*
Wait till the system downloads the file.

g. After downloading system pops up with a message box.
Are you sure you want to add the information in <driver_name>:\<APACHE>\bin\php\php5.3.5\PEAR_ENV.reg to the registry?

Click yes.

h. Click ok in the message box stating "Information in <driver_name>:\<APACHE>\bin\php\php5.3.5\PEAR_ENV.reg has been successfully entered into the registry"

i. After the installation of PEAR and when the build is success
Press any key to continue
Press any key.

Note

- When Pear is installed for the first time, the standards need not be downloaded.
- When Pear is already installed in the machine, Drupal standards should be downloaded for Drupal projects from download link and WordPress standards should be downloaded for WordPress projects (refer fig: 5.6)
- The standards should be pasted in the path,

For Windows,

"<DRIVER_NAME>\Apache \bin\php\php5.3.5\PEAR\PHP\CodeSniffer\Standards"

For Mac

"/usr/local/pear/share/pear/PHP/CodeSniffer/Standards"

5.3 Environment Configuration

Environment Configuration is the powerful feature of Phresco for managing multiple environments in a project, where a single build can run on multiple environments without any configuration changes in build.

Environment set up in Phresco user interface is made very simple and the developers can create any number of environments to deploy a single build.

The environment configurations with user credentials and other significant data are stored in Phresco-env-config.xml. Hence Phresco encrypts this xml file for the technologies like PHP, Drupal and WordPress to hold intact data.

Note

- For the successful installation of Drupal the following changes has to be enabled.
- Extension of php_mcrypt in php.ini should be added with php_mcrypt.dll.

Steps for creating an environment

Step1: Multiple environments can be created from the Edit Application page by clicking the Application created->configuration-> environments.

Step2: When the add button is clicked after entering the fields, an environment is created.

Name

It denotes the Name in which a project is created. Maximum text input of the field is restricted to 30 alphanumeric characters and is a mandatory field.

Description

It denotes the Description in which a project is created. This field supports 150 alphanumeric, special characters and is not mandatory.

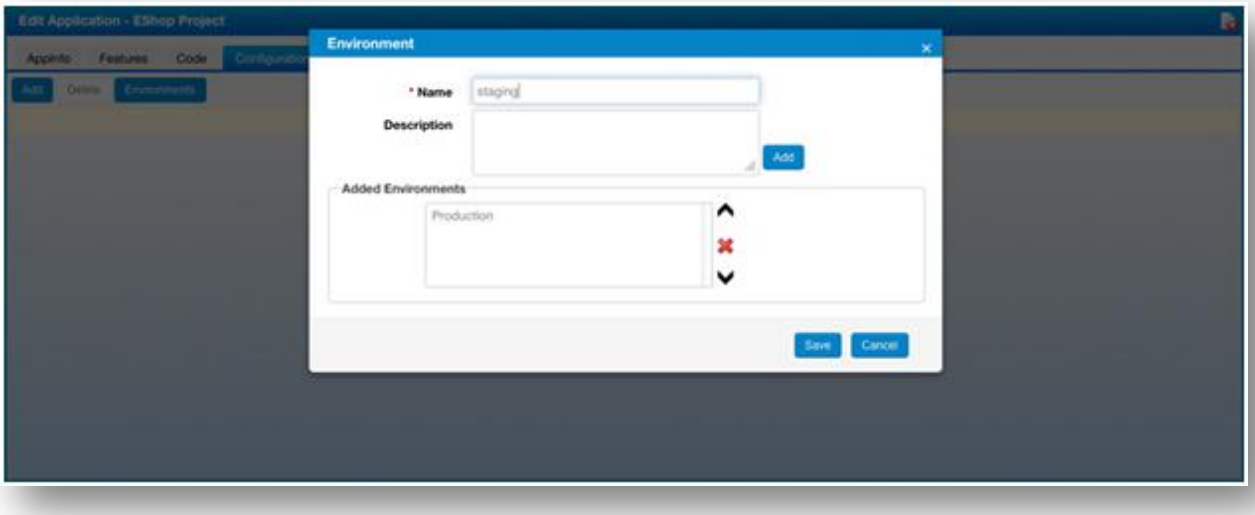


Figure 5-7: ***Environment creation***

Added environment

The added environment lists the environments created. Ordering of environment can be done by using the up and down arrow icons. Deletion of the environment can be done by using the delete icon.

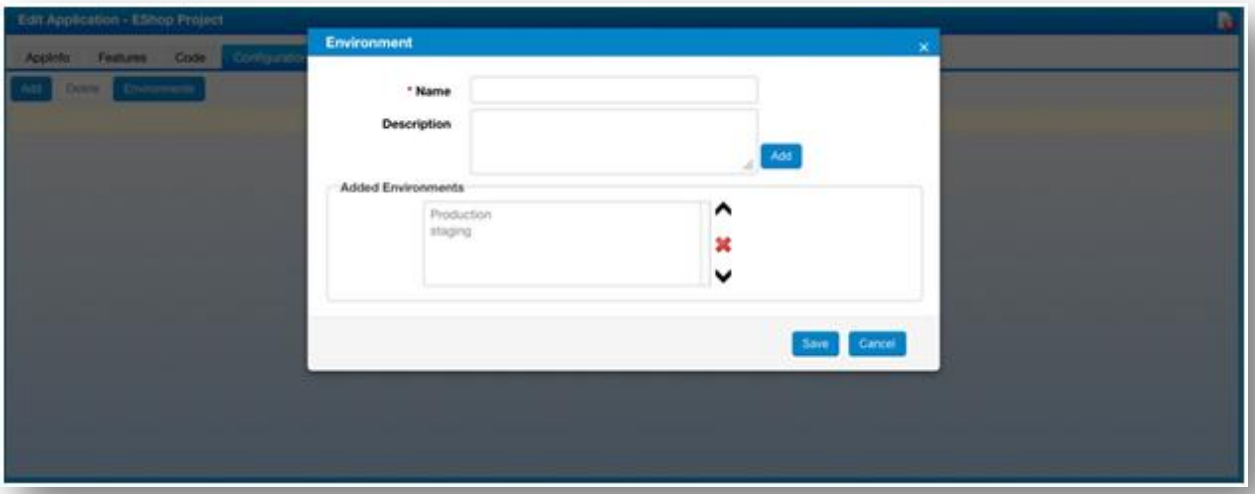


Figure 5-8: Environment creation and ordering

Step3: Server, Database, Web Service and Email can be configured in Application created->Configuration->Add. The created environment reflects in the environment

field and the configurations can be associated under an environment by selecting the environment name from the drop down box.

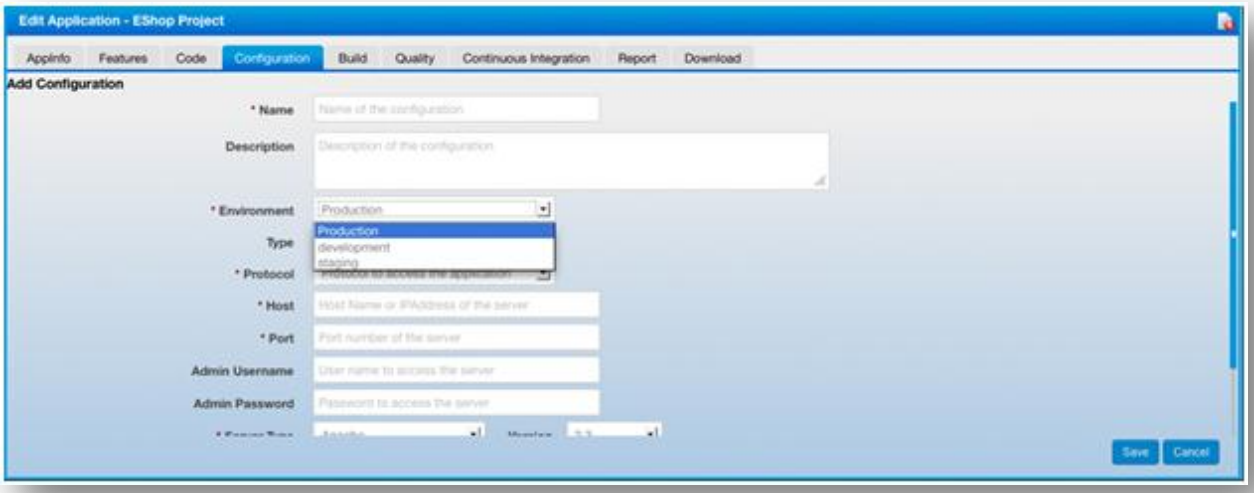


Figure 5-9: Environment selection for a configuration

Note

- Multiple environments can be associated when building a project.
- Projects can be deployed only in one environment at a time.
- In web application only one environment can be selected at a time for functional, performance and load testing whereas for mobile application environments cannot be selected.

Advantages

Apart from these configurations, Phresco also allows user defined configuration template. New configuration templates can be published in Phresco server which would be immediately seen in the drop down list of configuration page. For example log4j template can be published in Phresco server which can be used across a project.

When the environment is created in the global settings, it can be used globally across projects using the show settings option.

Five types of configurations are as follows:

5.3.1 Server Configuration

The server configuration can be added from Application created->Configuration-> Add. The configuration type varies based on the configurations selected in the App Info page.

Mentioned below are the details to be filled for server configuration:

Name

It denotes the Name in which a project is created. Maximum text input of the field is restricted to 30 alphanumeric characters and is a mandatory field.

Description

It denotes the Description of a project which is created. This field supports 150 alphanumeric, special characters and is not mandatory.

Environment

Created environments can be selected from the drop down box. If the environment is not created Production environment is selected as default.

Type

Configuration type can be selected from the drop down box. Server options should be selected for configuration.

Protocol

Protocol should be selected to access the application from dropdown box. http and https are the two types of protocols available. This field is mandatory.

Host

It denotes the Host in which a project is created and is mandatory.

Port

Port number of the server and it must be between 1 and 65535. This field is mandatory and only numeric characters are supported.
Port number should be selected such that the port does not run any other application which will not be known by the configuration.

Admin Username

It denotes the Admin Username of the server in which a project is created. This field supports alphanumeric, special characters and is not mandatory.

Admin Password

It denotes the Admin Password of the server in which a project is created. This field supports alphanumeric, special characters and is not mandatory.

Server Type

This contains the list of servers that was already short listed in the app info page. Only one server can be selected from the list.

Remote Deployment

The checkbox can be enabled to deploy a project in remote machine.

Deploy Directory

Deploy directory of the server on the instance should be filled in this field. It is not a mandatory field and supports alphanumeric and special characters.

Root Context

Root context of the server can be specified here. It does not allow special characters. This is a mandatory field.

Additional Context Path

A project's additional context path can be specified here which can also include special characters.

For example http//localhost:2468/Phresco/login#. Here Phresco is the root context and login# is the additional context path.

Note

- By providing a port number for the server configuration and clicking on RunAgainstSource button for java projects, the inbuilt tomcat will reserve the next immediate port number for tomcat shutdown service.
- For example: If "7070" is provided as port number in server configuration and RunAgainstSource is clicked, "7071" port will be reserved. So configuring the port number "7071" in any other server configuration in a different project and clicking on RunAgainstSource will result in bind exception.

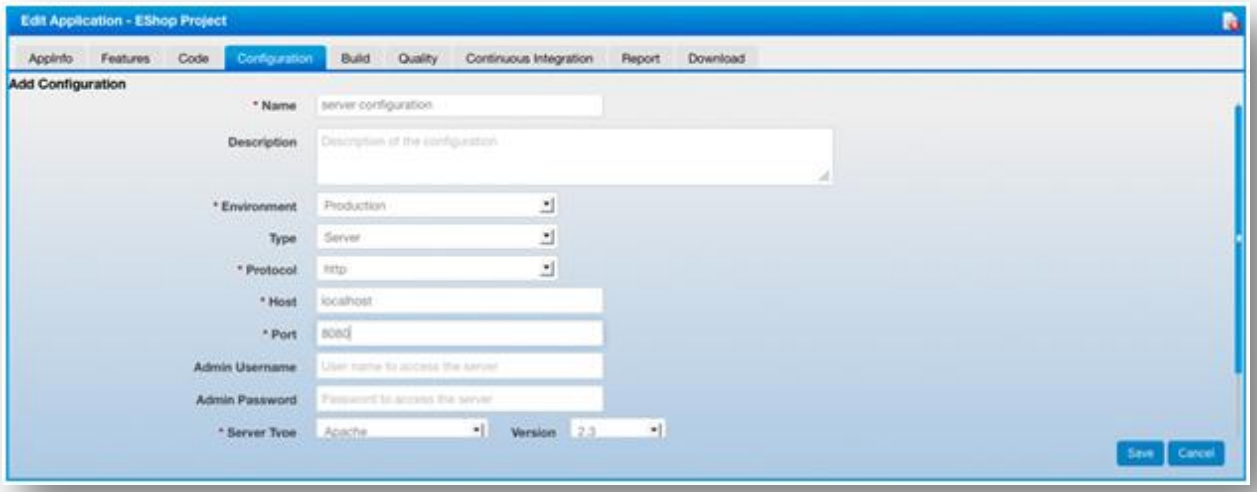


Figure 5-10: ***Server configuration for an environment***

5.3.2 Database Configuration

Name

It denotes the Name in which a project is created. Maximum text input of the field is restricted to 30 alphanumeric characters and is a mandatory field.

Description

It denotes the Description in which a project is created. This field supports 150 alphanumeric, special characters and is not mandatory.

Environment

Environment created can be selected from the drop down box. If the environment is not created, production environment is selected as default and this field is mandatory.

Type

Configuration type can be selected from the drop down box. Database option should be selected for database configuration.

Host

It denotes the Host in which a project is created and is mandatory.

Port

It denotes port number of the database. Port number must be between 1 and 65535. This field is mandatory and only numeric characters are supported. Port number should be selected such that the port does not run any other application which will not be known by the configuration.

Username & Password

It denotes Username and Password to access the database. Username is “root”.

DB Name

Database name should be entered and this field supports alphanumeric and special characters.

DB Type

This contains the list of database that is already short listed in the app info page.

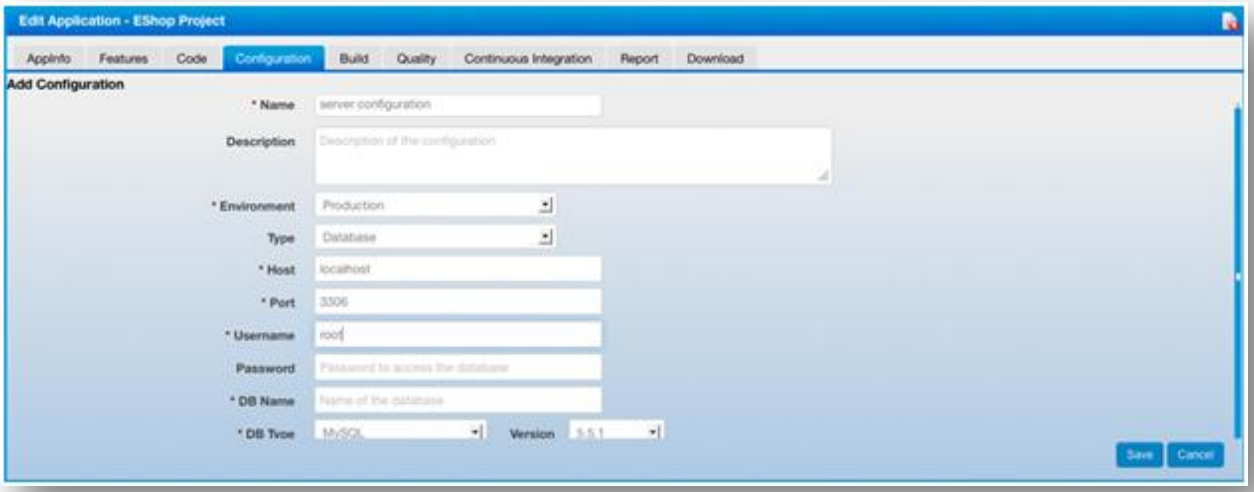


Figure 5-11: ***Database configuration for an environment***

5.3.3 Web Service Configuration

Name

It denotes the Name in which a project is created. Maximum text input of the field is restricted to 30 alphanumeric characters and is a mandatory field.

Description

It denotes the Description in which a project is created. This field supports 150 alphanumeric, special characters and is not mandatory.

Environment

Environment created can be selected from the drop down box. If the environment is not created Production environment is selected as default and this field is mandatory.

Type

Configuration type can be selected from the drop down box. Web Service option should be selected for Web Service configuration.

Host

It denotes the Host name of the web service in which a project is created and is mandatory.

Port

This field is the port number of Web Service. Port number must be between 1 and 65535. This field is mandatory and only numeric characters are supported. Port number should be selected such that the port does not run any other application which will not be known by the configuration.

Context

Context of the web service can be entered in this field and is mandatory.

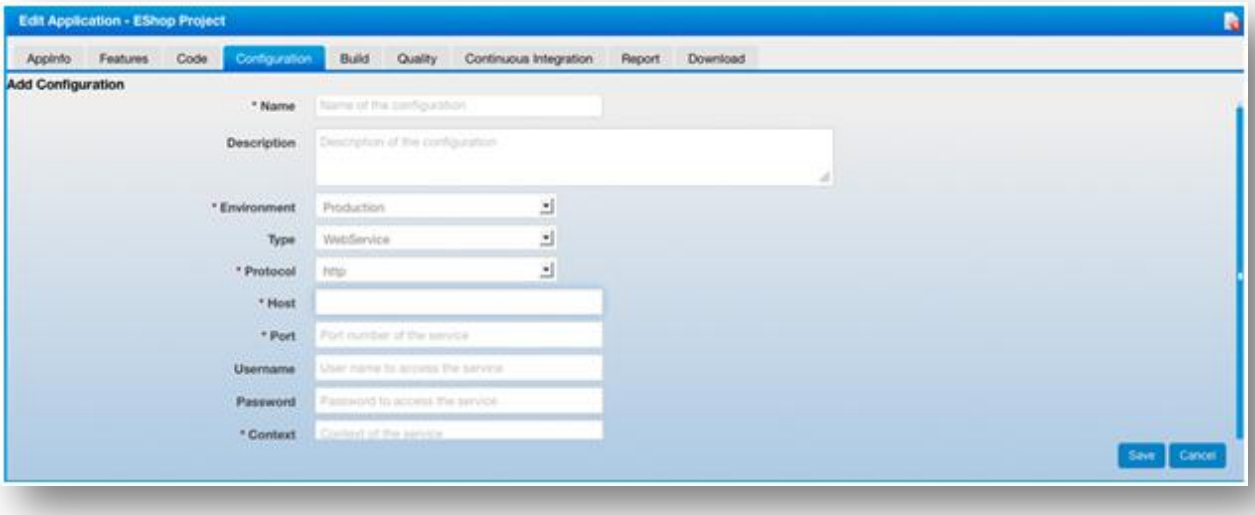


Figure 5-12: ***Web Service configuration for an environment***

5.3.4 Email Configuration

Name

It denotes the Name in which a project is created. Maximum text input of the field is restricted to 30 alphanumeric characters and is a mandatory field.

Description

It denotes the Description in which a project is created. This field supports 150 alphanumeric, special characters and is not mandatory.

Environment

Environment created can be selected from the drop down box. If the environment is not created Production environment is selected as default and this field is mandatory.

Type

Configuration type can be selected from the drop down box. Email option should be selected for Email configuration.

Incoming Mail Server

The incoming mail server configuration can be entered and this field is not mandatory.

Incoming Port

Incoming mail server's port can be entered and this field is not mandatory.

Outgoing Server name

The outgoing mail server configuration can be entered and this field is mandatory.

Outgoing port

The outgoing mail server's port can be entered and this field is mandatory.

Username

It denotes username credential to access the mail server.

Password

It denotes the password credential to access the mail server.

Email Id

It denotes the email id in which the recipient receives the email notifications.

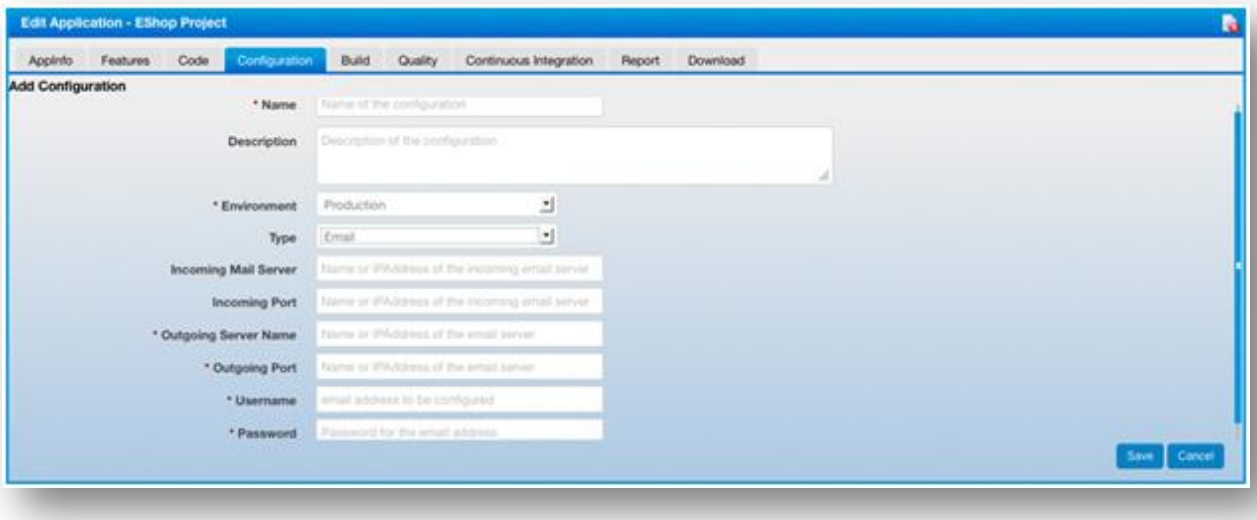


Figure 5-13: Email configuration for an environment

5.3.5 SAP Configurations

Name

It denotes the Name in which a project is created. Maximum text input of the field is restricted to 30 alphanumeric characters and is a mandatory field.

Description

It denotes the Description in which a project is created. This field supports 150 alphanumeric, special characters and is not mandatory.

Environment

Environment created can be selected from the drop down box. If the environment is not created Production environment is selected as default and this field is mandatory.

Type

Configuration type can be selected from the drop down box. SAP option should be selected for SAP configuration.

Search Hosts

This field is to search the host of the SAP Server. It supports alphanumeric, special characters and is not a mandatory field.

SapSvcHost

It denotes the Host name of the SAP Server in which a project is created and is not a mandatory field.

SapSvcPort

This field is the port number of SAP Server. This field supports alphanumeric, special characters and is not mandatory. Port number should be selected such that the port does not run any other application which will not be known by the configuration.

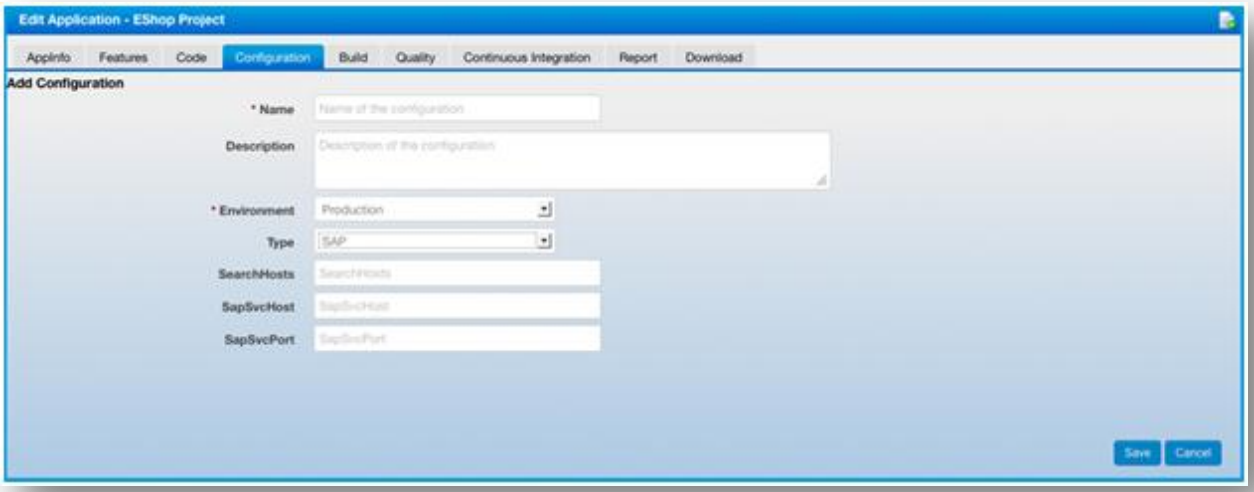


Figure 5-14: SAP configurations for an environment

5.3.6 Dynamic configurations

Name

It denotes the Name in which a project is created. Maximum text input of the field is restricted to 30 alphanumeric characters and is a mandatory field.

Description

It denotes the Description in which a project is created. This field supports 150 alphanumeric, special characters and is not mandatory.

Environment

Environment created can be selected from the drop down box. If the environment is not created Production environment is selected as default and this field is mandatory.

Type

Configuration type can be selected from the drop down box. When the other option is clicked property key and property value field can be entered.

+ & -(Dynamic Key Values)

By clicking "+" users can enter both property key and property the value of the field. Multiple entries can be added by clicking this and can be deleted by clicking '-' button

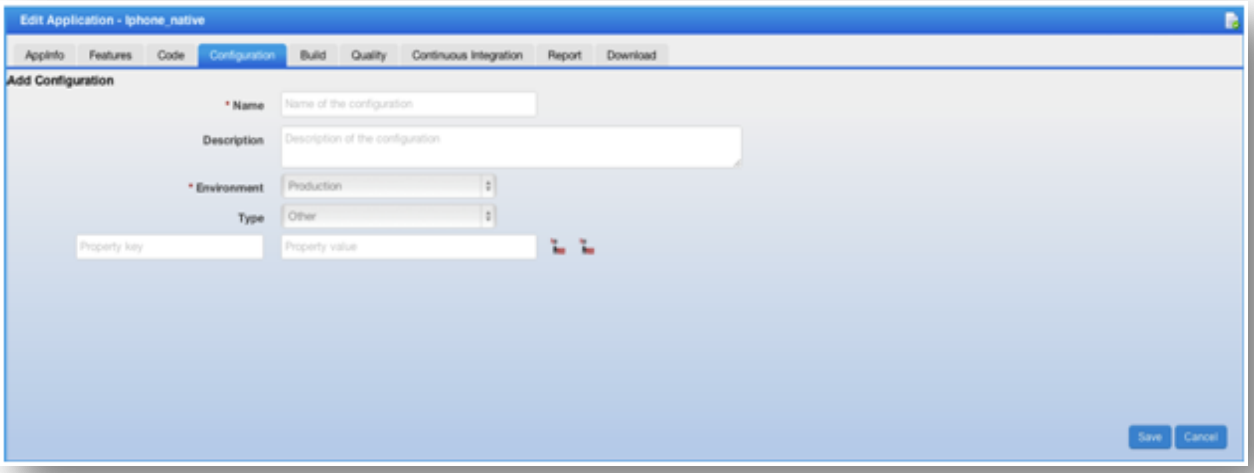


Figure 5-15: Dynamic configurations for an environment

5.4 Global Settings

Global settings is the method of creating configurations for server, database, email and web service globally and use it for different projects which requires the same configurations. Phresco also allows creating an environment and its configurations globally and reuse it for the other projects to make the work easier.

Global settings can be configured by clicking Settings->Add for Server, Web Service, email and Database while for environment click Settings->Environment.

5.5 Build Generation

The process of converting source code files into standalone software artifacts to run on a computer. The important process of the build generation is converting a source code into executable codes.

Project build process can be selected from Applications->project created->Build->Generate Build. A pop up box appears with environment selection, show settings, show error and Hide log.

Phresco allows user to name the build and also number the build that should be generated. The names or numbers can be provided by the users which may contain the version numbers or any other name related to the build.

Environment Selection

Multiple environments can be selected to build a project in the environment field and production environment will be already selected as default.

Show settings

The global configurations for server, database, web service and email can be selected using the show settings option.

Show Error

Show error checkbox provides the error information along with the log for build generation.

Hide Log

Hide log hides the log console and provides only the error message while generating the build.

Skip Unit Test

Phresco enables skip unit tests where the unit tests will be skipped on building a project. This consumes less time on building a project.

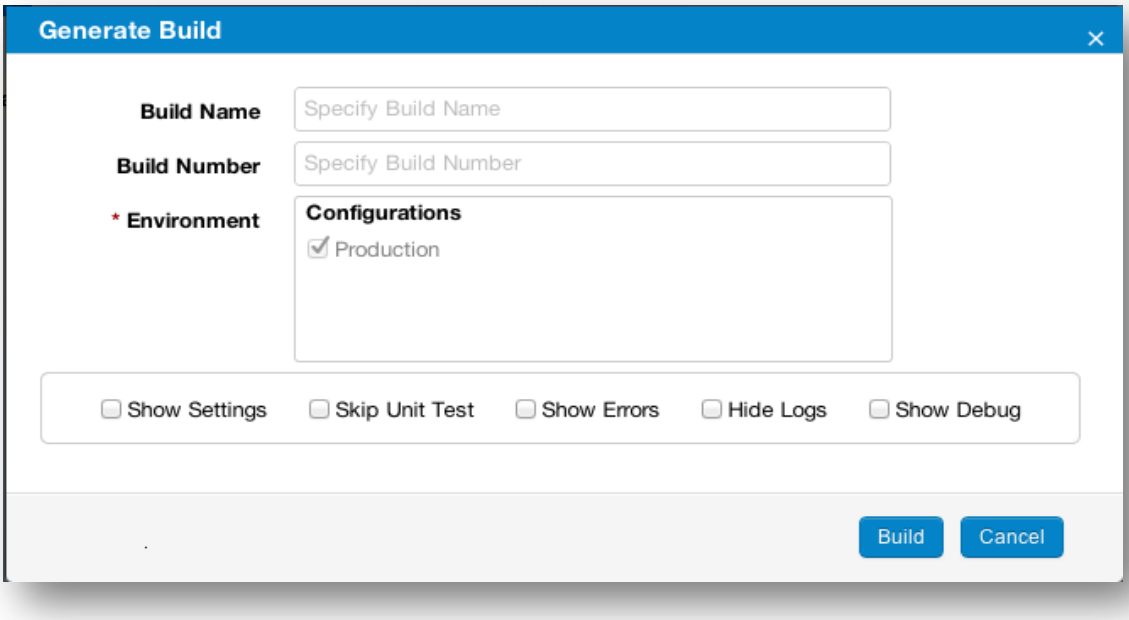


Figure 5-16: Pop up for generate build

5.6 Project Deployment

Project deployment is interpreted as a general process that should be customized according to the requirement and characteristic of a particular project which makes a project available for use. After the build generation process, a deploy icon appears on the screen. On clicking the icon, deploy pop up box appears which has environment selection, show error and hide log check boxes which has the same functionality as in build generation pop up box.

Deployment: Execute SQL

Phresco provides an option to execute the SQL scripts to the configured database while deploying the application. To enable this feature, the Execute SQL checkbox should be selected in the deploy pop up.

- When the environment is selected in the environment field, the database chosen for that corresponding environment will be displayed in the database field.

- The site.sql files depending on the features selected in the features page will be displayed in the box on the left side as a list.
- Users can select the desired site.sql using the arrows present in the user interface and deploy the application.

Note

The prerequisite for executing the SQL script is that, the configured database schema should be created in the database.

5.6.1 Remote Deployment

The application package can be deployed to a local machine or remote machine. Phresco enables the remote deployment concept whenever there is a requirement to deploy to a remote staging or production machine.

Enabling Remote Deployment:

Remote deployment in Phresco can be done by selecting the remote deployment in the server configuration screen and by entering the host (IP address of the remote system), port (server port of the remote system), admin username (admin username of the server) and password (admin password of the server).

Application Servers supported in Phresco for Remote Deployment

- Apache Tomcat – Above 6.x
- JBoss (Restricted to 7.x)
- WebLogic (Restricted to 10.3.6, 12c)

For enabling Remote deployment, following changes should be made in the application servers:

- Apache Tomcat:

Edit tomcat-users.xml in the path apache tomcat /conf/tomcat-users.xml and add the following

```
<role rolename="manager"/>
<role rolename="admin"/>
<role rolename="manager-gui"/>
```

```
        <role rolename="manager-script"/>
        <user username="admin" password="admin" roles="manager-gui, manager-script, admin,
        manager"/>

<plugin>
        <groupId>org.codehaus.mojo</groupId>
        <artifactId>tomcat-maven-plugin</artifactId>
        <version>1.1</version>
        <configuration>
        <username>admin</username>
        <password>admin</password>
        <url> URL </url>
        <path>/${project.build.finalName}</path>
        </configuration>
</plugin>
```

Note:

Server should be up and running during remote deployment.

5.7 Enabling https in Phresco

http:

Phresco supports http (Hypertext Transfer Protocol) which is used for accessing resources normally. Phresco runs in the port number “2468”.

https:

Phresco additionally supports https ((Hypertext Transfer Protocol over Secure Socket Layer). This can be configured for the projects by using the following steps.

1. Edit server.xml in server/conf folder of tomcat installation
2. Add the below details with preferred port, and keystore location

```
<Connector port="<PORT NUMBER>" SSLEnabled="true"
        maxThreads="150" scheme="https" secure="true"
        keystoreFile="<KEYSTORE FILE" keystorePass="<PASSWORD>"
        clientAuth="false" sslProtocol="TLS" />
```

3. Rerun the tomcat server and this will connect Phresco in secure connection

5.8 Project Testing

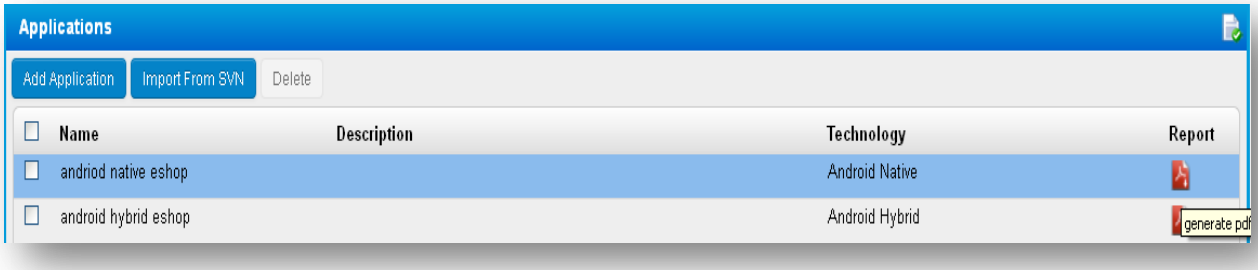
Testing can be stated as the process of validating and verifying that a project

- Meets the requirement that guides its design and development;
- Works as expected; and
- Can be implemented with the same characteristics.

Many type of testing process can be done for a single project and it consumes more time. Using Phresco framework circumvents the troubles associated with testing.

Report generated in PDF

Reports can be generated in the form of PDF which includes total report of all the test cases and also the code validation report when the report icon is clicked. The generated PDF can be sent to the stakeholders when there is a need.



There are four types of testing process available and they are categorized into the following:

5.8.1 Unit Testing5

Unit testing ensures that the developers write proper project implementation. For Unit testing, coverage and reporting Phresco framework uses many tools like JUnit, NUnit, WSUnit, PHPUnit, NodeUnit, and OCUnit for different technologies. The results are provided both in a tabular and graphical output format indicating the ratio of success, failure or error among the test cases.

5.8.2 Functional Testing

Functional testing involves testing on the specifications of a project under test. Functions are tested by feeding them input and examining the output.

Functional testing involves

- Identifying functions the software is expected to perform.
- Creating input data based on the function's specifications.
- Determining output based on the function's specifications.
- Executing test cases.
- Comparing actual outputs and expected outputs.

Functional testing uses tools like selenium, Robotium and Xcode. The result is given through static analysis and test cases. The test cases will point out the error and this will be very useful for the testing team.

5.8.3 Performance Testing

Performance testing determines the performance of a system in terms of receptiveness and solidity under a particular workload. This testing also determines the speed or effectiveness and the response time or throughput of a project.

In Phresco the result of the testing is given through static analysis and test cases.

5.8.4 Load Testing

Load testing refers to modeling the expected usage of a project by simulating the access of multiple users to the same project concurrently. Project should be designed such a way that when a maximum load is reached, a project should not crash. Instead a message saying the load has exceeded should appear. Load and performance testing is usually conducted in a test environment identical to the production environment before a project is permitted for real time usage.

Load testing also uses jMeter. The result is given through static analysis and test cases. The test cases will point out the error and this will be very useful for the testing team.

5.9 Continuous Integration

Continuous Integration process is used for generating the builds, deploying the generated build and testing the deployed project. The build, deploy and test process that are generated manually using build, deploy and test option respectively can be made automatic by scheduling the time. This automation process is done using Jenkins.

Phresco Framework also has the option of sending the build result directly to the developers' inbox.

5.10 Report

Report tab in Phresco is used to generate a site for the project. The generated project site includes the project's reports that were configured in the POM.

Below are the lists of reports that can be configured using phresco:

- Project-Info-Report

  Enabling this gives a report of all the information about the project. This report is common across all the technologies. Under Project-Info-Report there are few option listed. It generates a report on,

  - index: index of the project. This is a default enabled option.
  - modules: list of description of each module used in the project.
  - dependencies: list of dependencies collected from POM.
  - cim: continuous integration management report.
  - scm: source configuration management report.
  - summary: summary of the project.
  - licence: licence that is used in the project.

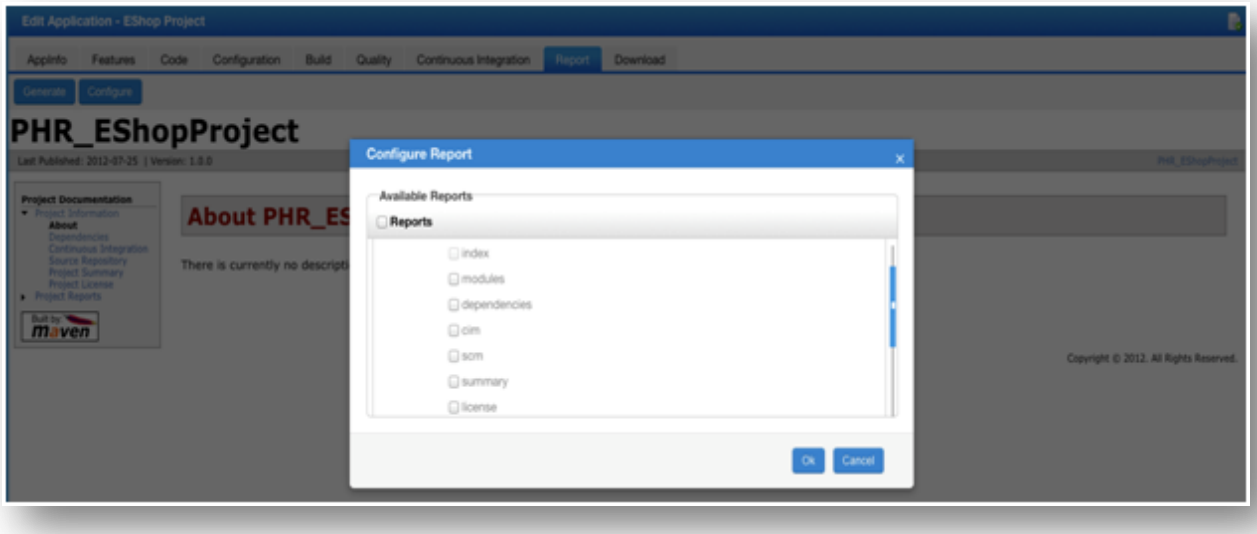


Figure 5-17: lists in Project-Info-Report

- JavaDoc Report

  Enabling this option gives a report of Java source files and produces a JavaDoc which has browsable source code containing hyperlinked cross-references within and across that set of files. This is applicable only for the java technology projects.

- jDepend Report

  Enabling this option generates a report of quality metrics for each java package. Degree of dependencies, its extensibility and reusability is measured and a report is generated on it. This is applicable only for the java technology projects.

- JXR Report

  Enabling this option generates a report of the source code in the html format which can be viewed in the browser. Cross-reference of the project's source is generated which allows the users to find the specific lines of code. This is applicable only for the java technology projects.

- PMD Report

  Enabling this option generates a report of the metrics using the PMD tool. This tool helps in sorting out the test case that are mostly used and the ones that are not important. The unimportant sets can be ruled out to suppress the code. This is applicable only for the java technology projects.

- Surefire Report

  Enabling this option generates a report of the unit test results. This is applicable only for the java technology projects.

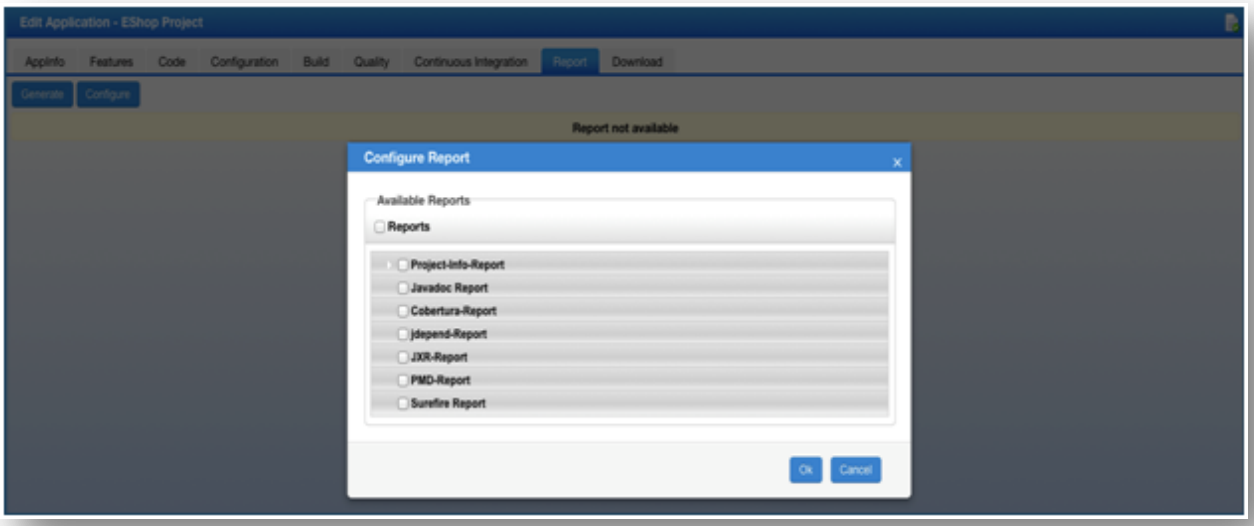


Figure 5-18: Configure Report tab

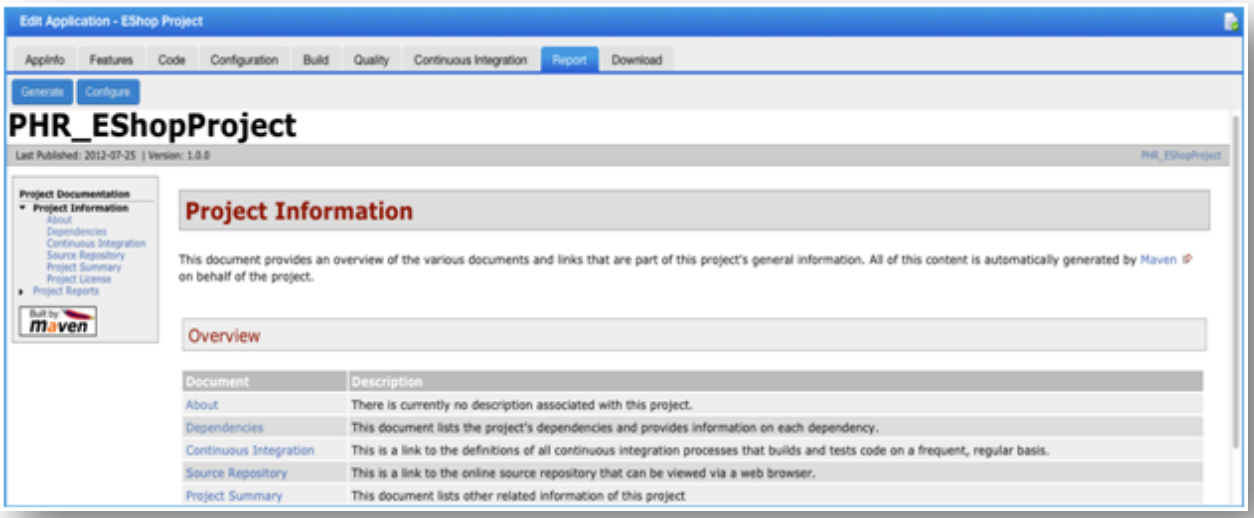


Figure 5-19: Report generated

5.11 Knowledge Repository

A knowledge repository is a computerized system that systematically captures, organizes and categorizes an organization's knowledge.

Knowledge repository in Phresco is the central repository which contains the application types, archetypes and features. It is a common repository where the admin can perform changes or modifications that will impact the user interface.
A Common Knowledge repository exists where any changes made will be reflected across all the account holders.
Each customer will have their own knowledge repository for their artifacts. Any changes in Customer Knowledge repository can be witnessed in the customer's user interface.

5.12 Applications

Using Phresco, web, mobile and web services combination applications can be created based on a project requirement. Created project contains archetype (Project structure), features, testing structure and documents with in it. Features which are provided out of the box can be adapted to a project based on the need. If, the expected features not part of the standard fare can be availed on demand from photon after validation of the license.

Phresco conjointly allows a project leads to import any project into Phresco framework by using the import from SVN. Apart from creating projects using Phresco, developers can also create their project with Phresco standards and import them into Phresco Framework. After importing a project, a project will be ready to proceed with building, deploying, testing etc. Import from SVN will automatically checks out a project in the desired path of the folder {Phresco-framework-1.0\Phresco-framework\workspace\projects}

Also projects to be imported in Phresco framework can be checked out in the desired folder manually. This can be done by copying a project in the path {Phresco-framework-1.0\Phresco-framework\workspace\projects}

Import from SVN

Version controlling system is possible through Phresco where the developers can view and make changes in a project created. Import from SVN tab is used for importing a replica of a project and changes can be made in a project. Many developers can check in a project and make changes which in turn reflect in main project. Conflictions may occur if two developers checks in.

Projects can be imported in the SVN only when they are in Phresco standards. That is .Phresco folder should be present in the structure.

Fields in the screenshot

Repository URL

URL of the SVN project to be imported into Phresco framework.

Other credentials

Other users can use their own credentials and import the projects to the Phresco framework.

Head revision

Head revision is to import the latest revision of a project that is changed in version controlling system.

Revision

Revision is to import a project by providing the revision number. Previously checked in projects can be imported using revision number

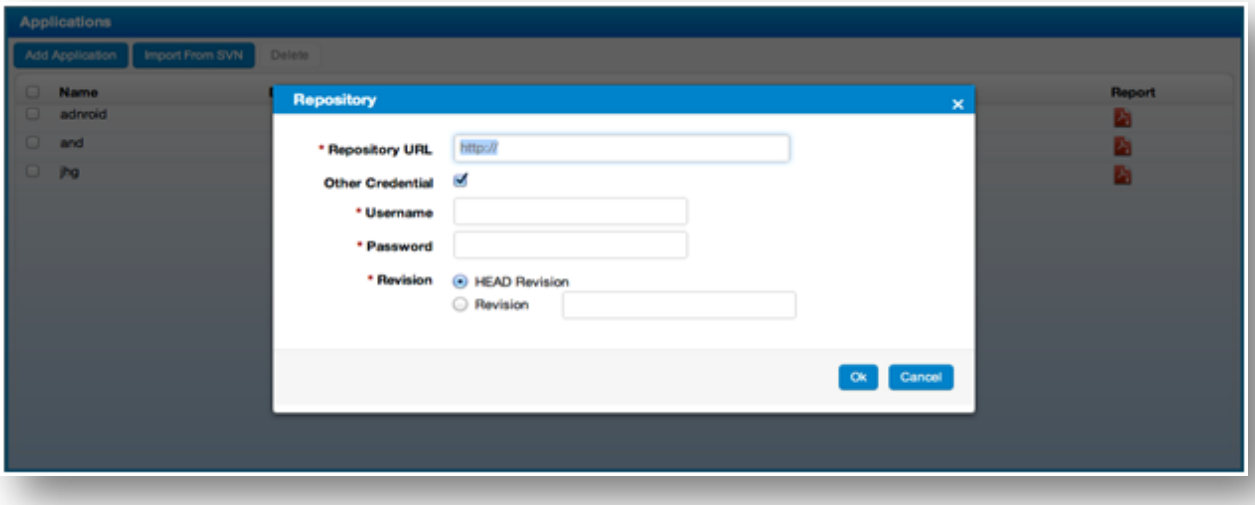


Figure 5-20: ***Import project from SVN***

5.13 Download

Downloads in Phresco are the configurations that are provided to the developers out of the box. Developers can choose the servers or databases or even editors that are available in the download.

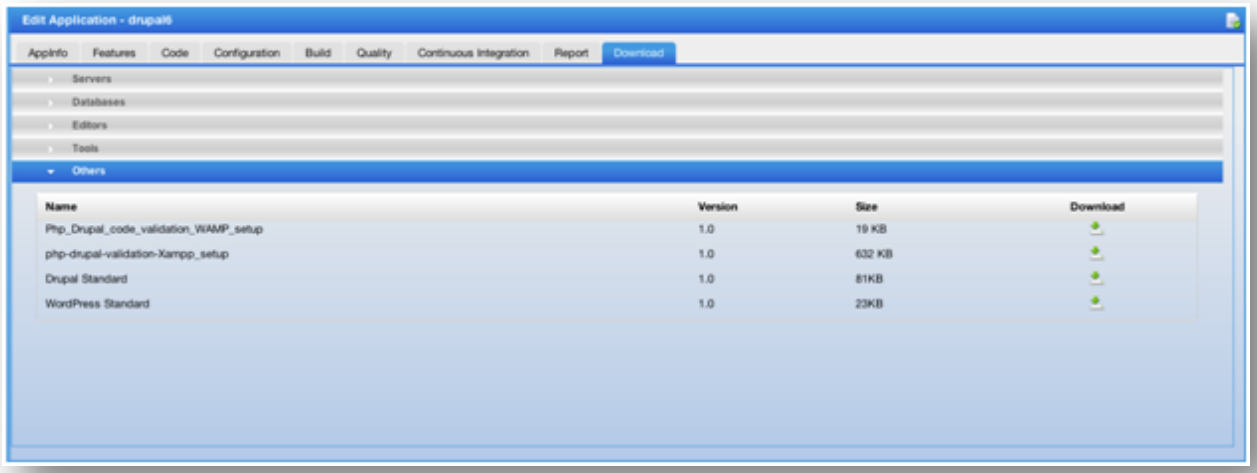


Figure 5-21: Thirdparty downloads

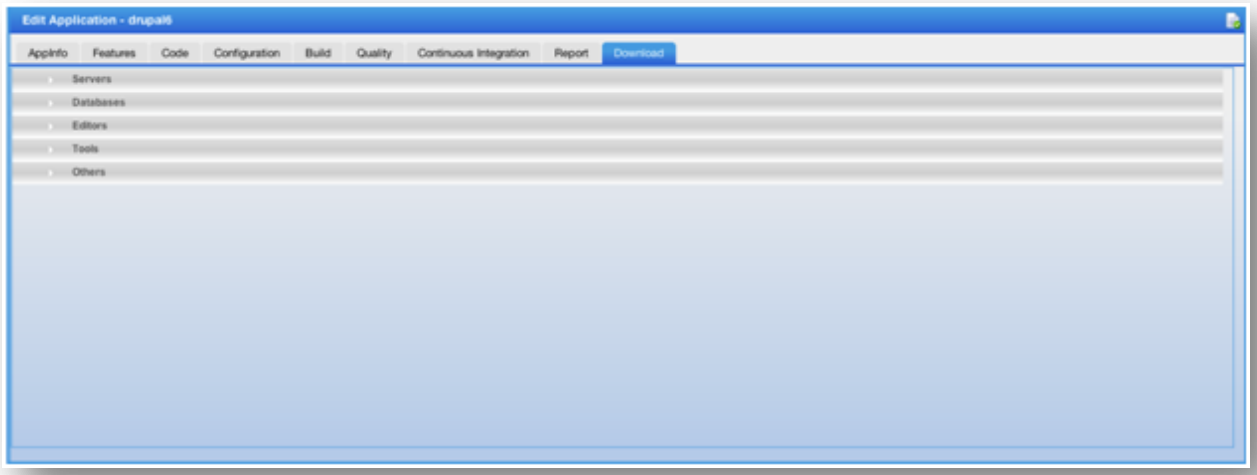


Figure 5-22: Thirdparty downloads

5.14 Phresco Framework Validation

Phresco Framework validation validates the archetype for all the technologies administered by the Nexus repository and generates the report as success or failure. That is, it checks if all the folder structure and files are available for all the technologies created.

If the files are available the success validation report will pop out and failure report will pop out along with the missing file name, if the files are missing.

Note:

If any file is to be deleted and Phresco validation report should succeed, name of the file can be added in Phresco- framework\workspace\projects\PHR_ (any project created)\.Phresco. File name can be mentioned in the exclude files with a forward slash in the beginning. Once the file name is added, the validation will show the success report, even though the files are missing.

5.15 iPhone Prerequisites

For developers

1. Mac machine
2. Mac OS 10.6 & above
3. Xcode tool 4.2 & above
4. IOS SDK 4.0 & above
5. iTunes 10.6 & above

For testers

1. Mac machine
2. Mac OS 10.6 & above
3. Xcode tool 4.2 & above
4. IOS SDK 4.0 & above
5. iTunes 10.6 & above
6. Instrumentation tool for automation test

Deploying Devices

1. iPhone 4, 4s, iPad 2

Note:

- Developers or testers should have registered in develop.apple.com or should have an id for downloading the following software.
- Development certificate is required for deploying a project in the device.

5.16 Android Prerequisites

To build a project:

Phresco supports versions like 2.2, 2.3.3 and 4.0.3 for building a project in ANDROID

To deploy a project:

Project can be deployed depending on the <minSdkVersion> which is defined in the manifest.xml file.

- Manifest.xml code:

```
<?xml version="1.0" encoding="utf-8" ?>

- <manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.photon.Phresco.nativeapp" android:versionCode="1" android:versionName="1.0">

<uses-sdk android:minSdkVersion="7" />

<application android:icon="@drawable/icon" android:label="@string/app_name">

< activity android:name=".activity.MainActivity" android:label="@string/app_name">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>
```

```
</application>

</manifest>
```

Depending on the versions given in the manifest.xml project can be deployed. In the above example

"<uses-sdk android:minSdkVersion="7" /> "

7 is the version mentioned while a project can be deployed in the versions of 7 and above 7.

Table 6: Android API levels

Platforms	API Level
Android 1.5	**3**
Android 1.6	**4**
Android 2.1	**7**
Android 2.2	**8**
Android 2.3- 2.3.2	**9**
Android 2.3.3-2.3.7	**10**
Android 3.0	**11**
Android 3.1	**12**
Android 3.2	**13**
Android 4.0-4.0.2	**14**
Android 4.0.3-4.0.4	**15**

Prerequisites for android for developers and testers

1. User should install Eclipse 3.7[indigo] IDE or above.
2. Also Android SDK for 2.3.3 or higher platform (API level 10 or higher) should be installed
3. "ANDROID_HOME" environment variable should be set, and should point to android sdk folder /tools and /platform-tools should be system PATH variable.

6 Archetypes

Archetype, provided in Phresco, is an ideal project from which similar projects can be created. Phresco Archetypes help the developers follow the best practices in creating projects by providing basic templates for any technology. Developers can create archetypes and deploy it in their organization's repository which will be available for the use of all developers within their organization. Users can also upload archetypes required for their projects with the help of admin console.

6.1 List of Available Archetypes

Archetypes are classified under three different applications - web, mobile and stand alone applications. Under each applications there are list of Archetypes available as given below.

List of Archetypes for Web Applications

- PHP
- Drupal6
- Drupal7
- SharePoint
- ASP .Net
- WordPress
- Java Standalone
- HTML5 Multichannel YUI Widget
- HTML5 Multichannel jQuery Widget
- HTML5 Mobile Widget
- Html5 YUI Mobile widget
- Html5 jQueryMobile Widget

List of Archetypes for Mobile Applications

- iPhone Native
- iPhone Hybrid
- Android Native
- Android Hybrid

List of Archetypes for Web Services Applications

- Java Web service
- NodeJS

7 Reusable Components

Phresco promotes reusability of components by providing a knowledge repository that systematically captures, organizes and categorizes an organization's knowledge.

Archetype:

Archetype, provided in Phresco, is an ideal project from which similar projects can be created. Phresco Archetypes help the developers follow the best practices in creating projects by providing basic templates for any technology. Users can also upload archetypes required for their projects with the help of admin console.

Features:

The below mentioned resources are represented in Phresco as features that can be selected when creating a project.

- Java libraries
- Android libraries
- iPhone libraries
- ASP.NET Libraries
- SharePoint Features
- PHP Modules
- Drupal Modules
- WordPress Modules
- NodeJS Modules
- JS Libraries

Pilot Projects:

Pilot project is an actual project built with the best practices of project development, libraries, components and validated code structures. Phresco has included Pilot projects for most of the technology it supports making sure that fewer efforts are needed in creating new projects. The inclusion is intended to cut maintenance time by synchronizing application and design. Phresco also allows the pilot projects to be re-branded, reused and redelivered when published in the repository.

Third Party Libraries:

Phresco's repository server is the main vital repository and any changes and modifications made by the administrators have a significant impact on the framework's user interface. It is a common knowledge repository that controls the

content accessed by the entire team. Organizations can move the artifacts and features that they desire to reuse across platforms in to the repository.

What happens when you select a pilot project?

Once a pilot project for the desired technology is selected, it will be added into <PHRESCO_HOME>/workspace/projects. A projects a user may create using Phresco archetypes can also be found in the above location.

What happens when you select an Archetype?

Archetypes provided by Phresco can be adapted to the user's project and the completed project can be hosted in Phresco's repository for access by other projects in the organization.

7.1 What happens when you select a feature in your project?

Table 7: Features and Js Libraries for the Technologies

Technology	Selecting a feature	Selecting a JS library
Drupal	The modules will be added into <PROJECT_HOME>/source/sites/all/modules	N/A
SharePoint	The features will be added into <PROJECT_HOME>/source	N/A
NodeJS	The modules will be added into <PROJECT_HOME>/source/node_modules	The modules will be added into <PROJECT_HOME>/source/lib
iPhone Native	The libraries will be added into <PROJECT_HOME>/source/Thirdparty	N/A
iPhone Hybrid	The libraries will be added into <PROJECT_HOME>/source/Thirdparty	N/A
PHP	The libraries will be added into <PROJECT_HOME>/source	The libraries will be added into <PROJECT_HOME>/source/public_html/js

WordPress	The modules will be added into <PROJECT_HOME>/source/wp-content	N/A
ASP.NET	The features will be added into <PROJECT_HOME>/source/src	N/A

Note:

For Java, Java WebService, HTML5 and Android the features are updated in the pom.xml as dependencies.

---

8 Testing

Phresco provides standardized structure for testing all the technologies.

8.1 Test Cases for Java Technology

8.1.1 Unit Test

8.1.1.1 Structure of AllTest and Test Cases

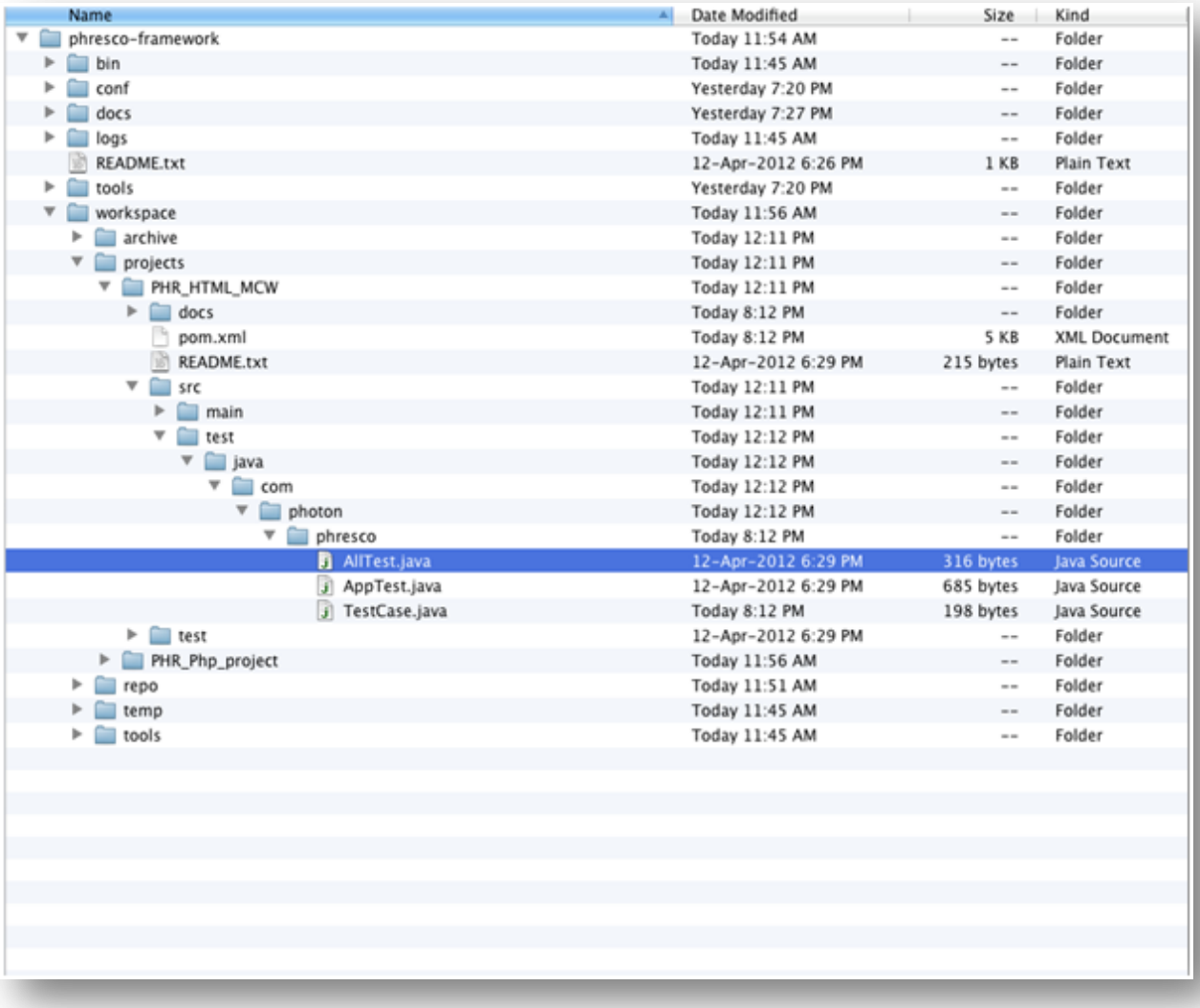


Figure 8-1: Java unit tests structure

a. **AllTest :** AllTest is the root file that carries all the test cases to initiate the testing process. Each test case can be called separately to run the unit test.

b. **Test case:** In unit test, Test cases are written in order to test the source code.

8.1.1.2 Existing Test Cases Out Of the Box in Phresco

AllTest for Java Technology

AllTest contains a bunch of test cases, each of which performs a unique scenario on the application. Test developers can start writing new suite classes and test cases by following the syntax and structure of Phresco frameworks out of the box class structure. AllTest class calls all the suite classes and the test cases within it. Once suite classes and test cases are written developers can execute those from Phresco Framework and can see the report. Following is the AllTest file which shows up how to add / include the class inside it.

```
package com.photon.Phresco;

import junit.framework.Test;
import junit.framework.TestSuite;


public class AllTest {

	public static Test suite() {
		TestSuite suite = new TestSuite(AllTest.class.getName());
		//$JUnit-BEGIN$
		suite.addTestSuite(AppTest.class);
		//$JUnit-END$
		return suite;
	}

}
```

Test Case Example for AppTest

Test cases examine all the aspects including inputs and outputs. It gives the detailed steps that should to be followed during the testing process. Testing process follows only the steps that have been written for each test case.

```
package com.photon.Phresco;

import junit.framework.Test;
import junit.framework.TestCase;
import junit.framework.TestSuite;

/**
 * Unit test for simple App.
 */
public class AppTest
    extends TestCase{

    /**
     * Create the test case
     *
     * @param testName name of the test case
     */
    public AppTest( String testName ) {

        super( testName );
        System.out.println("Printed");
    }

    /**
     * @return the suite of tests being tested
     */
    public static Test suite() {

        return new TestSuite( AppTest.class );
    }

    /**
     * Rigourous Test :-)
     */
    public void testApp() {
        assertTrue( true );
    }
}
```

8.1.1.3 Report generated after execution

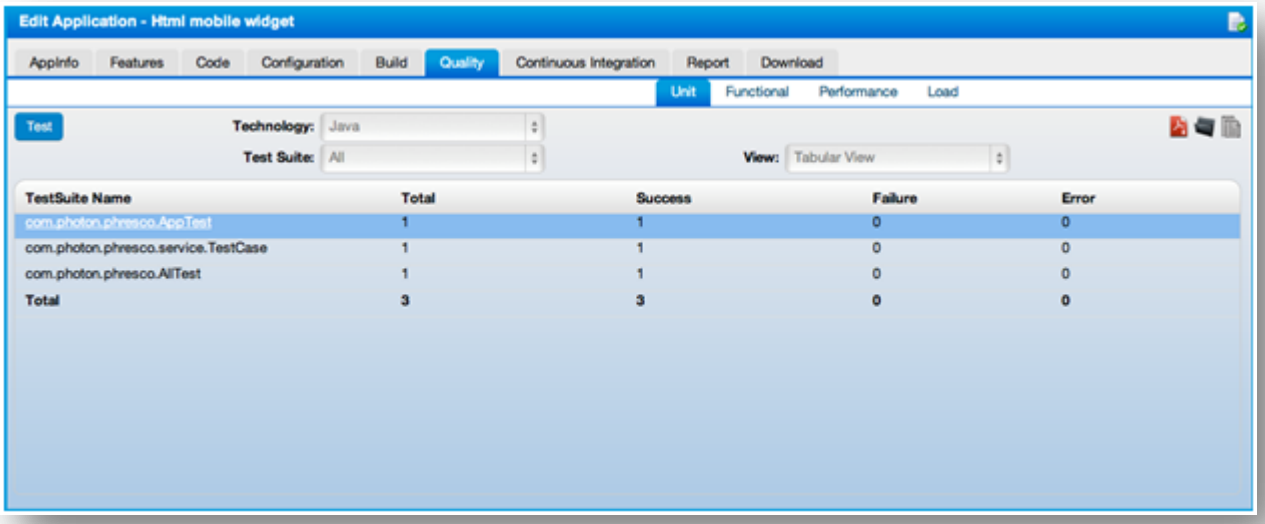


Figure 8-2: HTML5 widget unit test report for all test cases in tabular view

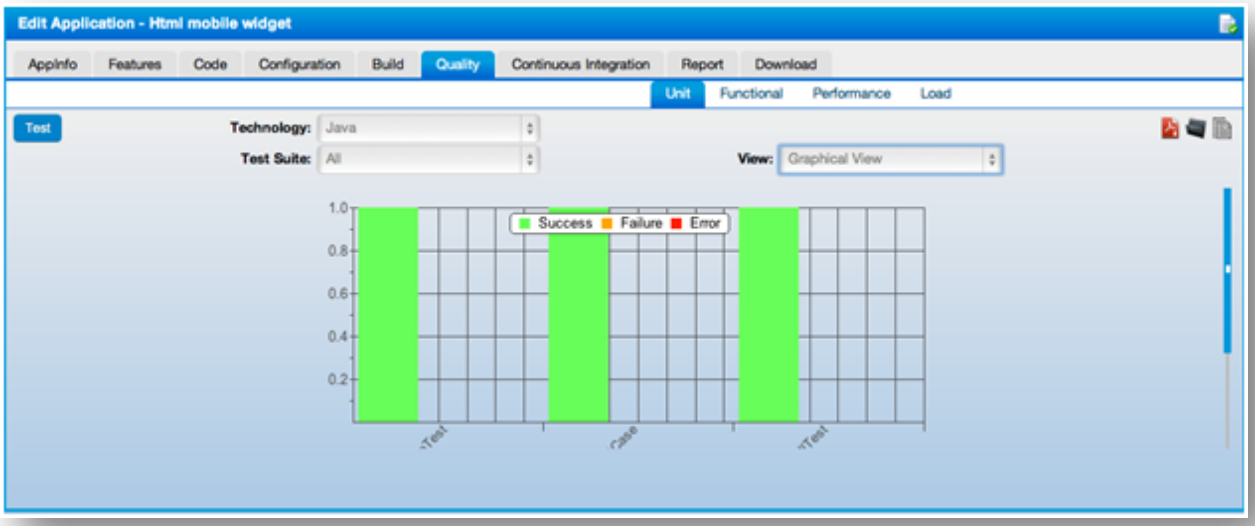


Figure 8-3: *HTML5 widget unit test report for all test cases in graphical view*

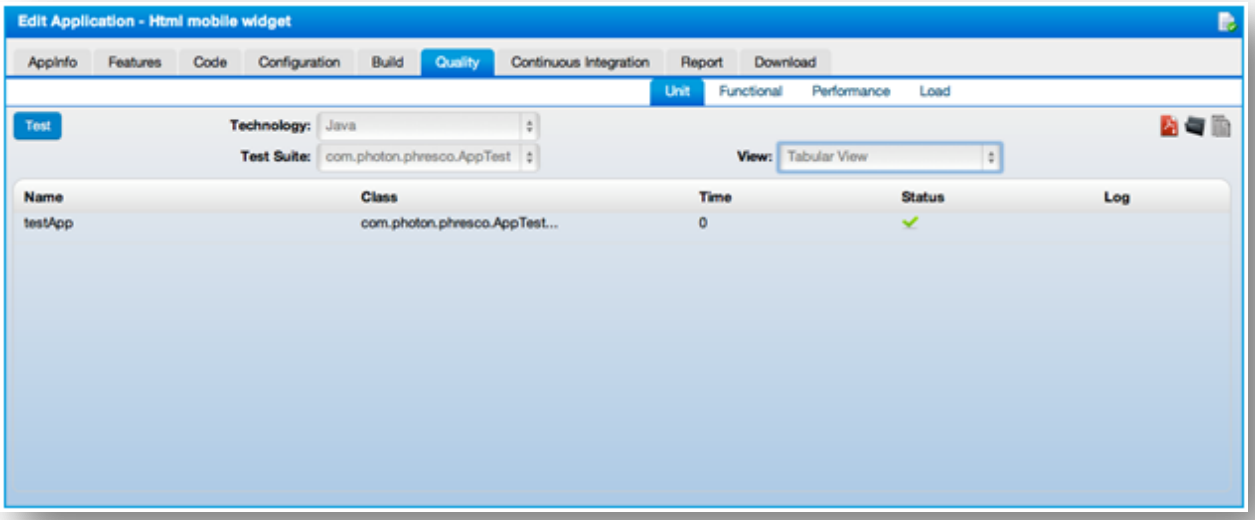


Figure 8-4: ***HTML5 widget unit test report for single test case in tabular view***

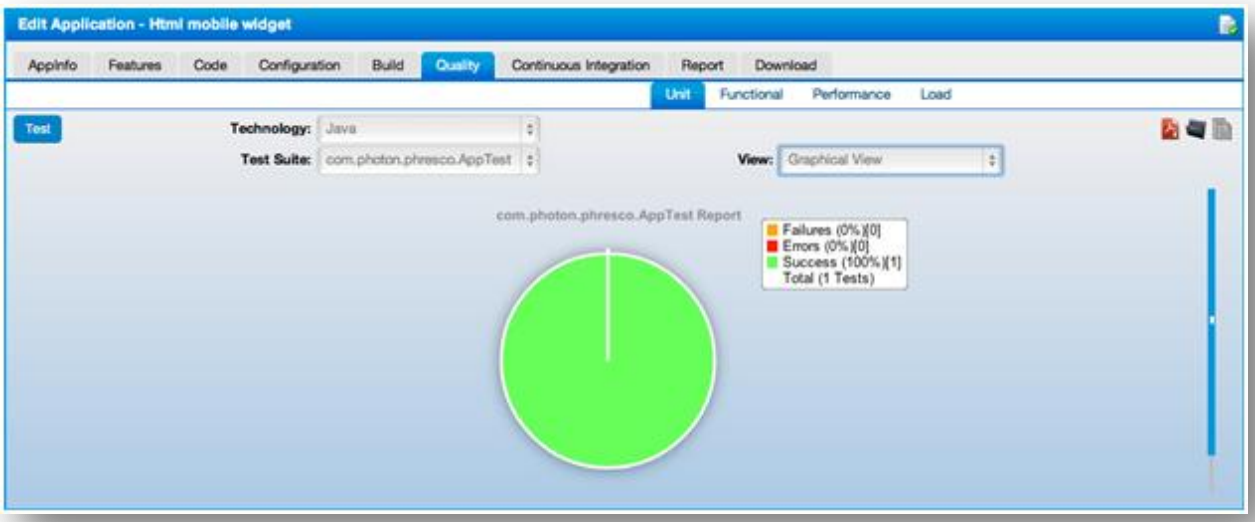


Figure 8-5: ***HTML5 widget unit test report for single test case in graphical view***

8.1.2 Functional Test cases

8.1.2.1 Structure of Functional Test in Java

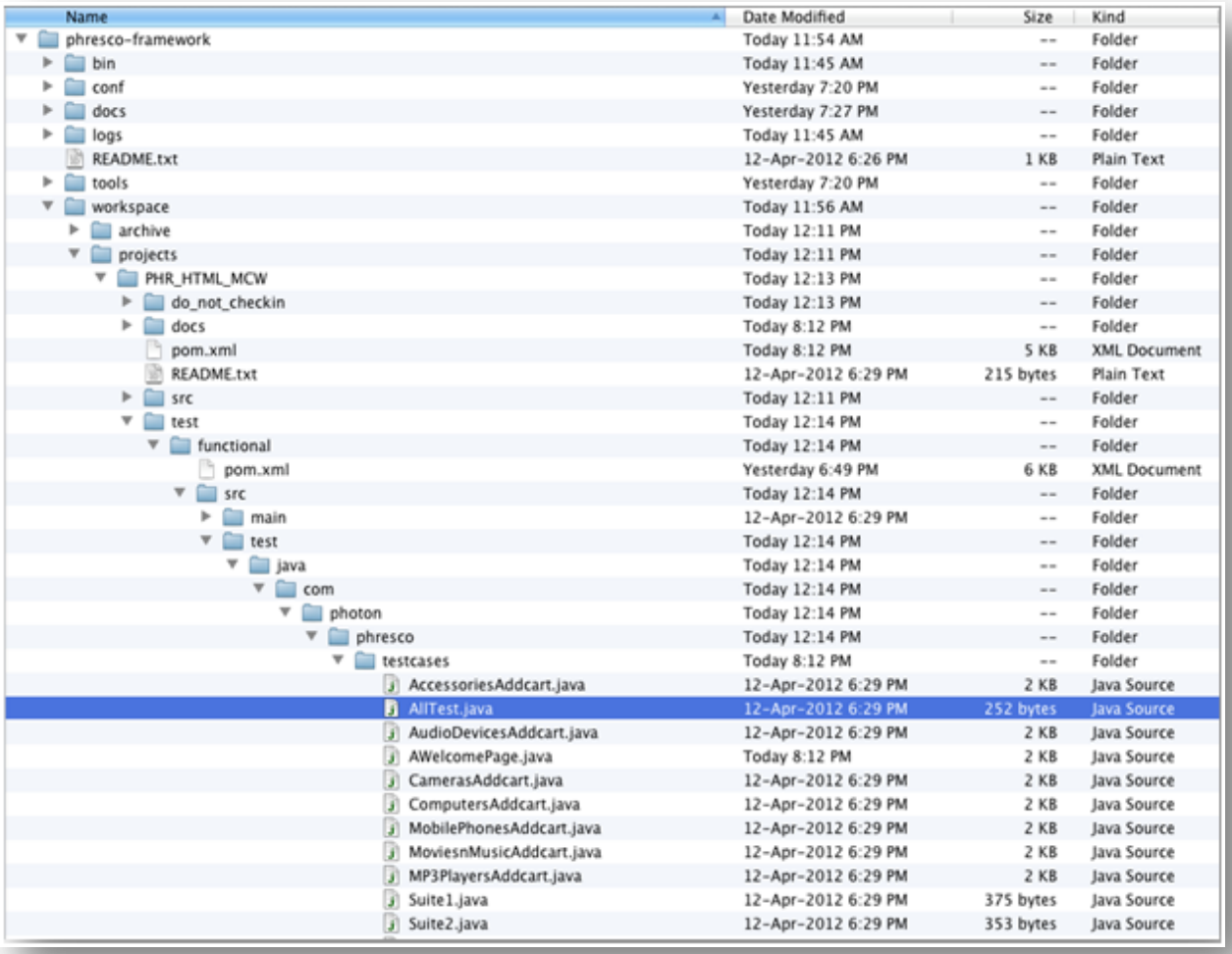


Figure 8-6: Java functional tests structure

a. AllTest: This is a root JUnit suite class that can either call a suite of classes or individual test cases.

b. Suite Class: This is also a JUnit suite class which calls the rest of the individual test cases.

c. Test Cases: These are individual java classes which perform unique testing scenarios against the application.

8.1.2.2 Existing Test Cases and Suites Out Of the Box in Phresco

In Phresco testing Framework a JUnit suite class is named as "AllTest".

The TestSuite contains a bunch of test cases, each of which performs a unique scenario on the application. Test developers can start writing new suite classes and test cases by following the syntax and structure of Phresco frameworks out of the box class structure. AllTest class calls all the suite classes and the test cases within it. Once suite classes and test cases are written developers can execute those from Phresco Framework and can see the report. Following is the AllTest file which shows up how to add / include the class inside it.

```
 Package com.photon.Phresco.testcases

import org.junit.runner.RunWith;
import org.junit.runners.Suite;
import org.junit.runners.Suite.SuiteClasses;

@RunWith(Suite.class)
@SuiteClasses({Suite1.class,Suite2.class})
public class AllTest {
}
```

Suite class

Suite class is a collection of test cases that are intended to be used to test a project code to show that it has some specified set of behaviors. A suite class contains detailed instructions or goals for each collection of test cases. New test cases can also be added by using the same syntax.

```
package com.photon.Phresco.testcases;
import org.junit.runner.RunWith;
import org.junit.runners.Suite;
import org.junit.runners.Suite.SuiteClasses;

@RunWith(Suite.class)
@SuiteClasses({ WelcomePage.class,TeleVisionAddcart.class,
                ComputersAddcart.class,
MobilePhonesAddcart.class,AudioDevicesAddcart.class, CamerasAddcart.class

})
public class Suite1 {

}
```

Test cases

A test case is a set of conditions under which a tester will determine whether an application is meeting the requirement. This helps the user to verify a particular functionality during the testing process. Elements can be identified and screen shots can be captured when the test fails.

```
package com.photon.Phresco.testcases;

import java.io.IOException;

import junit.framework.TestCase;

import org.junit.Test;
//import static org.testng.AssertJUnit.*;
import org.openqa.selenium.server.SeleniumServer;

import com.photon.Phresco.Screens.MenuScreen;
import com.photon.Phresco.Screens.WelcomeScreen;
import com.photon.Phresco.selenium.report.Reporter;
import com.thoughtworks.selenium.Selenium;
import com.photon.Phresco.uiconstants.PhrescoUiConstants;

public class AWelcomePage extends TestCase {

 private SeleniumServer serv;
 protected Selenium selenium;
 private PhrescoUiConstants phrsc;
 private int SELENIUM_PORT;
 private String browserAppends;

 @Test
 public void testWel() throws InterruptedException, IOException, Exception {
            try {
                        phrsc = new PhrescoUiConstants();
                        String serverURL = phrsc.PROTOCOL + "://"
                                    + phrsc.HOST + ":"
                                    + phrsc.PORT + "/";
                        browserAppends = "*" + phrsc.BROWSER;
                        assertNotNull("Browser name should not be null",browserAppends);
                        SELENIUM_PORT = Integer.parseInt(phrsc.SERVER_PORT);
                        assertNotNull("selenium-port number should not be null",
                                    SELENIUM_PORT);
                        WelcomeScreen wel=new WelcomeScreen(phrsc.SERVER_HOST,
SELENIUM_PORT,
                                    browserAppends, serverURL, phrsc.SPEED,
                                    phrsc.CONTEXT );
                        assertNotNull(wel);
                        MenuScreen menu = wel.menuScreen();
          assertNotNull(menu);
```

```
                } catch (Exception t) {
                        t.printStackTrace();
                        System.out.println("ScreenCaptured");
                        selenium.captureEntirePageScreenshot("\\WelPageFails.png",
                                        "background=#CCFFDD");
                }
        }

        @Override
        public void setUp() throws Exception {

                serv = new SeleniumServer();
                try {
                        serv.start();
                } catch (Exception e) {
                        clean();
                        throw e;
                }
        }

        @Override
        public void tearDown() {
                clean();
        }

        private void clean() {
                if (serv != null) {
                        serv.stop();
                }
                if (selenium != null) {
                        selenium.stop();
                }
        }

}
```

8.1.2.3 To Add A New Test Suite in Alltest Class

You can add a new test suite to the AllTest class as follows.

Example

```
package com.photon.Phresco.testcases;

import org.junit.runner.RunWith;
import org.junit.runners.Suite;
import org.junit.runners.Suite.SuiteClasses;

@RunWith(Suite.class)
@SuiteClasses({Suite1.class,Suite2.class})
```

```
public class AllTest {
}
```

8.1.2.4 To Add New Test Case in Test Suite

You can add a new test case to the Test suite using the following method. Here is an example for creating a new test case in the name “CamerasAddcart”

```
package com.photon.Phresco.testcases;

import org.junit.runner.RunWith;
import org.junit.runners.Suite;
import org.junit.runners.Suite.SuiteClasses;

@RunWith(Suite.class)
@SuiteClasses({ WelcomePage.class,TeleVisionAddcart.class,ComputersAddcart.class,
                MobilePhonesAddcart.class,AudioDevicesAddcart.class,
                CamerasAddcart.class
                 })
public class Suite1 {

}
```

8.1.2.5 Report generated after execution

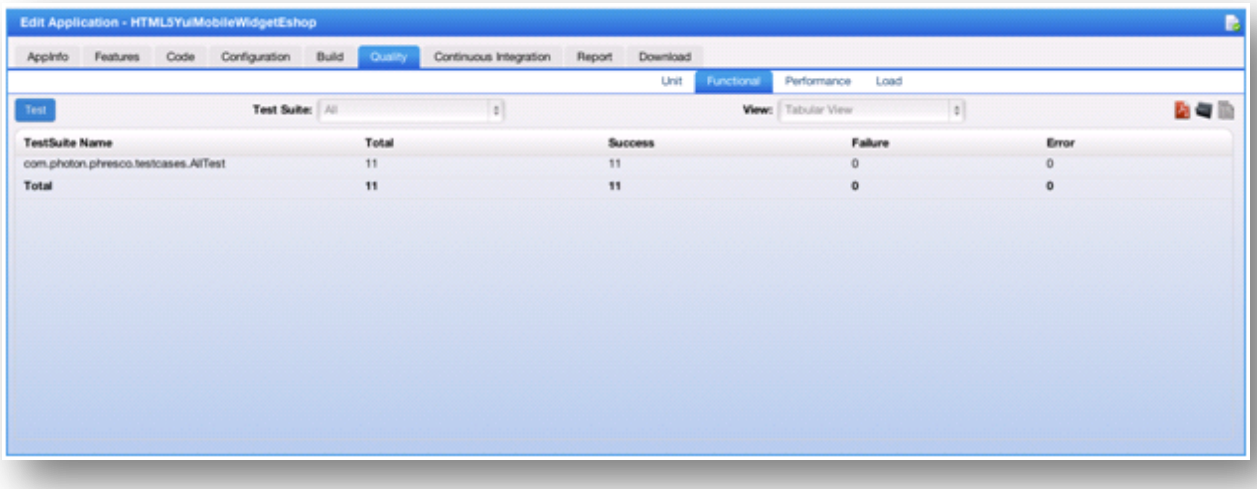


Figure 8-7: HTML5 widget functional test report for all test cases in tabular view

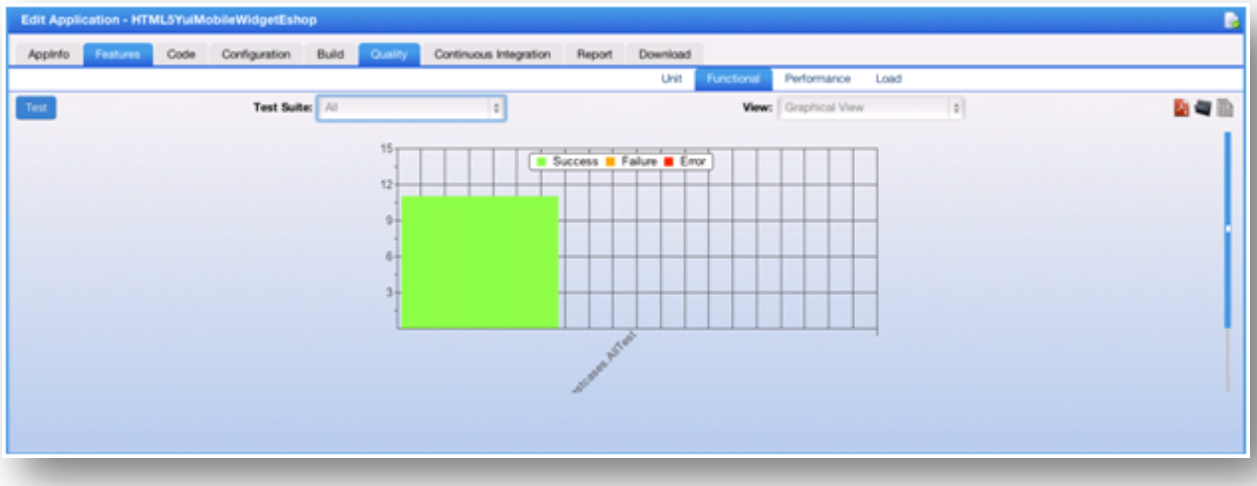


Figure 8-8: ***HTML5 widget functional test report for all test cases in graphical view***

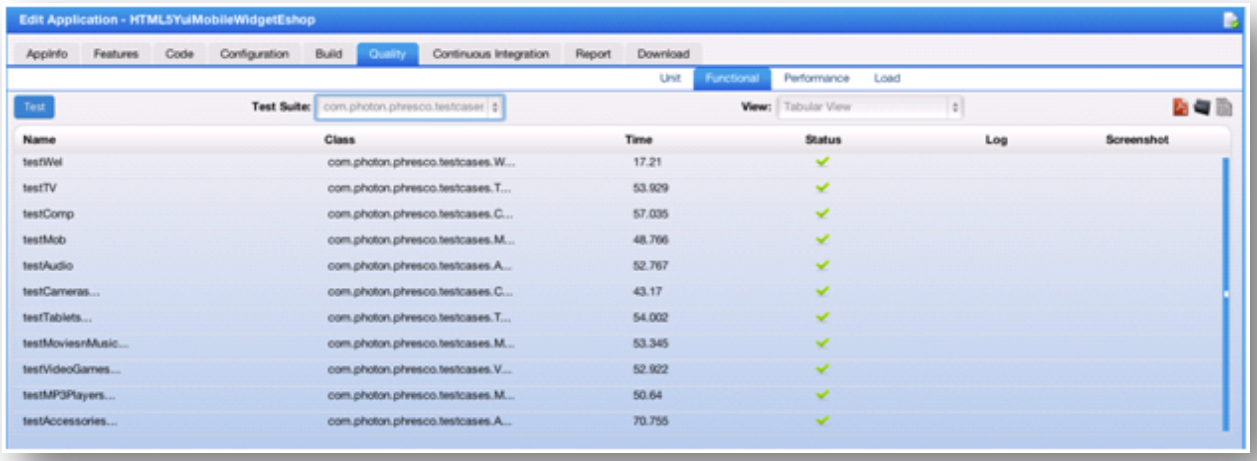


Figure 8-9: ***HTML5 widget functional test report for single test case in tabular view***

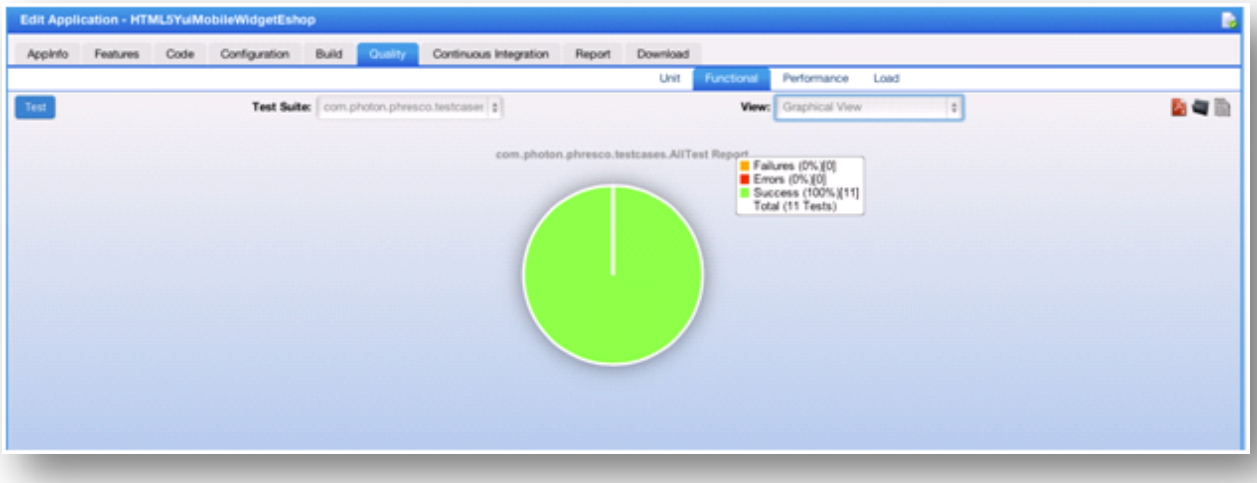


Figure 8-10: ***HTML5 widget functional test report for single test case in graphical view***

8.2 Test Cases for Php Technology

Selenium web driver is used to execute the test cases for Php, Drupal, and WordPress technologies in the latest version.

8.2.1 Unit Test Cases

8.2.1.1 Structure of Alltest and Test Cases

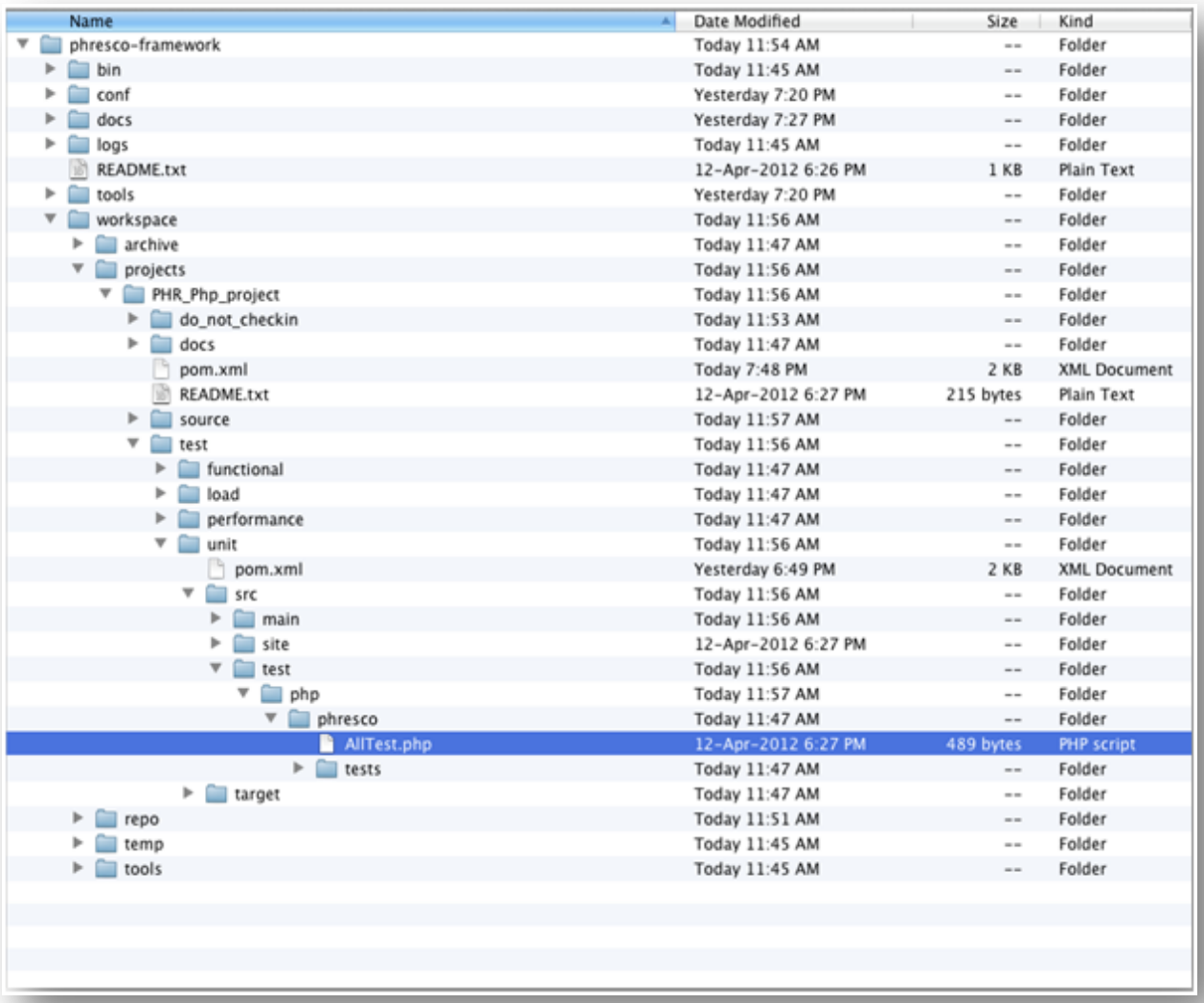


Figure 8-11: PHP unit tests structure

a. **AllTest:** It is the root file that carries all the php suite file to initiate the testing process. Each test case can be called separately to run the unit test.

b. **Test cases**: These are individual test classes which perform unique testing scenarios against the application.

8.2.1.2 Existing Test Cases and Suites Out Of the Box in Phresco

AllTest for PHP

AllTest contains a bunch of test cases, each of which performs a unique scenario on the application. Test developers can start writing new suite classes and test cases by following the syntax and structure of Phresco framework's out of the box class structure. AllTest class calls all the suite classes and the test cases within it. Once suite classes and test cases are written developers can execute those from Phresco Framework and can see the report. Following is the AllTest file which shows up how to add / include the class inside it.

```
<? php

require_once 'tests/UsernameValidation.php';
require_once 'tests/DateValidation.php';

class AllTest extends PHPUnit_Framework_TestSuite{

 protected function setUp(){

 }
 public static function suite(){
          $testSuite = new AllTest('Phpunittest');
          $testSuite->addTest(new UsernameValidation("testValidation"));
          $testSuite->addTest(new DateValidation("testDate"));

          return $testSuite;
   }
 protected function tearDown(){

}

}
```

Test case

A test case is a set of conditions under which a tester will determine whether an application is meeting the requirement. This helps the user to verify a particular functionality during the testing process.

This test case is written for testing the characters of the username to be entered manually. The characters are defined in the base screen and only if it matches with manually entered characters, the unit test case will pass. Testing the username is one unit test case and there can be 'n' number of test cases.

```
<?php

require_once 'PHPUnit/Framework.php';
require_once 'Phresco/tests/BaseScreen.php';
class UsernameValidation extends PHPUnit_Framework_TestCase {
      public function setUp() {

         $this->objValidator = new BaseScreen;

      }
      public function testValidation() {

         $this->assertEquals(true,
         $this->objValidator->check_username('jhonson'));
      }
}
?>
```

8.2.1.3 To Add New Test Case in Alltest

You can add new test cases to the AllTest. An example for creating a new test case is given below.

```
public static function suite(){
          $testSuite = new AllTest('Phpunittest');
          $testSuite->addTest(new UsernameValidation("testValidation"));
          $testSuite->addTest(new DateValidation("testDate"));
           $testSuite->addTest(new EmailVerification("testEmail"));
```

8.2.1.4 Report generated after execution

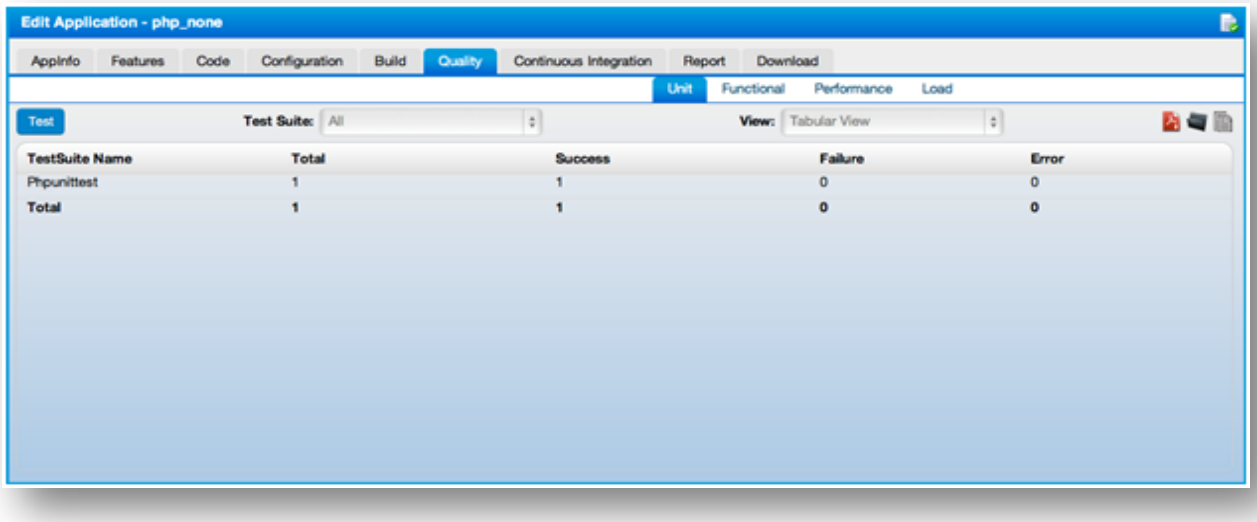


Figure 8-12: PHP unit test report for all test cases in tabular view

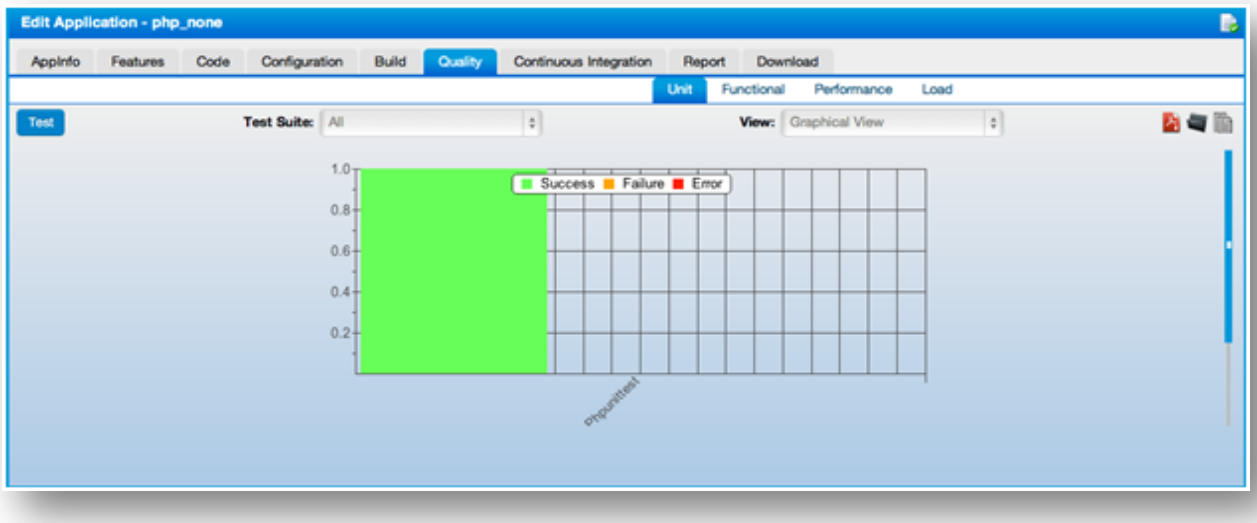


Figure 8-13: PHP unit test report for all test cases in graphical view

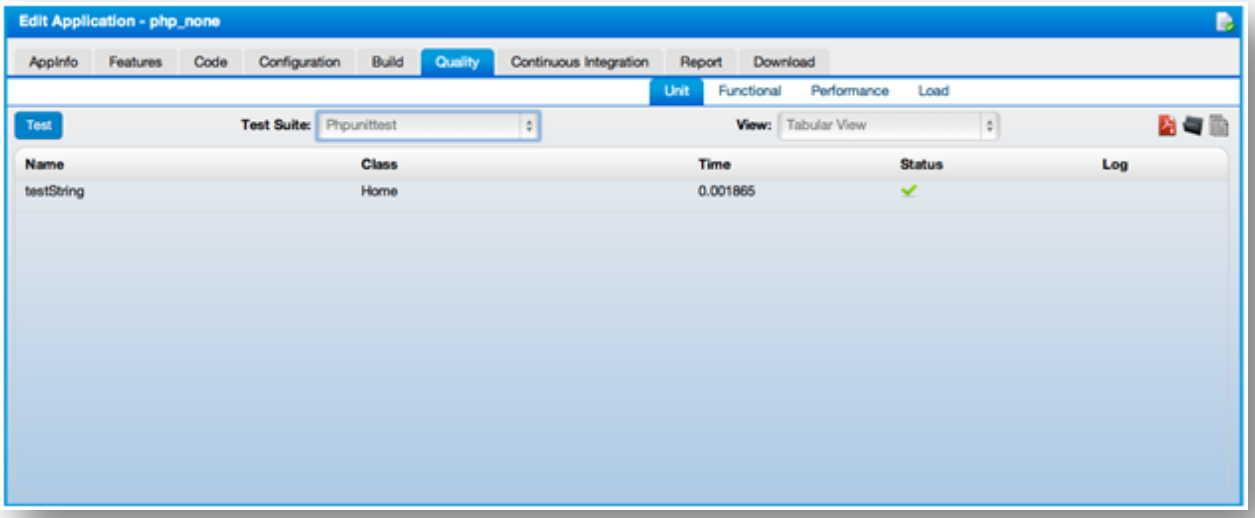


Figure 8-14: PHP unit test report for single test case in tabular view

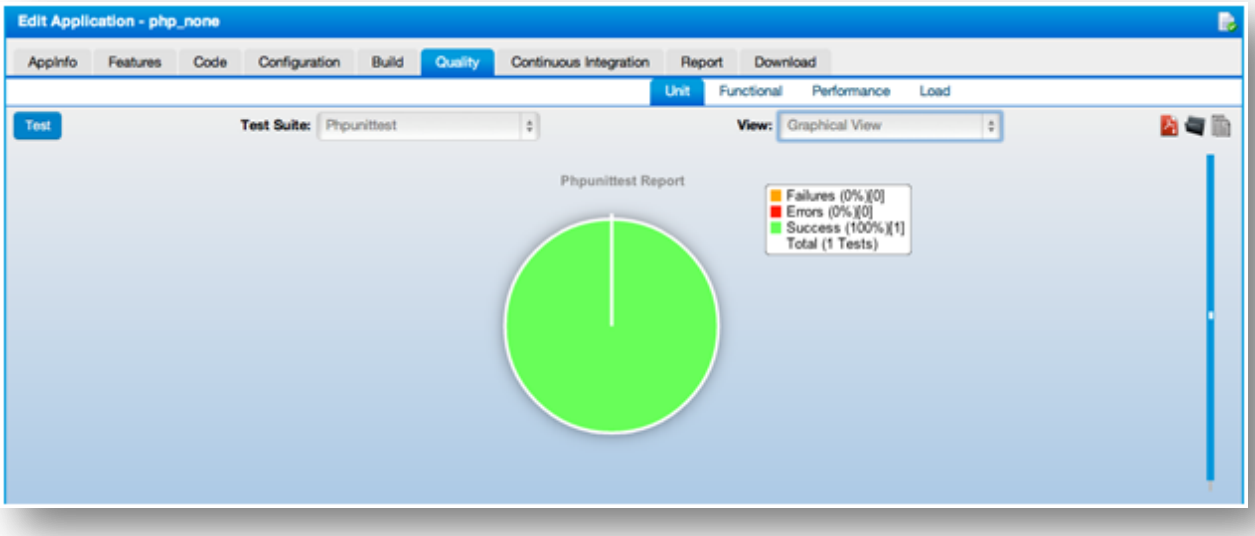


Figure 8-15: PHP unit test report for single test case in graphical view

8.2.2 Functional Test Case

8.2.2.1 Structure of Test Suites and Test Case

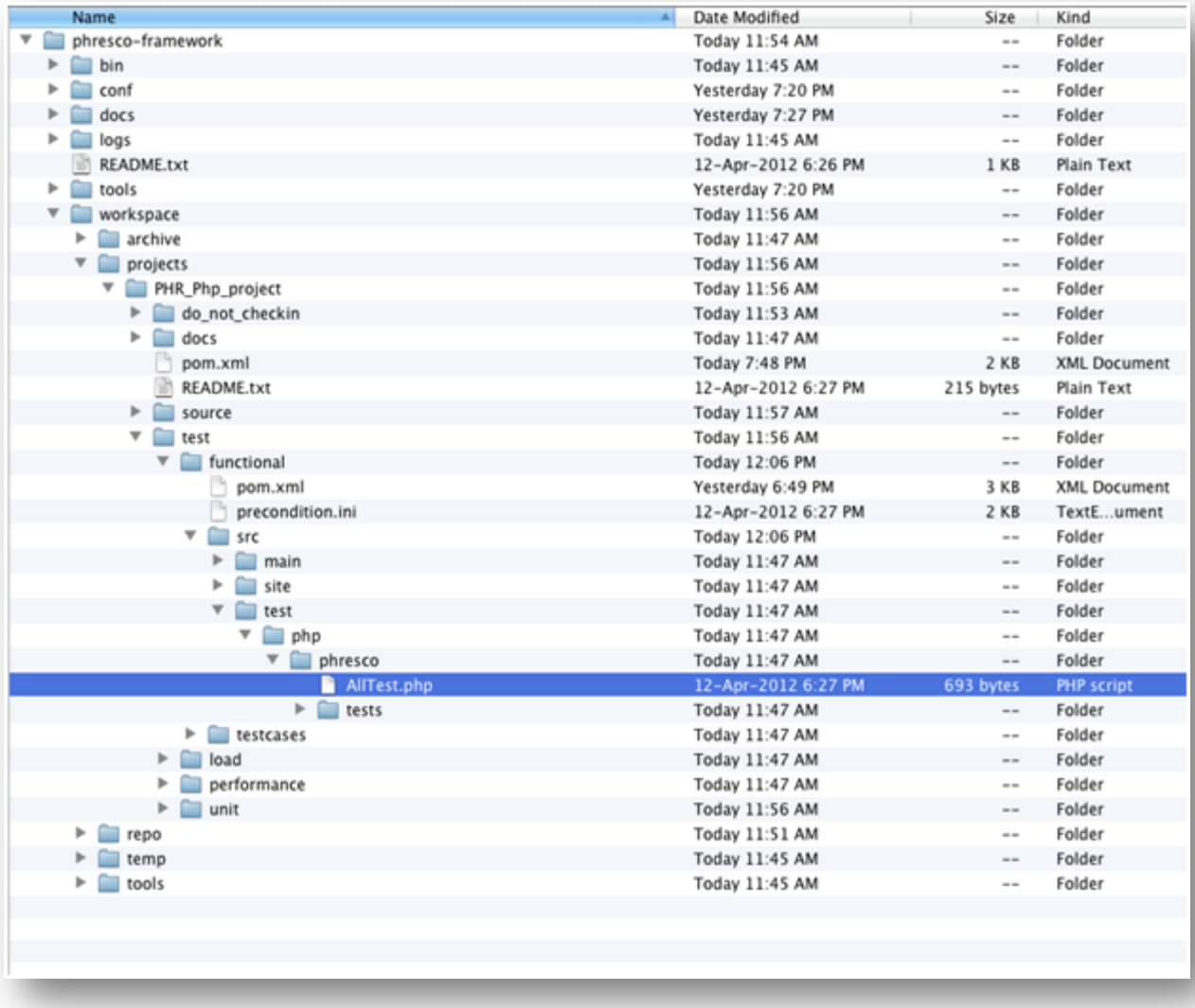


Figure 8-16: PHP functional tests structure

a. **AllTest**: It is the root file that carries all the test suites to initiate the testing.

b. **Test suite**: All the Test suites should be incorporated in the **AllTest**

c. **Test case**: These are individual test classes which perform unique testing scenarios against the application.

8.2.2.2 Existing Test Cases and Suites Out Of the Box in Phresco

Alltest for Php

AllTest contains a bunch of test cases, each of which performs a unique scenario on the application. Test developers can start writing new suite classes and test cases by following the syntax and structure of Phresco framework's out of the box class structure. AllTest class calls all the suite classes and the test cases within it. Once suite classes and test cases are written developers can execute those from Phresco Framework and can see the report. Following is the AllTest file which shows up how to add / include the class inside it.

```
require_once 'tests/UserModules.php';
require_once 'tests/SearchModules.php';
require_once 'tests/DbUpdateModules.php';
require_once 'tests/DbDeleteModules.php';

class AllTest extends PHPUnit_Framework_TestSuite
{
        protected function setUp(){
                parent::setUp();
        }
        public static function suite(){

                $suite = new AllTest();
                $suite->setName('AllTestsuite');
    $suite->addTest(UserModules::suite());
                $suite->addTest(SearchModules::suite());
                $suite->addTest(DbUpdateModules::suite());
                $suite->addTest(DbDeleteModules::suite());
    return $suite;
  }
        protected function tearDown(){
                parent::tearDown();
        }
}
```

Test Suites Example for Usermodules

Test suite is a collection of test cases that are intended to be used to test a project code to show that it has some specified set of behaviors. A test suite contains detailed instructions or goals for each collection of test cases. New test cases can also be added by using the following syntax.

```
require_once 'Register_NewUser.php';
require_once 'LoginLogout_User.php';
require_once 'Account_Update.php';


class UserModules extends PHPUnit_Framework_TestSuite
{
        protected function setUp(){
                parent::setUp();
        }
        public static function suite(){

                $testSuite = new UserModules();
                $testSuite->setName('UserModules');
                $testSuite->addTestSuite('Register_NewUser');
                $testSuite->addTestSuite('LoginLogout_User');
                $testSuite->addTestSuite('Account_Update');
    return $testSuite;
  }
  protected function tearDown(){

        }
}
```

Test Case Example for Testregistration

Test cases examine all the aspects including inputs and outputs. It gives the detailed steps that should to be followed during the testing process. Testing process follows only the steps that have been written for each test case. It also helps in identifying an element and capturing screen shots when the test fails.

```
require_once 'PhpCommonFun.php';
class Register_NewUser extends PhpCommonFun
{
  protected function setUp(){

        parent::setUp();
```

```
	}
	public function testRegisters(){
		parent::Browser();
			$testcases = __FUNCTION__;
			$doc = new DOMDocument();

			$doc->load('test-classes/Phresco/tests/phpsetting.xml');

			$users = $doc->getElementsByTagName("register");

			foreach( $users as $register ){

				$names = $register-
>getElementsByTagName("username");
				$name = $names->item(0)->nodeValue;

				$emails = $register->getElementsByTagName("email");
				$email = $emails->item(0)->nodeValue;

				$passwords = $register-
>getElementsByTagName("password");
				$password = $passwords->item(0)->nodeValue;
			}

			$this->getElement(PHP_REG_LINK,$testcases);
			$this->clickandLoad(PHP_REG_LINK);
			$this->getElement(PHP_REG_UNAME_TBOX,$testcases);
			$this->type(PHP_REG_UNAME_TBOX,$name);
			$this->getElement(PHP_REG_EMAIL_TBOX,$testcases);
			$this->type(PHP_REG_EMAIL_TBOX,$email);
			$this->getElement(PHP_REG_PASS_TBOX,$testcases);
			$this->type(PHP_REG_PASS_TBOX,$password);
			$this->getElement(PHP_REG_SUBMIT,$testcases);
			$this->submit(PHP_REG_SUBMIT,$testcases);
			try{
			$this->assertTrue($this->isTextPresent(PHP_REG_MSG));
			}
			catch (PHPUnit_Framework_AssertionFailedError $e) {
			$this->doCreateScreenShot(__FUNCTION__);

			}
	}
		public function tearDown(){
			$this->closeWindow();
		}
```

8.2.2.3 To Add New Test Suite in Alltest

An example for creating new test suite in the name 'DbDeleteModules'.

```
$suite = new AllTest();
        $suite->setName('AllTestsuite');
         $suite->addTest(UserModules::suite());
        $suite->addTest(SearchModules::suite());
        $suite->addTest(DbUpdateModules::suite());
        $suite->addTest(DbDeleteModules::suite());
```

8.2.2.4 To Add New Test Case in Test Suite

An example for creating a new test case in the name 'testAccountUpdate'

```
$testSuite = new UserModules();
        $testSuite->setName('UserModules');
        $testSuite->addTestSuite('Register_NewUser');
        $testSuite->addTestSuite('LoginLogout_User');
        $testSuite->addTestSuite('Account_Update');
```

8.2.2.5 Report generated after execution

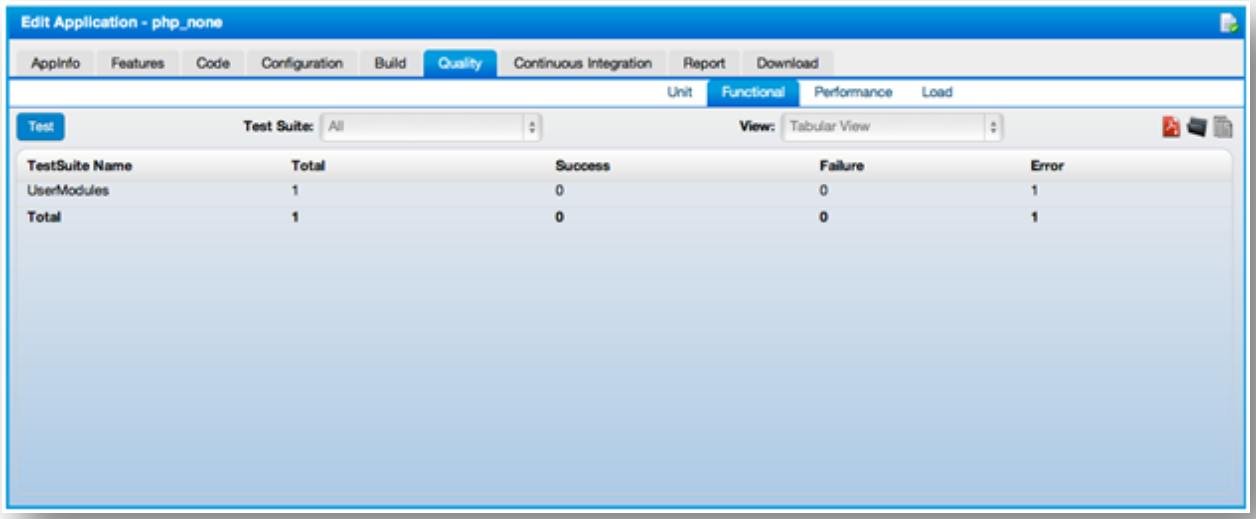


Figure 8-17: PHP functional test report for all test cases in tabular view

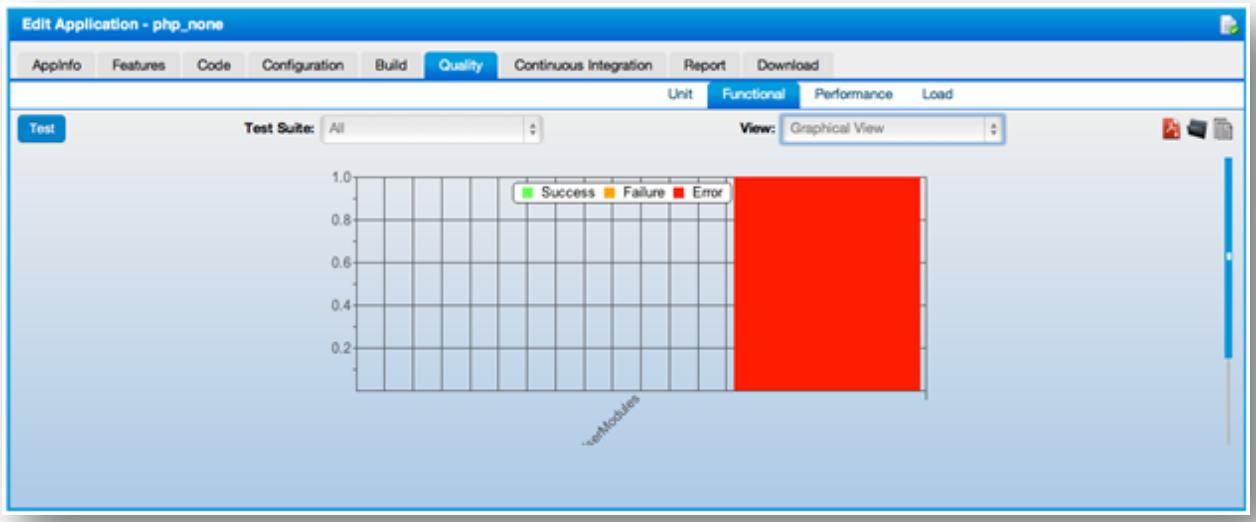


Figure 8-18: PHP functional test report for all test cases in graphical view

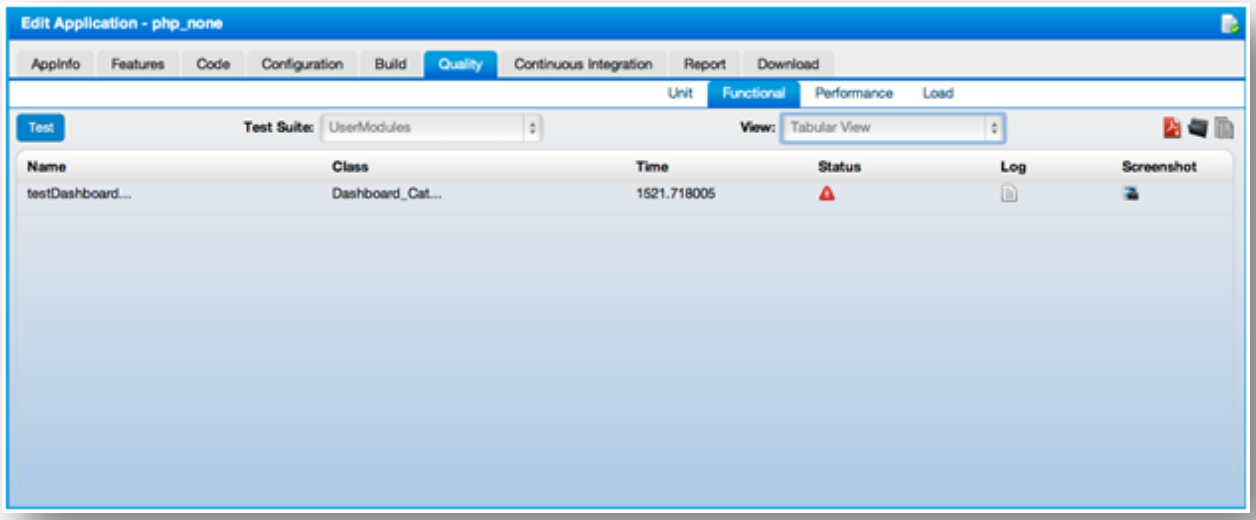


Figure 8-19: PHP functional test report for single test case in tabular view

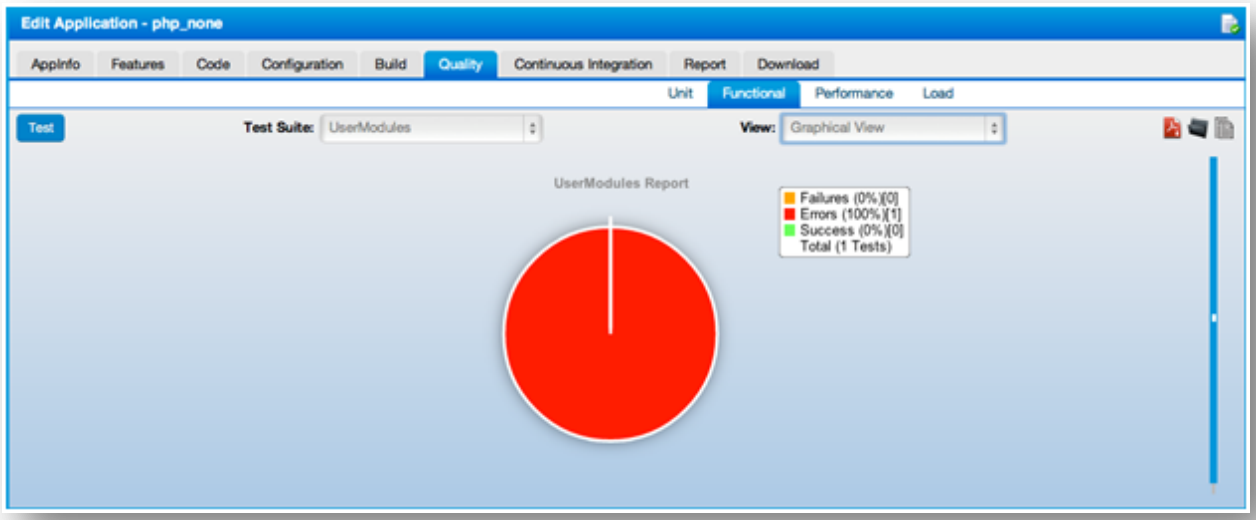


Figure 8-20: PHP functional test report for single test case in graphical view

8.3 Test Cases for SharePoint Technology

8.3.1 Unit Test Case

8.3.1.1 Structure Of Alltest And Test Cases

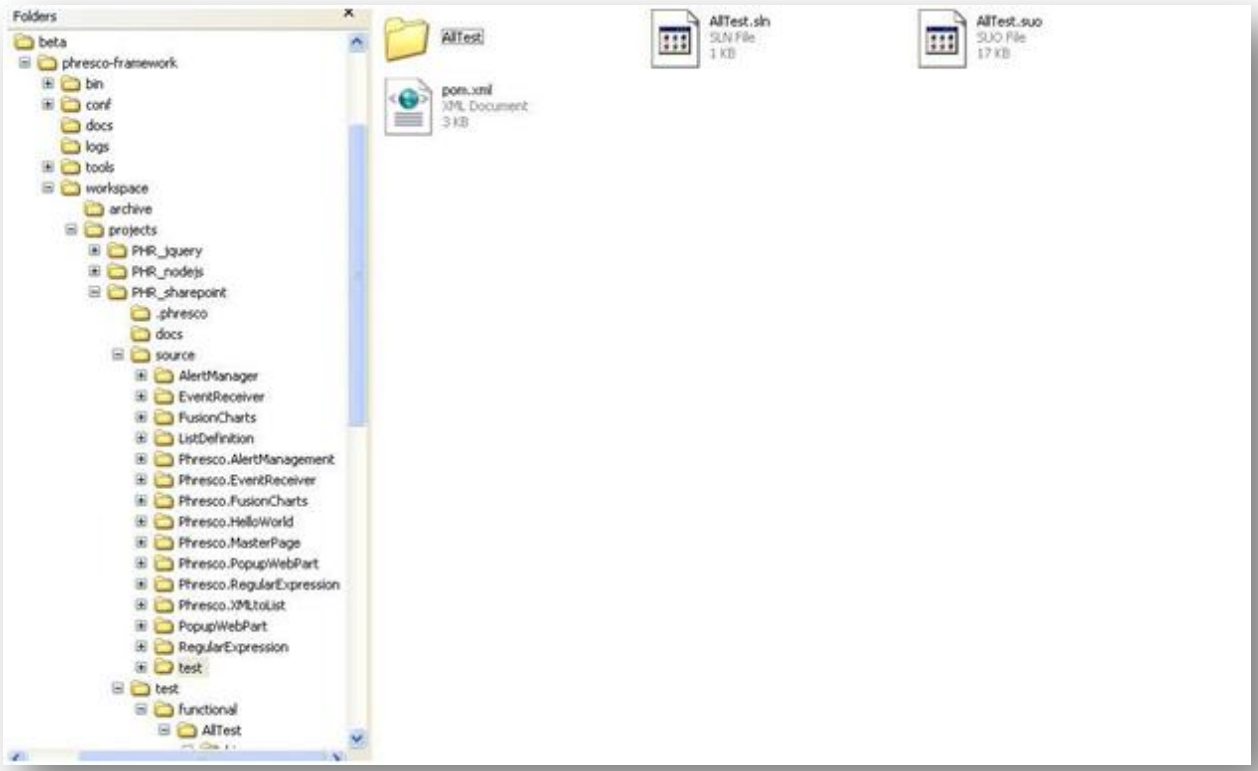


Figure 8-21: SharePoint unit tests structure

a. **AllTest**: This is a root folder/file that carries all the classes to initiate the testing.

b. **Class**: All the classes is incorporated in the AllTest

c. **Test Case**: All the test cases should be added within the Class

Alltest for SharePoint

AllTest contains a bunch of test cases, each of which performs a unique scenario on the application. Test developers can start writing new suite classes and test cases by following the syntax and structure of Phresco framework's out of the box class structure. AllTest class calls all the suite classes and the test cases within it.

Once suite classes and test cases are written developers can execute those from Phresco Framework and can see the report. Following is the AllTest file which shows up how to add / include the class inside it.

Class example

Class is a collection of test cases that are intended to be used to test a project code to show that it has some specified set of behaviors. A class contains detailed instructions or goals for each collection of test cases. New test cases can also be added by using the same syntax.

```
using System;
using System.Web;
using System.Text;
using System.Collections.Generic;
using System.Linq;
using NUnit.Framework;
using Phresco.EventReceiver;

namespace Phresco.UnitTesting{

   [TestFixture]
   public class ItemEventReceiver{

      [Test]
      public void ItemEventReceiverTests() {

         ItemEventReceiver itemEventReceiver = new ItemEventReceiver();
         // string itemEventReceiverMessage = "";
         //Assert.AreEqual("", itemEventReceiver.);
      }
      }

     }
```

Test case

Test cases examine all the aspects including inputs and outputs. It gives the detailed steps that should to be followed during the testing process. Testing process follows only the steps that have been written for each test case.

8.3.1.2 Report generated after execution

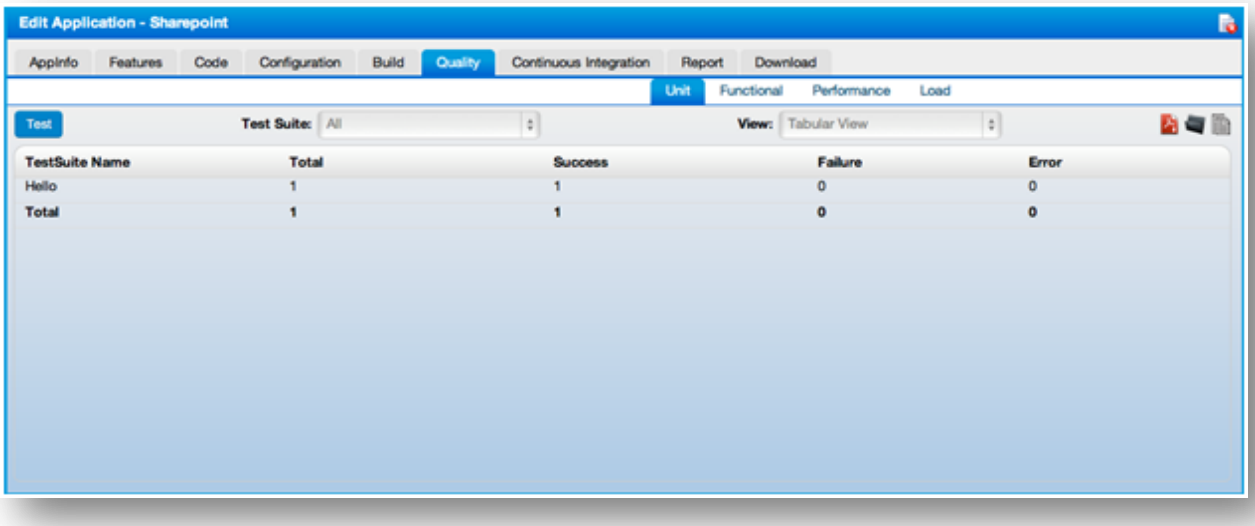


Figure 8-22: SharePoint unit test report for all test cases in tabular view

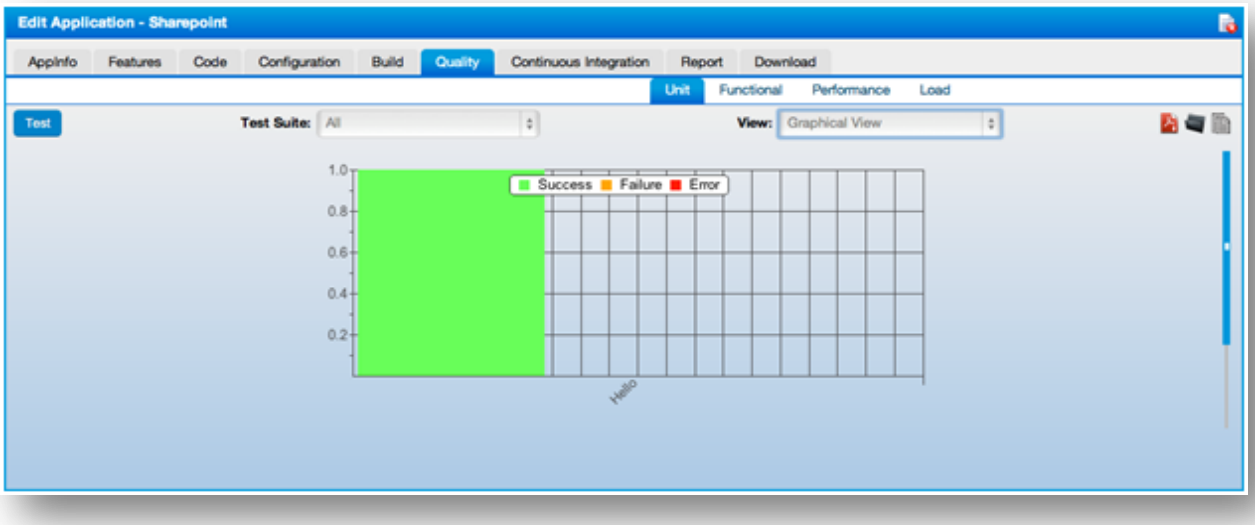


Figure 8-23: SharePoint unit test report for all test cases in tabular view

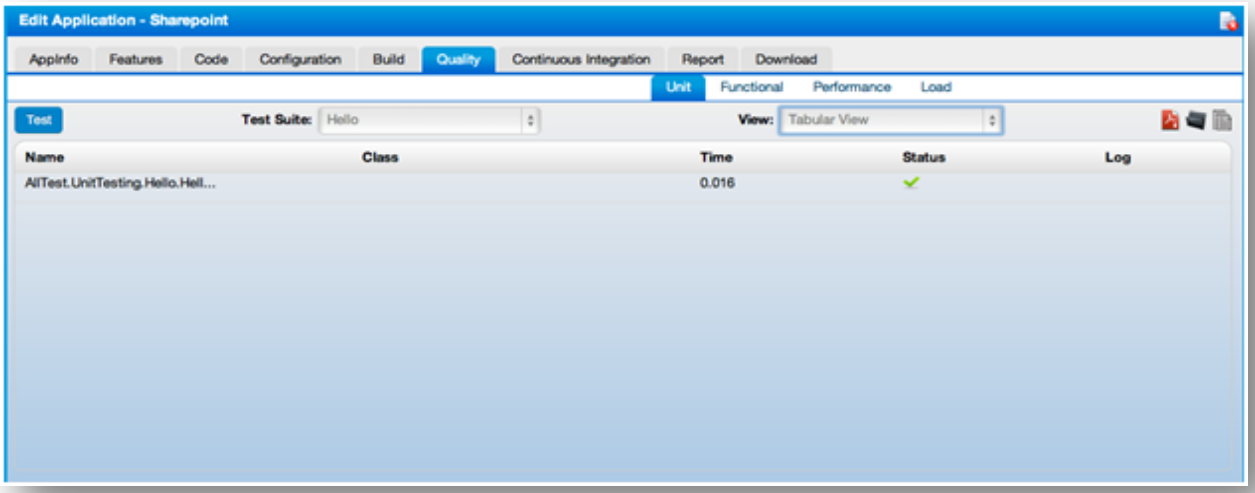


Figure 8-24: SharePoint unit test report for single test case in tabular view

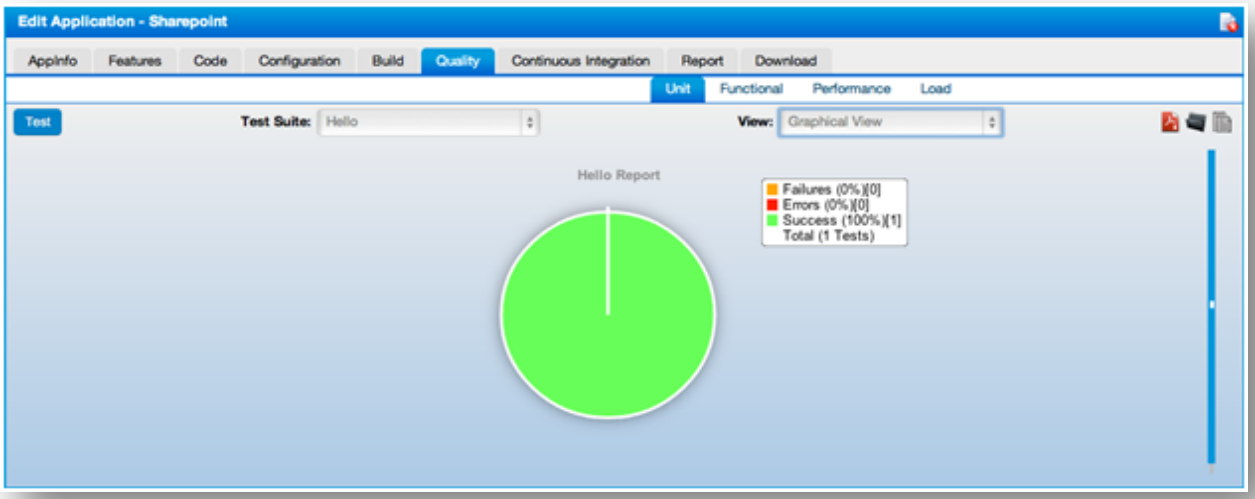


Figure 8-25: SharePoint unit test report for single test case in graphical view

8.3.2 Functional Test Case

8.3.2.1 Structure of Classes and Test Case

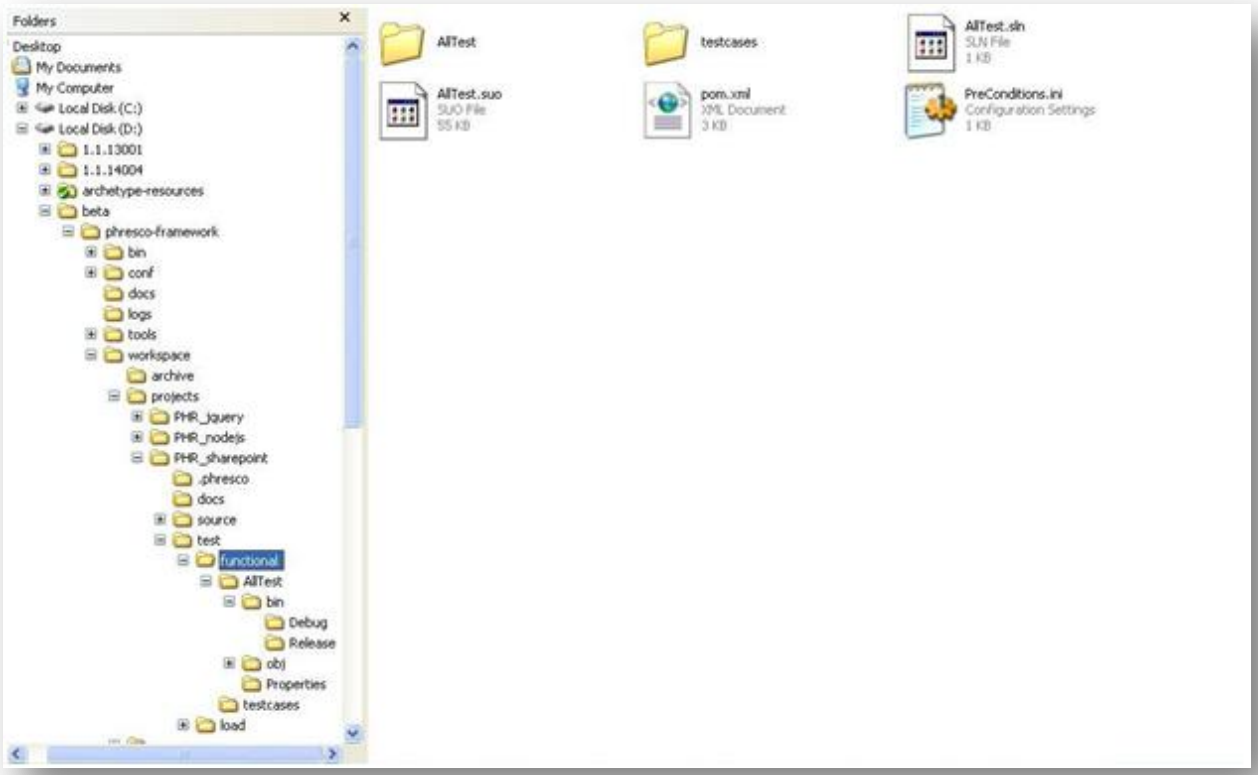


Figure 8-26: SharePoint functional tests structure

a. **AllTest:** It is a root folder/file that carries all the classes to initiate testing.

b. **Class:** All the classes is incorporated in the AllTest

c. **Test Case:** All the test cases should be added within the Class

8.3.2.2 Existing Test Cases and Classes Out Of the Box

Alltest for Share Point

AllTest contains a bunch of test cases, each of which performs a unique scenario on the application. Test developers can start writing new suite classes and test cases by following the syntax and structure of Phresco framework's out of the box class structure.

AllTest class calls all the suite classes and the test cases within it. Once suite classes and test cases are written developers can execute those from Phresco Framework and can see the report.

Following is the AllTest file which shows up how to add / include the class inside it.

```
using System;
using System.Configuration;
using System.Data;
using System.IO;
using System.Text;
using System.Threading;
using NUnit.Framework;
using OpenQA.Selenium;
using OpenQA.Selenium.IE;
using Selenium;
namespace AllTest{

   [TestFixture]
   public class HelloWorld {

      public IWebDriver driver;
      private StringBuilder verificationErrors;
      String baseUrl, sceenshotpath;
      [SetUp]
      public void SetupTest() {

         DataSet helloworld = new DataSet();
         helloworld.ReadXml(ConfigurationSettings.AppSettings["XmlPath"].ToString());
         baseUrl = helloworld.Tables[1].Rows[0].ItemArray[1].ToString() + ":" + "//" +
        helloworld.Tables[1].Rows[0].ItemArray[2].ToString() + ":" +
        helloworld.Tables[1].Rows[0].ItemArray[3].ToString() + "/" +
        helloworld.Tables[1].Rows[0].ItemArray[0].ToString();
         try {

            driver = new InternetExplorerDriver();
            driver.Navigate().GoToUrl(baseUrl);
            verificationErrors = new StringBuilder();
         }
         catch (Exception e) {

            Assert.AreEqual(helloworld.Tables[0].Rows[0].ItemArray[2].ToString(),
e.ToString());
         }
      }

      [TearDown]
      public void TeardownTest() {
```

```
            try {

                driver.Quit();
            }
            catch (Exception){

                // Ignore errors if unable to close the browser
            }
            Assert.AreEqual("", verificationErrors.ToString());
        }
        public void TakeScreenshot(IWebDriver driver, string path) {

            ITakesScreenshot screenshotDriver = driver as ITakesScreenshot;
            Screenshot screenshot = screenshotDriver.GetScreenshot();
            screenshot.SaveAsFile(path, System.Drawing.Imaging.ImageFormat.Png);
            screenshot.ToString();
        }

        [Test]
        public void helloWorldTest() {

            DataSet helloworld = new DataSet();
            helloworld.ReadXml(ConfigurationSettings.AppSettings["XmlPath"].ToString());
            sceenshotpath=helloworld.Tables[0].Rows[0].ItemArray[10].ToString();
            baseUrl = helloworld.Tables[1].Rows[0].ItemArray[1].ToString() + ":" + "//" +
helloworld.Tables[1].Rows[0].ItemArray[2].ToString() + ":" +
helloworld.Tables[1].Rows[0].ItemArray[3].ToString() + "/" +
helloworld.Tables[1].Rows[0].ItemArray[0].ToString();
            Directory.CreateDirectory(helloworld.Tables[0].Rows[0].ItemArray[9].ToString());
           try {

              //Opens Phresco Home Page.
              driver.Navigate().GoToUrl(baseUrl);
              //Click on site actions and go to edit page.

driver.FindElement(By.XPath(helloworld.Tables[0].Rows[0].ItemArray[4].ToString())).Cli
ck();

driver.FindElement(By.XPath(helloworld.Tables[0].Rows[0].ItemArray[5].ToString())).Clic
k();

Thread.Sleep(Convert.ToInt32(helloworld.Tables[0].Rows[0].ItemArray[3].ToString()));
              //Close the hello world web part.

driver.FindElement(By.XPath(helloworld.Tables[0].Rows[0].ItemArray[6].ToString())).Cli
ck();

Thread.Sleep(Convert.ToInt32(helloworld.Tables[0].Rows[0].ItemArray[3].ToString()));
              driver.Navigate().Refresh();
              driver.Navigate().GoToUrl(baseUrl);

              }
```

```
            catch (Exception e) {

                TakeScreenshot(driver, helloworld.Tables[0].Rows[0].ItemArray[10].ToString());
                throw e;

            }
        }
    }
}
```

Class Example for Chartlist

Class is a collection of test cases that are intended to be used to test a project code to show that it has some specified set of behaviors. A class contains detailed instructions or goals for each collection of test cases. New test cases can also be added by using the same syntax.

```
using System;
using System.Configuration;
using System.Data;
using System.IO;
using System.Text;
using System.Threading;
using NUnit.Framework;
using OpenQA.Selenium;
using OpenQA.Selenium.IE;
using Selenium;
using OpenQA.Selenium.Support.UI;

namespace ChartList_Create_Delete{

    [TestFixture]
    public class Chart_Create_Delete {

        public IWebDriver driver;
        private StringBuilder verificationErrors;
        String baseUrl;
        [SetUp]
        public void SetupTest() {

            DataSet Phresco = new DataSet();
            Phresco.ReadXml(ConfigurationSettings.AppSettings["XmlPath"].ToString());
                baseUrl = Phresco.Tables[1].Rows[0].ItemArray[0].ToString() + ":" + "//" +
                Phresco.Tables[1].Rows[0].ItemArray[1].ToString() + ":" +
                Phresco.Tables[1].Rows[0].ItemArray[2].ToString() + "/" +
                Phresco.Tables[1].Rows[0].ItemArray[3].ToString();
            try{

                verificationErrors = new StringBuilder();
```

```
            driver = new InternetExplorerDriver();
            driver.Navigate().GoToUrl(baseUrl);
        }
        catch (Exception e) {

            Assert.AreEqual(Phresco.Tables[0].Rows[0].ItemArray[3].ToString(), e.ToString());
        }
    }

    [TearDown]
    public void TeardownTest(){

        try{

            driver.Quit();
        }
        catch (Exception){

            // Ignore errors if unable to close the browser
        }
        Assert.AreEqual("", verificationErrors.ToString());
    }
    public void TakeScreenshot(IWebDriver driver, string path){

        ITakesScreenshot screenshotDriver = driver as ITakesScreenshot;
        Screenshot screenshot = screenshotDriver.GetScreenshot();
        screenshot.SaveAsFile(path, System.Drawing.Imaging.ImageFormat.Png);
        screenshot.ToString();
    }
    [Test]
    public void ChartListPrepTest(){

        DataSet Phresco = new DataSet();
        Phresco.ReadXml(ConfigurationSettings.AppSettings["XmlPath"].ToString());
        try {

            //Opens Phresco Home Page.
            driver.Navigate().GoToUrl(baseUrl);
            Thread.Sleep(Convert.ToInt32(Phresco.Tables[0].Rows[0].ItemArray[2].ToString()));

driver.FindElement(By.XPath(Phresco.Tables[0].Rows[0].ItemArray[4].ToString())).Click();
            //Opens VM_allocation List Web Page.

driver.FindElement(By.XPath(Phresco.Tables[0].Rows[0].ItemArray[5].ToString())).Click();
            Thread.Sleep(Convert.ToInt32(Phresco.Tables[0].Rows[0].ItemArray[2].ToString()));
            //Opens New VM_Allocation List Definition Page.

driver.FindElement(By.XPath(Phresco.Tables[0].Rows[0].ItemArray[6].ToString())).Click();
            Thread.Sleep(Convert.ToInt32(Phresco.Tables[0].Rows[0].ItemArray[2].ToString()));

driver.FindElement(By.XPath(Phresco.Tables[0].Rows[0].ItemArray[7].ToString())).SendKeys(
Phresco.Tables[0].Rows[0].ItemArray[8].ToString());
```

```
driver.FindElement(By.XPath(Phresco.Tables[0].Rows[0].ItemArray[9].ToString())).SendKeys(
Phresco.Tables[0].Rows[0].ItemArray[10].ToString());
            IWebElement selectWindows =
driver.FindElement(By.XPath(Phresco.Tables[0].Rows[0].ItemArray[11].ToString()));
            SelectElement clickThis = new SelectElement(selectWindows);
            clickThis.SelectByText(Phresco.Tables[0].Rows[0].ItemArray[12].ToString());

driver.FindElement(By.XPath(Phresco.Tables[0].Rows[0].ItemArray[13].ToString())).SendKeys
(Phresco.Tables[0].Rows[0].ItemArray[14].ToString());
            IWebElement hardwareType =
driver.FindElement(By.XPath(Phresco.Tables[0].Rows[0].ItemArray[15].ToString()));
            SelectElement clickHT = new SelectElement(hardwareType);
            clickHT.SelectByText(Phresco.Tables[0].Rows[0].ItemArray[16].ToString());

driver.FindElement(By.XPath(Phresco.Tables[0].Rows[0].ItemArray[17].ToString())).SendKeys
(Phresco.Tables[0].Rows[0].ItemArray[18].ToString());

driver.FindElement(By.XPath(Phresco.Tables[0].Rows[0].ItemArray[19].ToString())).SendKeys
(Phresco.Tables[0].Rows[0].ItemArray[20].ToString());
            IWebElement statusType =
driver.FindElement(By.XPath(Phresco.Tables[0].Rows[0].ItemArray[21].ToString()));
            SelectElement clickActive = new SelectElement(statusType);
            clickActive.SelectByText(Phresco.Tables[0].Rows[0].ItemArray[22].ToString());

driver.FindElement(By.XPath(Phresco.Tables[0].Rows[0].ItemArray[23].ToString())).Click();
            Thread.Sleep(Convert.ToInt32(Phresco.Tables[0].Rows[0].ItemArray[2].ToString()));

        }
        catch (Exception e){

            TakeScreenshot(driver, Phresco.Tables[0].Rows[0].ItemArray[24].ToString());
            throw e;
        }
    }
```

Test Case for Chartlistpreptest

Test cases examine all the aspects including inputs and outputs. It gives the detailed steps that should be followed during the testing process. Testing process follows only the steps that have been written for each test case.

It also helps in identifying an element and capturing screen shots when the test fails.

```
    [Test]
     public void ChartListPrepTest(){

        DataSet Phresco = new DataSet();
```

```
        Phresco.ReadXml(ConfigurationSettings.AppSettings["XmlPath"].ToString());
        try {

            //Opens Phresco Home Page.
            driver.Navigate().GoToUrl(baseUrl);
            Thread.Sleep(Convert.ToInt32(Phresco.Tables[0].Rows[0].ItemArray[2].ToString()));

driver.FindElement(By.XPath(Phresco.Tables[0].Rows[0].ItemArray[4].ToString())).Click();
            //Opens VM_allocation List Web Page.

driver.FindElement(By.XPath(Phresco.Tables[0].Rows[0].ItemArray[5].ToString())).Click();
            Thread.Sleep(Convert.ToInt32(Phresco.Tables[0].Rows[0].ItemArray[2].ToString()));
            //Opens New VM_Allocation List Definition Page.

driver.FindElement(By.XPath(Phresco.Tables[0].Rows[0].ItemArray[6].ToString())).Click();
            Thread.Sleep(Convert.ToInt32(Phresco.Tables[0].Rows[0].ItemArray[2].ToString()));

driver.FindElement(By.XPath(Phresco.Tables[0].Rows[0].ItemArray[7].ToString())).SendKeys(Ph
resco.Tables[0].Rows[0].ItemArray[8].ToString());

driver.FindElement(By.XPath(Phresco.Tables[0].Rows[0].ItemArray[9].ToString())).SendKeys(P
hresco.Tables[0].Rows[0].ItemArray[10].ToString());
            IWebElement selectWindows =
driver.FindElement(By.XPath(Phresco.Tables[0].Rows[0].ItemArray[11].ToString()));
            SelectElement clickThis = new SelectElement(selectWindows);
            clickThis.SelectByText(Phresco.Tables[0].Rows[0].ItemArray[12].ToString());

driver.FindElement(By.XPath(Phresco.Tables[0].Rows[0].ItemArray[13].ToString())).SendKeys(P
hresco.Tables[0].Rows[0].ItemArray[14].ToString());
            IWebElement hardwareType =
driver.FindElement(By.XPath(Phresco.Tables[0].Rows[0].ItemArray[15].ToString()));
            SelectElement clickHT = new SelectElement(hardwareType);
            clickHT.SelectByText(Phresco.Tables[0].Rows[0].ItemArray[16].ToString());

driver.FindElement(By.XPath(Phresco.Tables[0].Rows[0].ItemArray[17].ToString())).SendKeys(P
hresco.Tables[0].Rows[0].ItemArray[18].ToString());

driver.FindElement(By.XPath(Phresco.Tables[0].Rows[0].ItemArray[19].ToString())).SendKeys(P
hresco.Tables[0].Rows[0].ItemArray[20].ToString());
            IWebElement statusType =
driver.FindElement(By.XPath(Phresco.Tables[0].Rows[0].ItemArray[21].ToString()));
            SelectElement clickActive = new SelectElement(statusType);
            clickActive.SelectByText(Phresco.Tables[0].Rows[0].ItemArray[22].ToString());

driver.FindElement(By.XPath(Phresco.Tables[0].Rows[0].ItemArray[23].ToString())).Click();
            Thread.Sleep(Convert.ToInt32(Phresco.Tables[0].Rows[0].ItemArray[2].ToString()));

        }
        catch (Exception e) {

            TakeScreenshot(driver, Phresco.Tables[0].Rows[0].ItemArray[24].ToString());
            throw e;    }
```

8.3.2.3 To Add A New Class

To add a class you can just right click and add new class. Name of the class can be entered and saved. Any number of test cases can be added in this new class.

8.3.2.4 To Add New Test Case in Class

You can add a new test case to the class which is created.

8.3.2.5 Report generated after execution

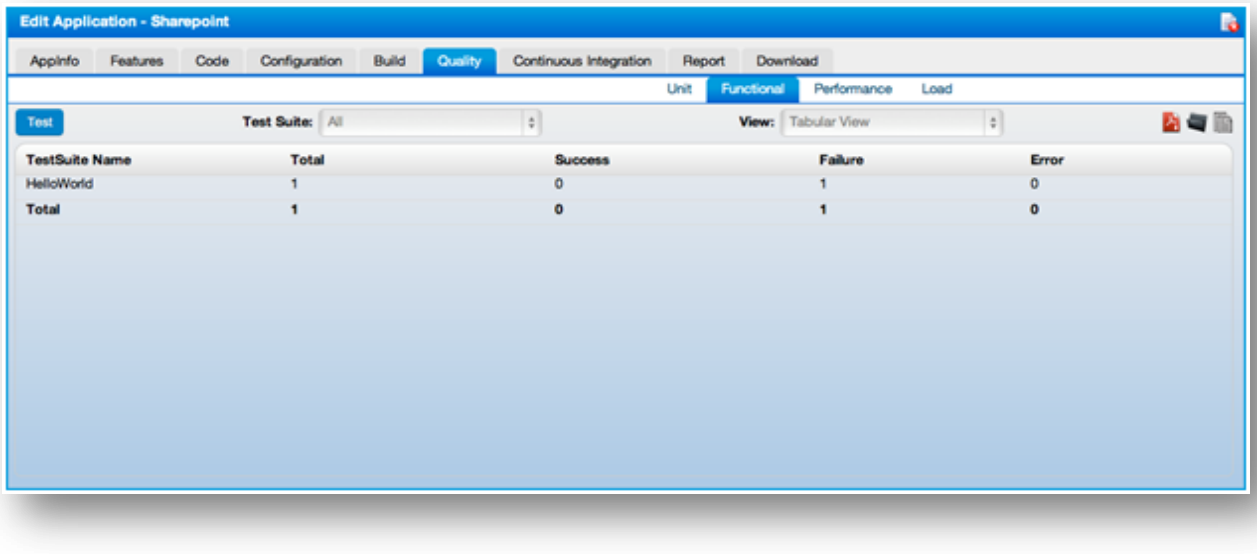


Figure 8-27: SharePoint functional test report for all test cases in tabular view

:

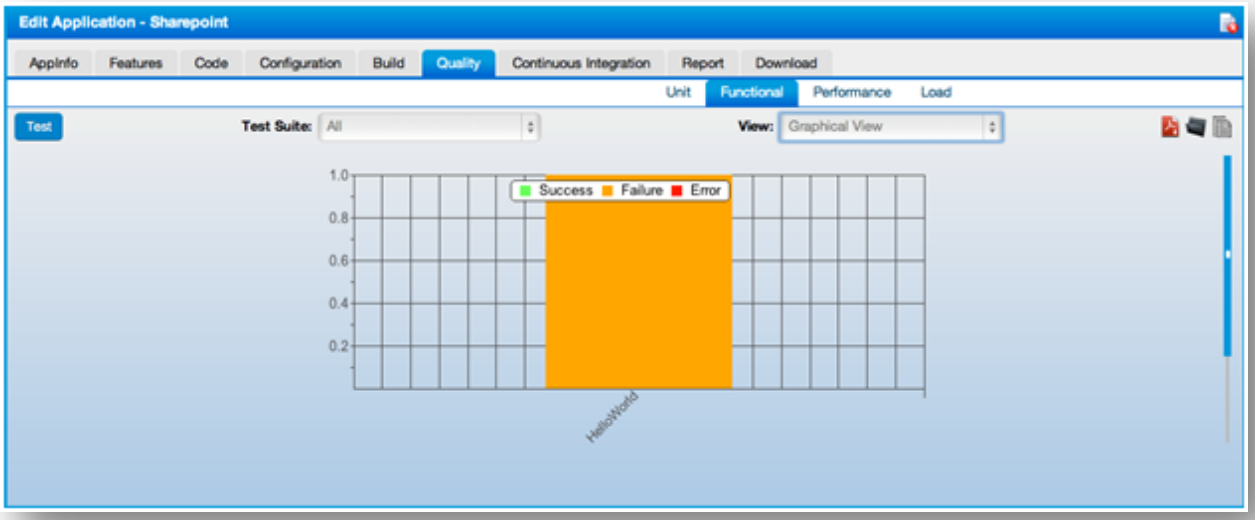


Figure 8-28: SharePoint functional test report for all test cases in graphical view

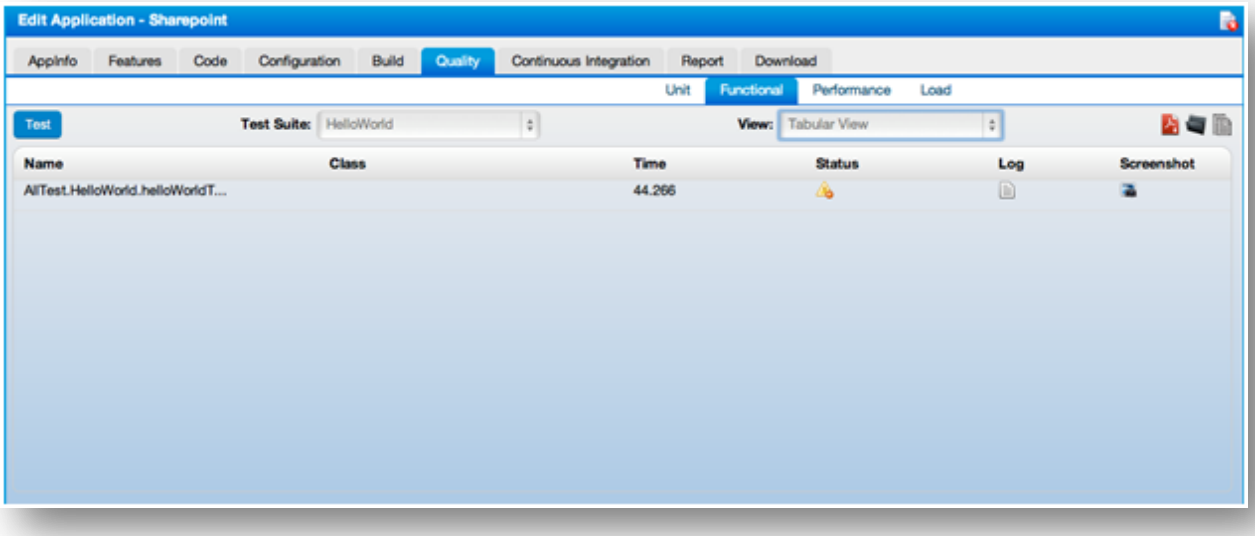


Figure 8-29: SharePoint functional test report for single test case in tabular view

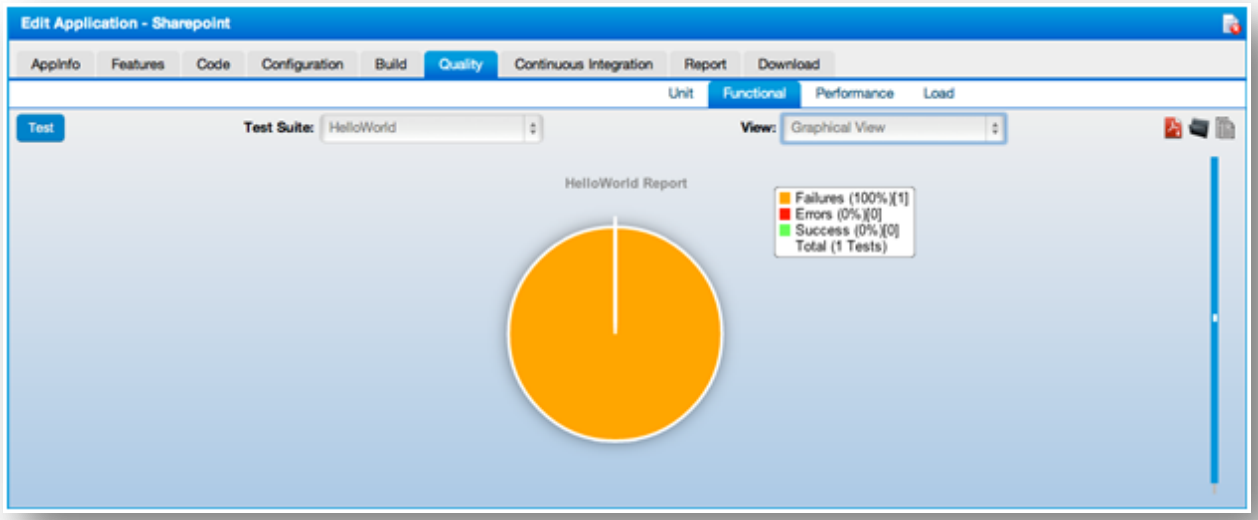


Figure 8-30: ***SharePoint functional test for single test case in graphical view***

8.4 Java Standalone Application

8.4.1 Functional Test Case

8.4.1.1 Structure of Test Suites and Test Case

a. **AllTest**: Alltest is a java suite class that can either initiate a suite class or a group of test cases.

b. **Test suite**: Test suite is a collection of test cases.

c. **Test case**: All the Test cases should be added within the test suite.

8.4.1.2 Existing Test Cases and Suites Out Of the Box in Phresco

Alltest for Java Standalone Application

AllTest is the root category which holds the test suite. AllTest contains a bunch of test suites, each of which performs a unique scenario on the application. Test developers can start writing new suite classes by following the syntax and structure of Phresco framework's out of the box class structure. Once test suite classes and test cases are written developers can execute those from Phresco Framework and can see the report. Following is the AllTest file which shows up how to add / include the class inside it.

```
package com.photon.Phresco.testcases;

import org.junit.runner.RunWith;
import org.junit.runners.Suite;
import org.junit.runners.Suite.SuiteClasses;

@RunWith(Suite.class)
@SuiteClasses({SampleHelloWorld.class})
public class AllTest {

}
```

Test Suites Example

Test suite is a collection of test cases that are intended to be used to test a project code to show that it has some specified set of behaviors. A test suite contains detailed instructions or goals for each collection of test cases. New test cases can also be added.

Test Case Example

A test case is a set of conditions under which a tester will determine whether an application is meeting the requirement. This helps the user to verify a particular functionality during the testing process. Functional test case for java standalone application is written using the FEST tool.

```
package com.photon.Phresco.testcases;

import static org.fest.assertions.Assertions.assertThat;
import static org.fest.swing.edt.GuiActionRunner.execute;

import org.fest.swing.annotation.RunsInEDT;
import org.fest.swing.edt.GuiQuery;
import org.fest.swing.fixture.FrameFixture;
import org.fest.swing.junit.testcase.FestSwingJUnitTestCase;
import org.junit.Test;

import com.photon.Phresco.HelloWorld;

public class HelloWorldTest extends FestSwingJUnitTestCase {

  private FrameFixture Phresco;

  protected void onSetUp() {
        Phresco = new FrameFixture(robot(), createNewEditor());
        Phresco.show();
        Phresco.maximize();
        System.out.println("**************Executed onsetup**************");
  }

  @RunsInEDT
  private static HelloWorld createNewEditor() {
        return execute(new GuiQuery<HelloWorld>() {
              protected HelloWorld executeInEDT() throws Throwable {
                    return new HelloWorld();
              }
        });
  }
```

```
@Test
public void testHelloWorld() throws InterruptedException {
        Thread.sleep(5000);
        assertThat(Phresco.label().text()).contains("Hello World");

  }
}
```

8.4.1.3 To Add New Test Case In Test Suite

You can add a new test case to the Test suite using the following method.

```
@Test
public void testHelloWorld() throws InterruptedException {
        Thread.sleep(5000);
        assertThat(Phresco.label().text()).contains("Hello World");
```

8.4.1.4 Report generated after execution

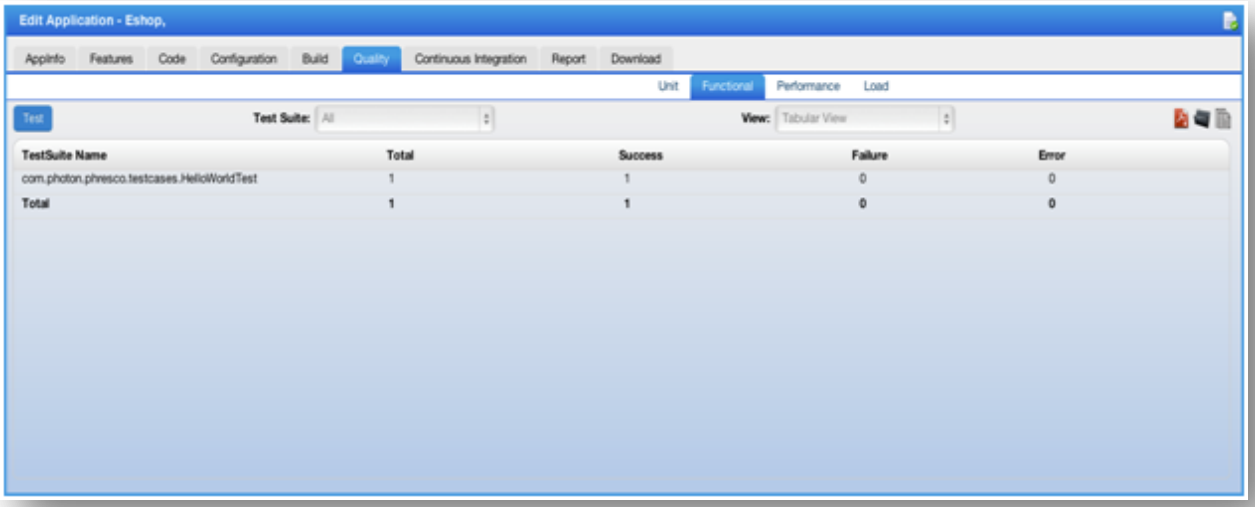


Figure 8-31: ***Java Standalone functional test report for all test cases in tabular view***

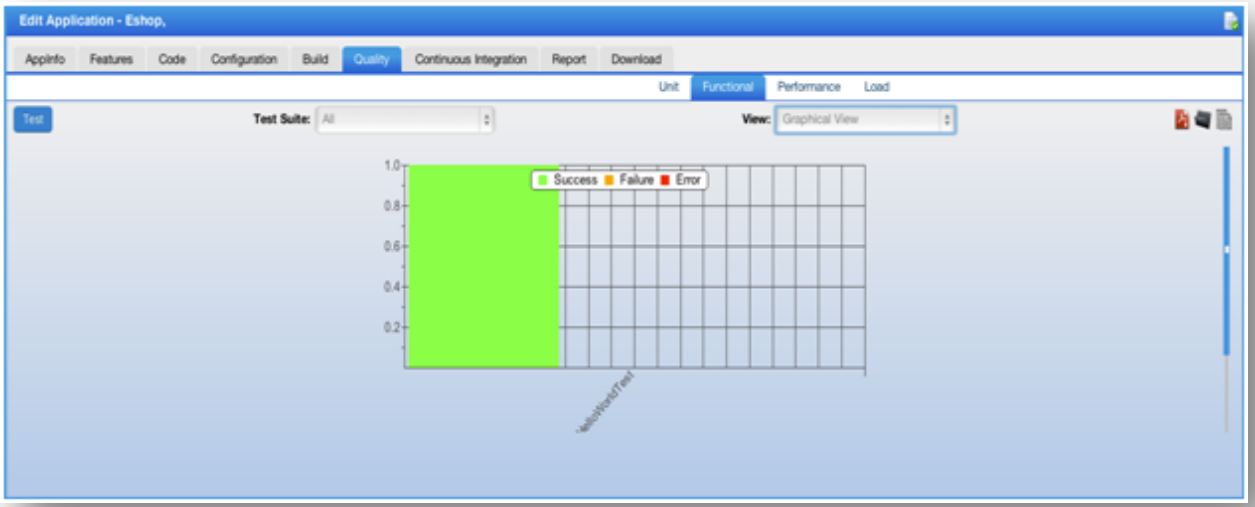


Figure 8-32: ***Java Standalone functional test report for all test cases in graphical view***

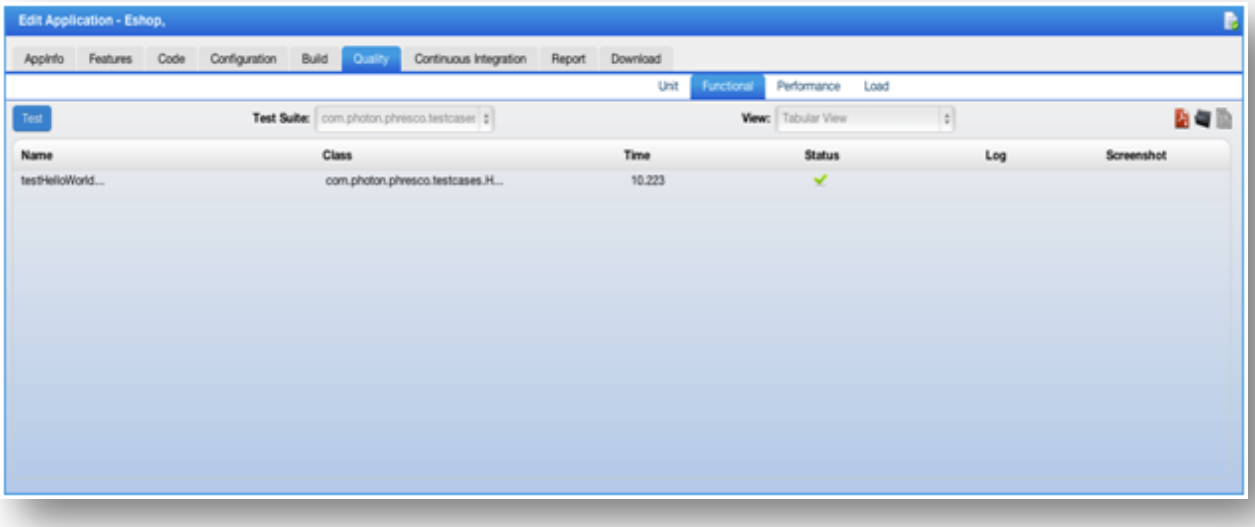


Figure 8-33: ***Java Standalone functional test report for single test case in graphical view***

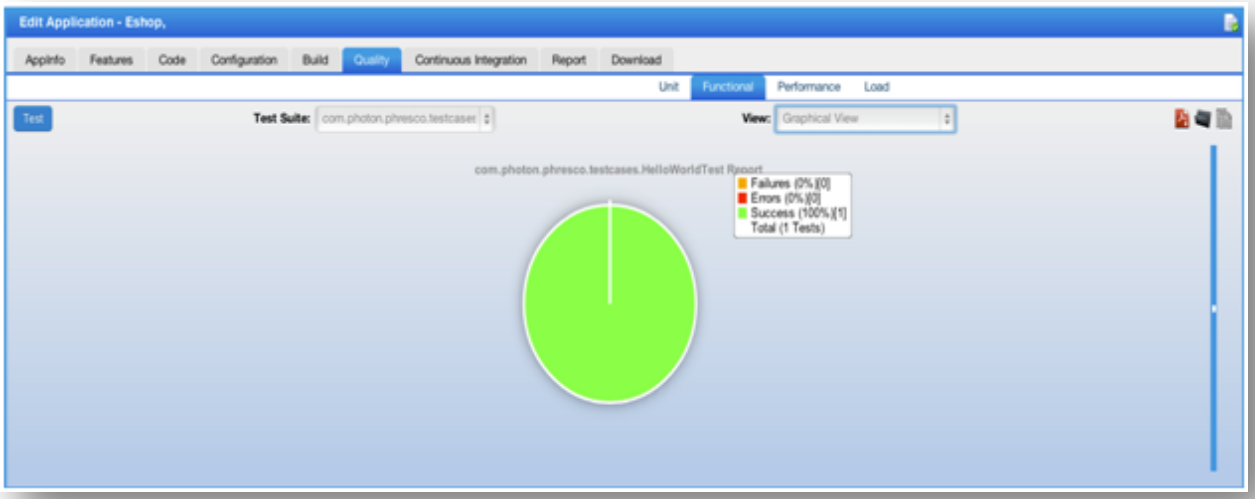


Figure 8-34: ***Java Standalone functional test report for single test case in graphical view***

8.5 Android application

8.5.1 Unit Test Case

8.5.1.1 Structure of Alltest and Test Cases

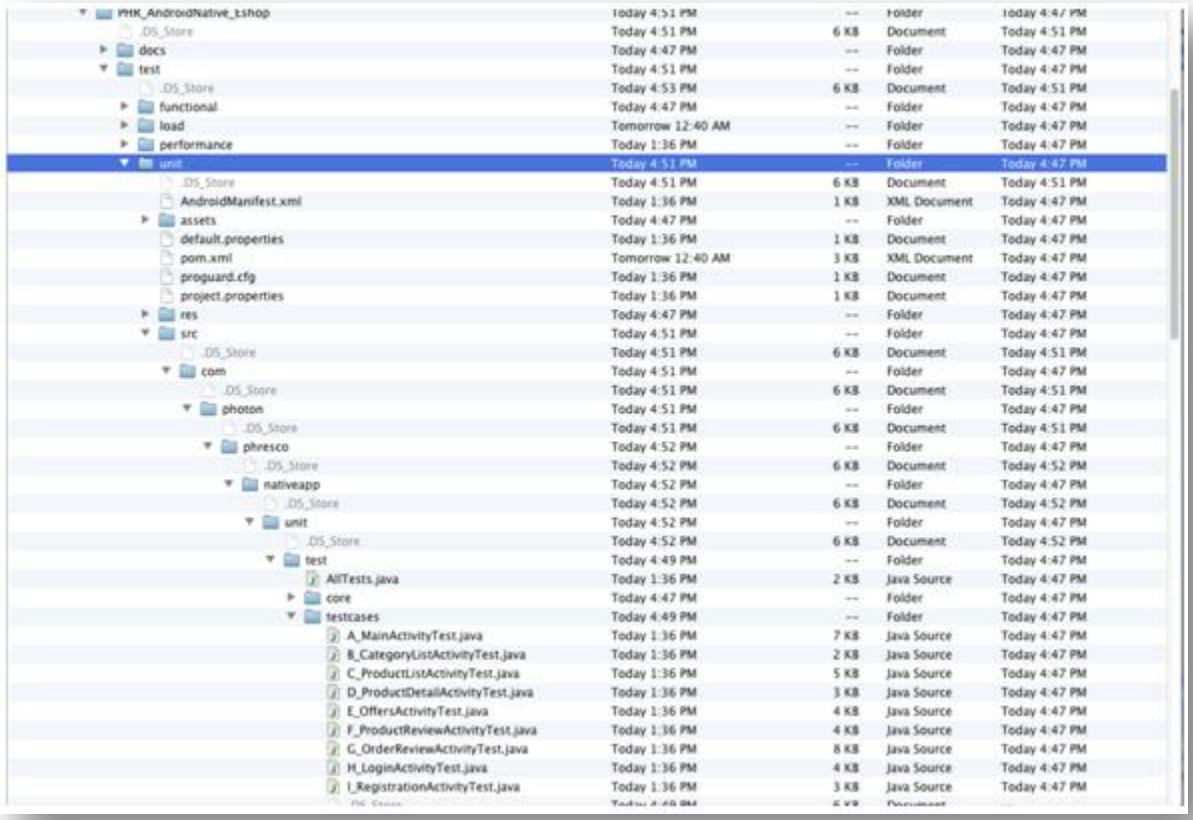


Figure 8-35: Android unit tests structure

a. **AllTest (test suite):** AllTest is the class that carries all the test classes to initiate the testing process. Each test class can be called within the AllTest.

b. **Test class:** Test classes hold different test methods

c. **Test methods/test cases:** Test cases are called in the test classes which test the source code.

8.5.1.2 Existing Test Cases Out Of the Box in Phresco

AllTest for Android Technology

AllTest contains a bunch of test cases, each of which performs a unique scenario on the application. Test developers can start writing new suite classes and test cases by following the syntax and structure of Phresco framework's out of the box class structure. AllTest class calls all the suite classes and the test cases within it. Once suite classes and test cases are written developers can execute those from Phresco Framework and can see the report. Following is the AllTest file which shows up how to add / include the class inside it.

```
package com.photon.Phresco.nativeapp.unit.test;

import junit.framework.TestCase;
import junit.framework.TestSuite;

import com.photon.Phresco.nativeapp.unit.test.testcases.A_MainActivityTest;
import com.photon.Phresco.nativeapp.unit.test.testcases.B_CategoryListActivityTest;
import com.photon.Phresco.nativeapp.unit.test.testcases.C_ProductListActivityTest;
import com.photon.Phresco.nativeapp.unit.test.testcases.D_ProductDetailActivityTest;
import com.photon.Phresco.nativeapp.unit.test.testcases.E_OffersActivityTest;
import com.photon.Phresco.nativeapp.unit.test.testcases.F_ProductReviewActivityTest;
import com.photon.Phresco.nativeapp.unit.test.testcases.G_OrderReviewActivityTest;
import com.photon.Phresco.nativeapp.unit.test.testcases.H_LoginActivityTest;

public class AllTests extends TestCase {
  public static TestSuite suite() {

            TestSuite suite = new TestSuite(AllTests.class.getName());

            suite.addTestSuite(A_MainActivityTest.class);
            suite.addTestSuite(B_CategoryListActivityTest.class);
            suite.addTestSuite(C_ProductListActivityTest.class);
            suite.addTestSuite(D_ProductDetailActivityTest.class);
            suite.addTestSuite(E_OffersActivityTest.class);
            suite.addTestSuite(F_ProductReviewActivityTest.class);
            suite.addTestSuite(G_OrderReviewActivityTest.class);
            suite.addTestSuite(H_LoginActivityTest.class);

            return suite;

  }
   public ClassLoader getLoader() {
        return AllTests.class.getClassLoader();
     }
}
```

Test Method Example for Loginactivitytest

This test method is written for testing the login activity. There can be n number of test methods under a test class. Test class calls the test methods. An example of a test class is given below

```
package com.photon.Phresco.nativeapp.unit.test.testcases;

import java.io.IOException;

import junit.framework.TestCase;

import org.json.JSONException;
import org.json.JSONObject;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;

import com.photon.Phresco.nativeapp.eshop.json.JSONHelper;
import com.photon.Phresco.nativeapp.eshop.logger.PhrescoLogger;
import com.photon.Phresco.nativeapp.unit.test.core.Constants;

public class H_LoginActivityTest extends TestCase{
 private static final String TAG = "H_LoginActivityTest  ********** ";

 /**
  * @throws java.lang.Exception
  */
 @BeforeClass
 public static void setUpBeforeClass() {
 }

 /**
  * @throws java.lang.Exception
  */
 @AfterClass
 public static void tearDownAfterClass() {
 }

 /**
  * @throws java.lang.Exception
  */
 @Before
 public void setUp() {
 }

 /**
  * @throws java.lang.Exception
```

```
 */
@After
public void tearDown() {
}

/**
 *  send valid login email id and password to web server
 *
 */
@Test
public final void testLogin() {

        JSONObject jObjMain = new JSONObject();
        JSONObject jObj = new JSONObject();

        try {
                PhrescoLogger.info(TAG + " testLogin ---------- START ");

                        jObj.put("loginEmail", "tester@Phresco.com");
                        jObj.put("password", "123");
                        jObjMain.put("login", jObj);

                        JSONObject responseJSON = null;

 responseJSON=JSONHelper.postJSONObjectToURL(Constants.getWebContextURL() +
Constants.getRestAPI() + Constants.LOGIN_POST_URL, jObjMain.toString());
                        assertNotNull("Login response is not null",responseJSON.length()
> 0);

                        PhrescoLogger.info(TAG + " testLogin ---------- END ");

        } catch (IOException ex) {
                PhrescoLogger.info(TAG + " - testLogin  - IOException : " + ex.toString());
                PhrescoLogger.warning(ex);
        } catch (JSONException ex) {
                PhrescoLogger.info(TAG + " - testLogin  - JSONException : " +
ex.toString());
                PhrescoLogger.warning(ex);
        }
}
```

Test Case Example for Test Login

A test case is a set of conditions under which a tester will determine whether an application is meeting the requirement. This helps the user to verify a particular functionality during the testing process.

Example of test case in login activity is given below:

```
@Test
  public final void testLogin() {

		JSONObject jObjMain = new JSONObject();
		JSONObject jObj = new JSONObject();

		try {
			PhrescoLogger.info(TAG + " testLogin ---------- START ");

				jObj.put("loginEmail", "tester@Phresco.com");
				jObj.put("password", "123");
				jObjMain.put("login", jObj);

				JSONObject responseJSON = null;

  responseJSON=JSONHelper.postJSONObjectToURL(Constants.getWebContextURL() +
Constants.getRestAPI() + Constants.LOGIN_POST_URL, jObjMain.toString());
				assertNotNull("Login response is not null",responseJSON.length()
> 0);

				PhrescoLogger.info(TAG + " testLogin ---------- END ");

		} catch (IOException ex) {
			PhrescoLogger.info(TAG + " - testLogin  - IOException : " + ex.toString());
			PhrescoLogger.warning(ex);
		} catch (JSONException ex) {
			PhrescoLogger.info(TAG + " - testLogin  - JSONException : " +
ex.toString());
			PhrescoLogger.warning(ex);
		}
}
```

8.5.1.3 To Add New Test Method in Alltest Class

You can add a new test method to the AllTest class as follows.
Here is an example for creating a new test suite in the name 'RegistrationActivityTest'

```
import com.photon.Phresco.nativeapp.unit.test.testcases.A_MainActivityTest;
import com.photon.Phresco.nativeapp.unit.test.testcases.B_CategoryListActivityTest;
import t com.photon.Phresco.nativeapp.unit.test.testcases.C_ProductListActivityTest;
import com.photon.Phresco.nativeapp.unit.test.testcases.D_ProductDetailActivityTest;
import com.photon.Phresco.nativeapp.unit.test.testcases.E_OffersActivityTest;
import com.photon.Phresco.nativeapp.unit.test.testcases.F_ProductReviewActivityTest;
import com.photon.Phresco.nativeapp.unit.test.testcases.G_OrderReviewActivityTest;
import com.photon.Phresco.nativeapp.unit.test.testcases.H_LoginActivityTest;
import com.photon.Phresco.nativeapp.unit.test.testcases.I_RegistrationActivityTest;
```

8.5.1.4 Report generated after execution

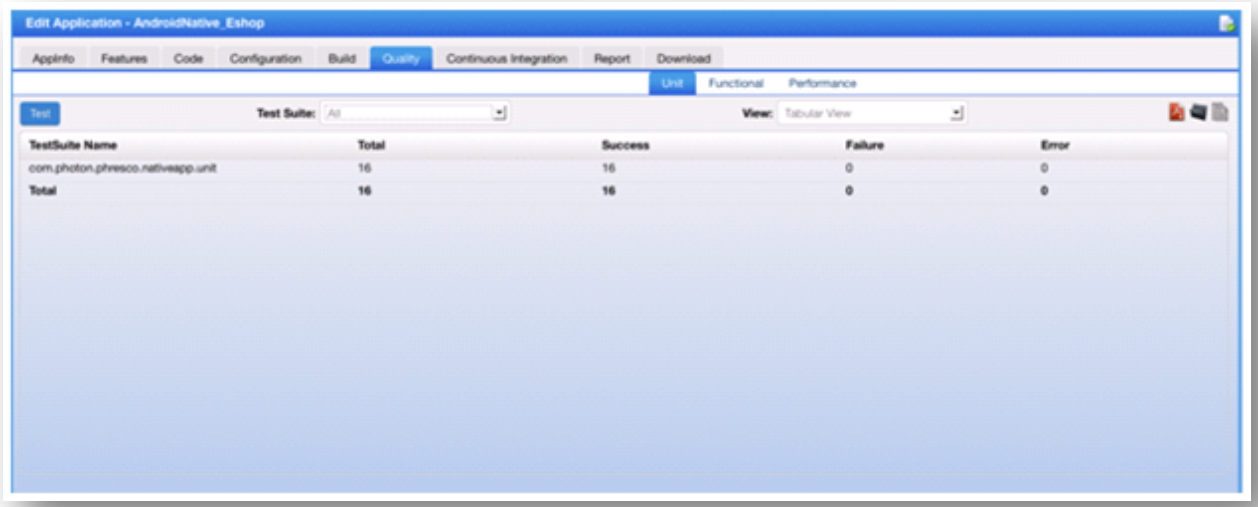


Figure 8-36: Android unit test report for all test cases in tabular view

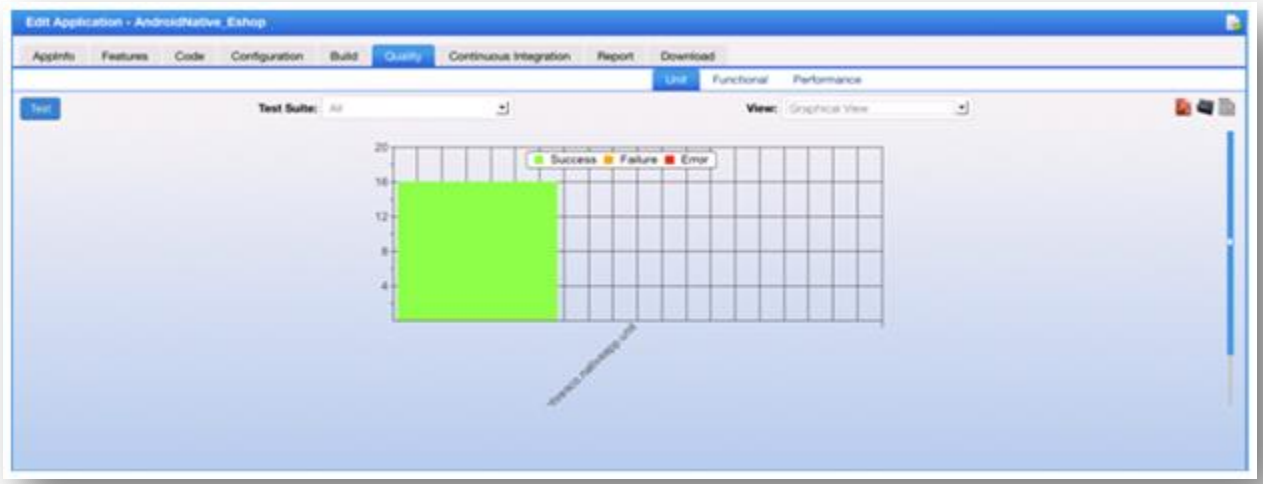


Figure 8-37: ***Android unit test report for all test cases in graphical view***

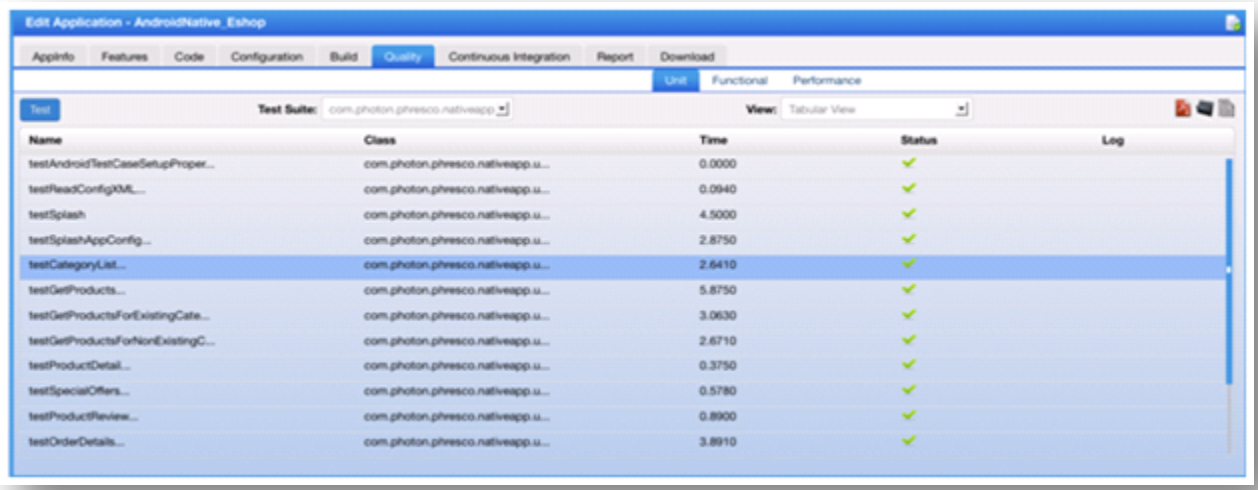


Figure 8-38: ***Android unit test report for single test case in graphical view***

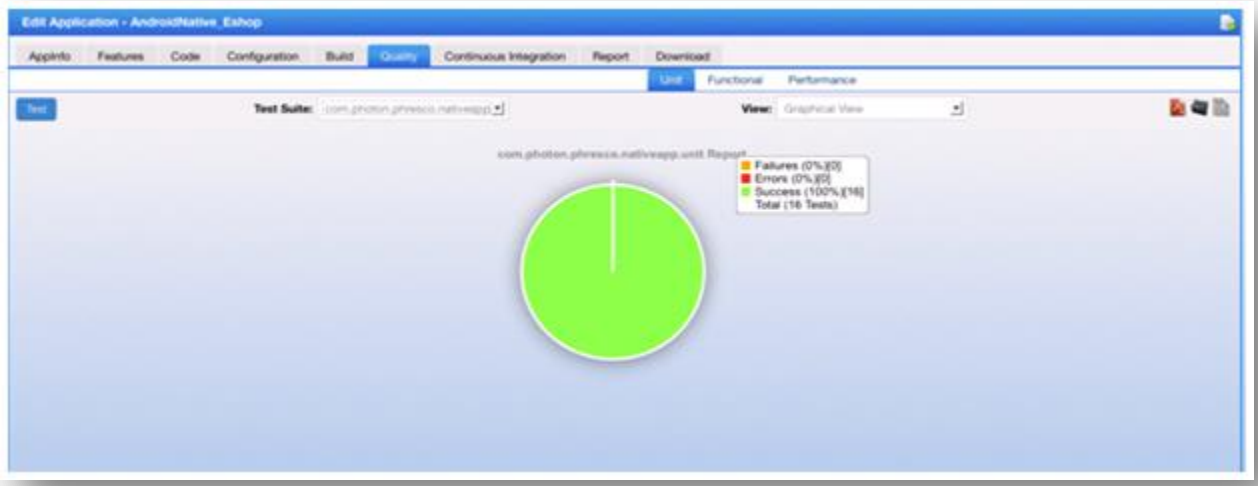


Figure 8-39: ***Android unit test report for single test case in graphical view***

8.5.2 Functional Test Case

8.5.2.1 Structure of Functional Test of Android (Functional)

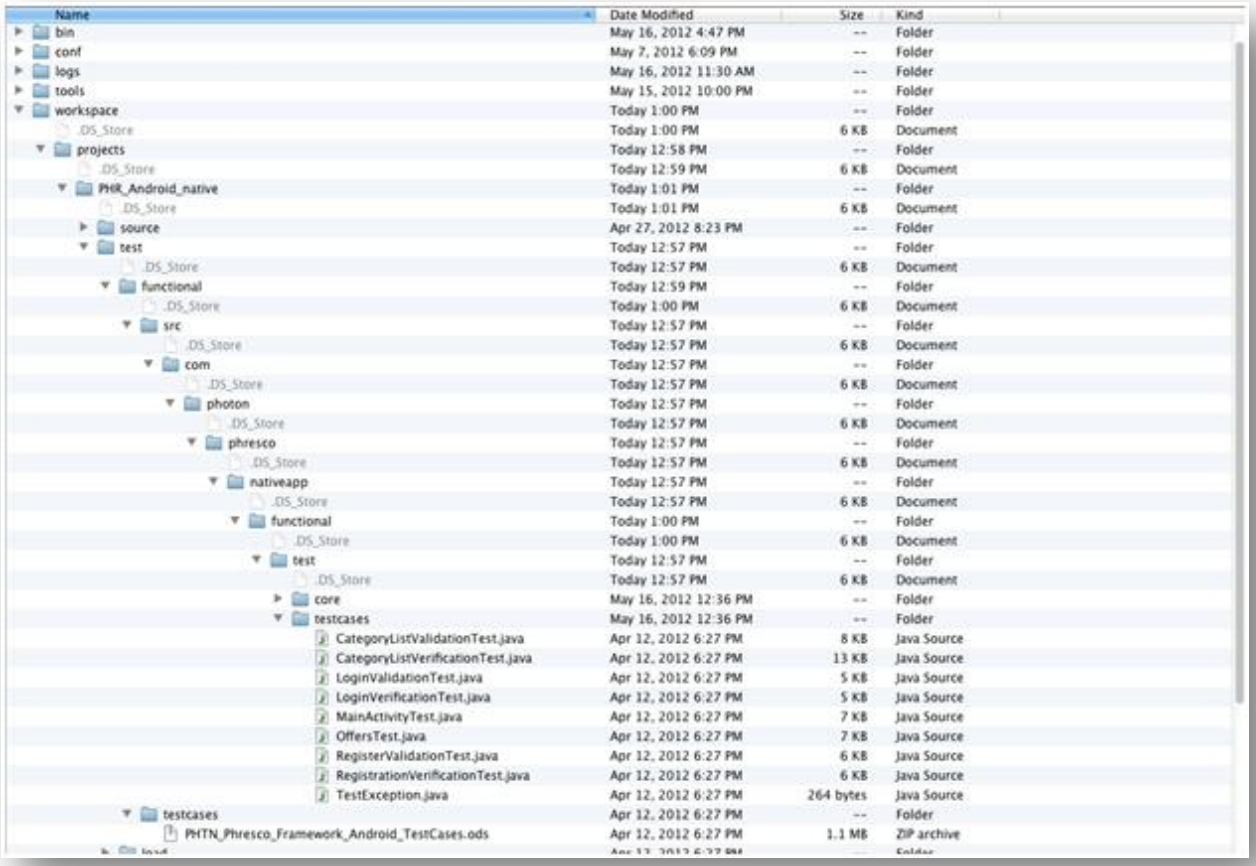


Figure 8-40: Android functional tests structure

a. **MainActivity test:** Main activity test is the root folder that calls all the test cases written separately.This initiates the testing process.
b. **Test cases :** Test cases are the test methods that are called by the main activity test. Test cases can be many in number

Main Activity Test:

Main activity test is a class that calls all the test cases within itself. It contains a bunch of test cases, each of which performs a unique scenario on the application. Test developers can start writing new suite classes and test cases by following the syntax and structure of Phresco framework's out of the box class structure. Once test cases are written developers can execute those from Phresco Framework and can see the report. Following is the file which shows up how to add / include the class inside it.

```
package com.photon.Phresco.nativeapp.functional.test.testcases;

import android.test.ActivityInstrumentationTestCase2;
import android.test.suitebuilder.annotation.Smoke;
import android.util.Log;

import com.jayway.android.robotium.solo.Solo;
import com.photon.Phresco.nativeapp.eshop.activity.MainActivity;

@SuppressWarnings("unchecked")
public class MainActivityTest extends ActivityInstrumentationTestCase2<MainActivity> {

 /**
  * This is suite testcase by this testcase will call other testcases . In
  * static block we are loading the MainActivity class and from the
  * constructor will pass the package and activity full class name then in
  * setUp() created the Solo class object
  *
  */
 public static final String PACKAGE_NAME = "com.photon.Phresco.nativeapp";
 private static final String LAUNCHER_ACTIVITY_FULL_CLASSNAME =
"com.photon.Phresco.nativeapp.eshop.activity.MainActivity";
 private static Class<MainActivity> mainActivity;
 private Solo soloMain;
 private LoginValidationTest loginValid;
 private final String TAG = "MainTestCase****";

 /**
  * This block will be executed first and it will loads the SplashActivity .
  */
 static {
        try {
                mainActivity = (Class<MainActivity>)
Class.forName(LAUNCHER_ACTIVITY_FULL_CLASSNAME);
        }

        catch (ClassNotFoundException e) {
                throw new RuntimeException(e);
        }
 }

 /**
  * In this constructor , we have to send the packagename and activity full
  * class name.
  *
  * @throws Exception
  */
 public MainActivityTest() throws Exception {
        super(PACKAGE_NAME, mainActivity);
 }
```

```
/**
 * this method for create the Solo class object having two super class
 * methods.
 *
 */
@Override
public void setUp() {

        soloMain = new Solo(getInstrumentation(), getActivity());

}
        /**
 * This test method will cAllTestLoginValidation() method.It verifies
 * the Validation for Login screen.
 *
 */

public void testValidationLogin() throws TestException {

        try {
                Log.i(TAG, "testValidationLogin---------Start");
                // creating objectof the class LoginValidationTestCase
                loginValid = new LoginValidationTest(soloMain);
                loginValid.testLoginValidation();
                Log.i(TAG, "testValidationLogin---------End");
        } catch (TestException e) {
                e.printStackTrace();

        }

}
```

Test cases

A test case is a set of conditions under which a tester will determine whether an application is meeting the requirement. This helps the user to verify a particular functionality during the testing process. Elements can be identified and screen shots can be captured when the test fails. Test cases can be added by writing the test cases separately and by calling within the Main activity test.

```
package com.photon.Phresco.nativeapp.functional.test.testcases;

import java.util.ArrayList;
import java.util.Iterator;

import junit.framework.TestCase;
import android.util.Log;
import android.view.View;
```

```
import android.widget.EditText;
import android.widget.ImageView;
import com.photon.Phresco.nativeapp.R;

import com.jayway.android.robotium.solo.Solo;

public class LoginValidationTest extends TestCase {

 private Solo soloLoginValid;
 private String activityName;
 public ImageView clickCancel;
 private String TAG = "*****LoginValidationTestCase******";

 public LoginValidationTest(Solo solo) {
        this.soloLoginValid = solo;

 }

 public void testLoginValidation() throws TestException {

        try {
                Log.i(TAG, "------It is testLoginValidation()----------");

                activityName =
soloLoginValid.getCurrentActivity().getClass().getSimpleName();

                if (activityName.equalsIgnoreCase("MainActivity")) {
                        Log.i(TAG, "------It is MainActivity----------" + activityName);
                        soloLoginValid.waitForActivity("HomeActivity", 2000);

                        for (int i = 0; i < 40; i++) {
                                activityName =
soloLoginValid.getCurrentActivity().getClass().getSimpleName();
                                if (activityName.equalsIgnoreCase("HomeActivity")) {

                                        Log.i(TAG, "------for()-- loop-----");
                                        break;
                                }

                                soloLoginValid.waitForActivity("HomeActivity", 2000);
                        }

                } else {
                        Log.i(TAG, "------ testLoginValidation failed----------");
                        throw new TestException("Current Activity Failed----"
                                        +
soloLoginValid.getCurrentActivity().getClass().getSimpleName() + "failed");
                }

                if (activityName.equalsIgnoreCase("HomeActivity")) {
                        Log.i(TAG, "------HomeActivity----------");
                        System.out.println(" Activity name ---->" +
```

```
soloLoginValid.getCurrentActivity());
                        ArrayList<View> al = soloLoginValid.getViews();
                        Iterator<View> it = al.iterator();
                        while (it.hasNext()) {
                                String viewName = it.next().getClass().getSimpleName();
                                if (viewName.equalsIgnoreCase("ImageView")) {
                                        Log.i(TAG, "------ImageView found----------");
                                        break;
                                }
                                continue;
                        }

                } else {
                        Log.i(TAG, "------HomeActivity not  found----------");
                        throw new TestException(TAG +
soloLoginValid.getCurrentActivity().getClass().getSimpleName() + "failed");
                }
                // click on Loginbutton
                soloLoginValid.waitForActivity("MainActivity", 5000);
                // get the login button view with id i.e home_login_btn
                ImageView loginButton = (ImageView) soloLoginValid
                                .getView(R.id.home_login_btn);
                // click on login button with view id
                soloLoginValid.clickOnView(loginButton);
                // control waits for 2 seconds to activate the screen
                soloLoginValid.waitForActivity("SplashActivity", 2000);
                // clears the text at first Editfield
                EditText emailField = (EditText) soloLoginValid.getView(R.id.txt_email);
                soloLoginValid.clearEditText(emailField);
                // it will type the text at first field which i gave in method
                soloLoginValid.enterText(emailField, "android@");
                // clear the text at second Editfield
                EditText passwordField = (EditText) soloLoginValid
                                .getView(R.id.txt_password);
                soloLoginValid.clearEditText(passwordField);
                // finding the password field view
                // click the password field based on EditText view object
                soloLoginValid.clickOnView(passwordField);
                soloLoginValid.waitForActivity("MainActivity", 1000);
                soloLoginValid.enterText(passwordField, "*****");
                soloLoginValid.waitForActivity("MainActivity", 1000);
                // click on Login button
                ImageView clickLogin = (ImageView) soloLoginValid
                                .getView(R.id.login_btn);
                soloLoginValid.clickOnView(clickLogin);
                // soloSplash.clickOnImageButton(1);
                soloLoginValid.waitForActivity("LoginActivity");
                soloLoginValid.clickOnView(emailField);
                soloLoginValid.waitForActivity("LoginActivity", 1000);
                boolean valid = soloLoginValid.searchText("Invalid Email address!");
                if (valid) {
                        assertTrue("Invalid Email address!", valid);
```

```
                } else {
                        throw new TestException("Testcase failed");
                }
                soloLoginValid.waitForActivity("LoginActivity", 1000);
                // Get the view location of cancel button.
                clickCancel = (ImageView) soloLoginValid.getView(R.id.cancel_btn);
                soloLoginValid.waitForActivity("MainActivity", 1000);
                // click on cancel button
                soloLoginValid.clickOnView(clickCancel);
                soloLoginValid.waitForActivity("LoginActivity", 1000)

        } catch (TestException e) {
                e.printStackTrace();
        }

 }

}
```

8.5.2.2 To Add A New Test Case

You can add a new test case to the Main activity test using the following method. Here is an example for creating a new test case 'testRegististrationValidation'

```
public static final String PACKAGE_NAME = "com.photon.Phresco.nativeapp";
 private static final String LAUNCHER_ACTIVITY_FULL_CLASSNAME =
"com.photon.Phresco.nativeapp.eshop.activity.MainActivity";
 private static Class<MainActivity> mainActivity;
 private Solo soloMain;
 private LoginValidationTest loginValid;
 private testRegististrationValidation;
 private final String TAG = "MainTestCase****";

     @Smoke
 public void testRegististrationValidation() throws TestException {

        try {
                Log.i(TAG, "testRegististrationValidation---------Start");
                // creating object of the testclass RegisterValidationTestCase
                registrationValid = new RegisterValidationTest(soloMain);
                registrationValid.testRegisterValidation();
                Log.i(TAG, "testRegististrationValidation---------End");

        } catch (TestException e) {
                e.printStackTrace();

        }

 }
```

8.5.2.3 Report generated after execution

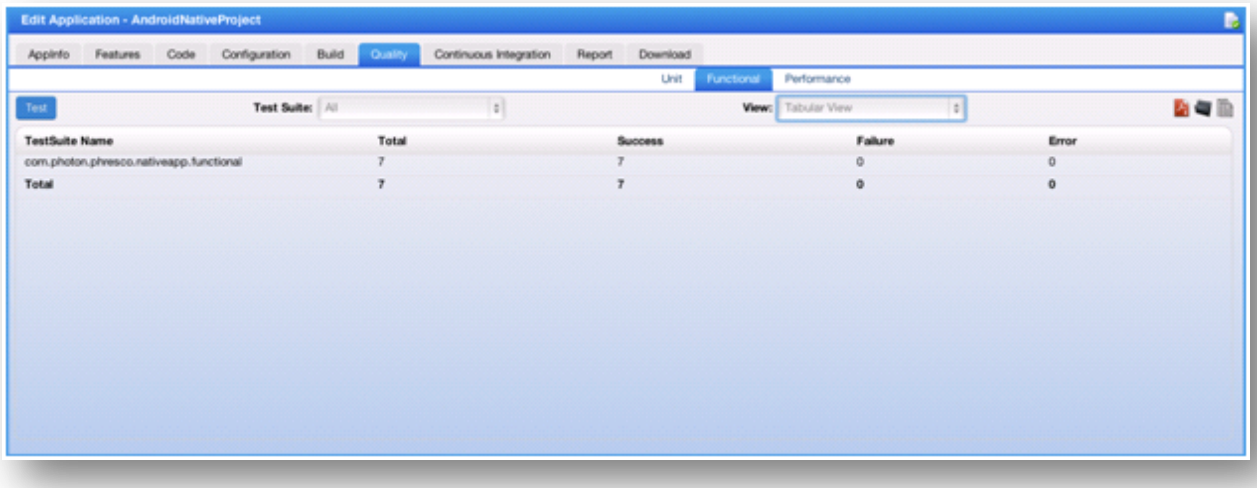


Figure 8-41: ***Android functional test report for all test cases in tabular view***

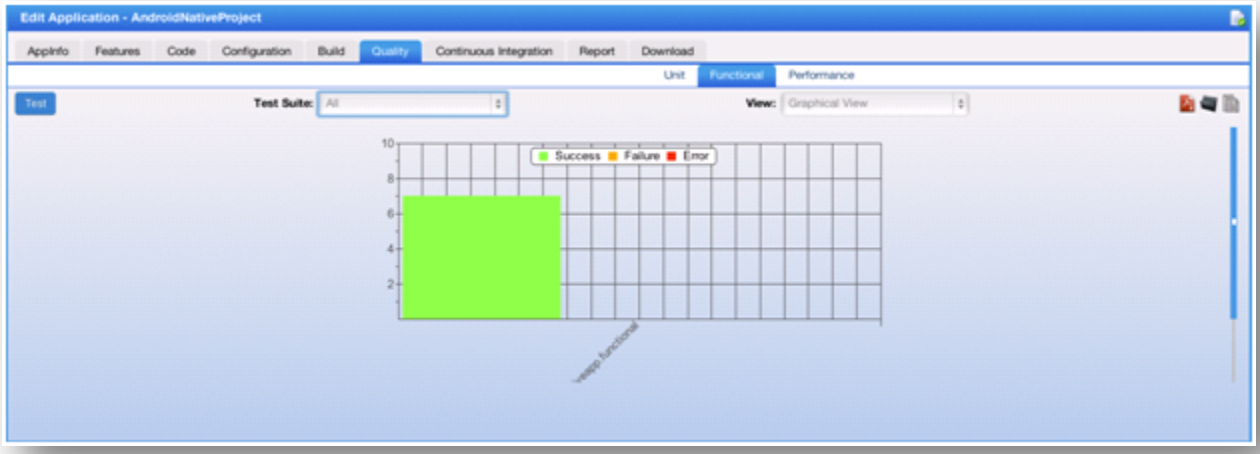


Figure 8-42: ***Android functional test report for all test cases in graphical view***

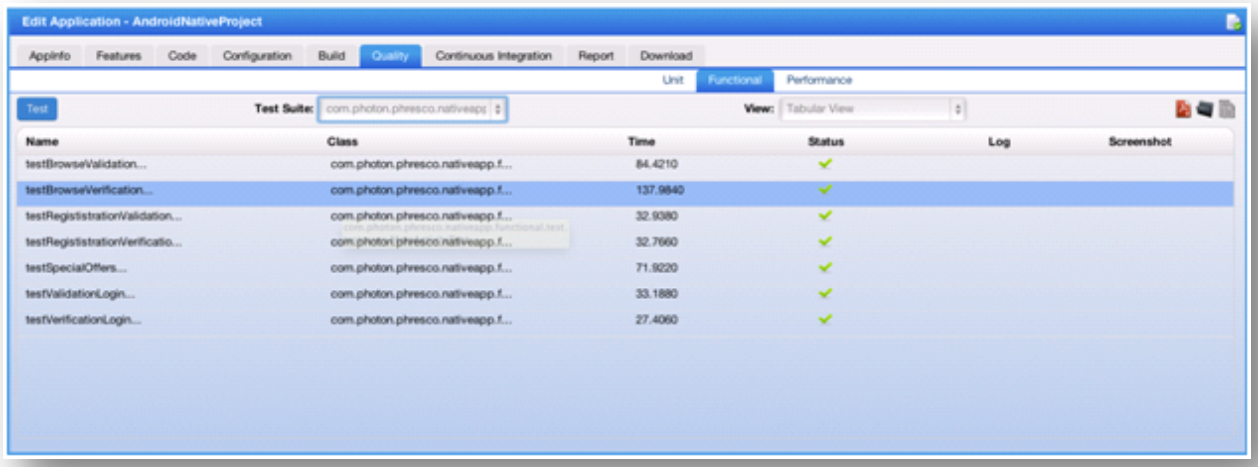


Figure 8-43: ***Android functional test report for single test case in tabular view***

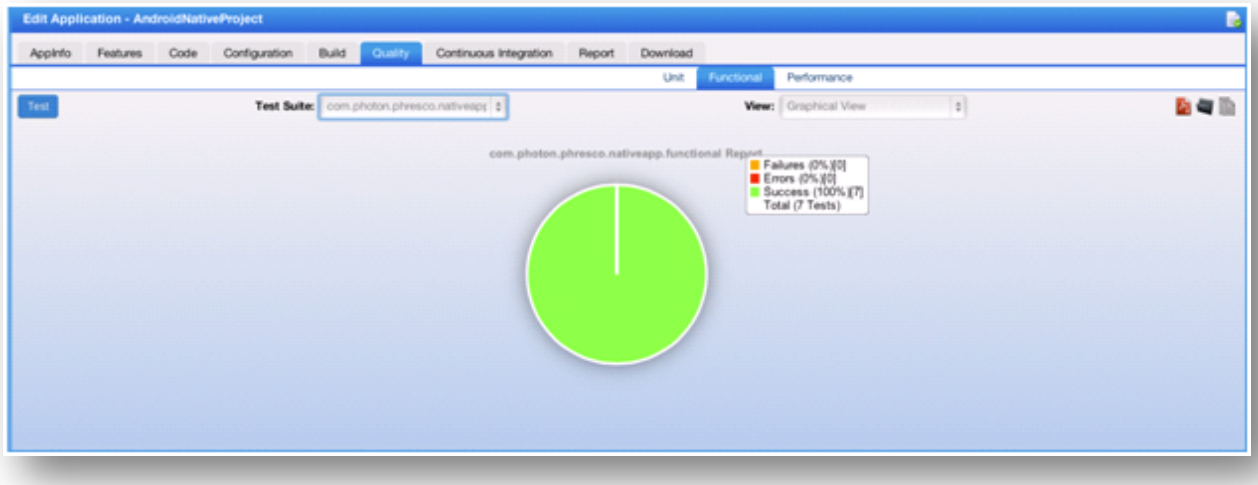


Figure 8-44: ***Android functional test report for single test case in graphical view***

8.5.3 Performance Test for Android

Phresco enables the users to test the performance for their android project. It triggers the test cases simultaneously on all the selected devices and once the tests are completed the results will be available on screen in tabular and graphical formats.

Steps to be followed to trigger performance test

1. Click the Applications-> Project created->Quality->Performance test->Test
2. A Performance Test pop up box appears and the number of android devices can be selected for testing.
3. When three devices are connected the performance testing occurs simultaneously as shown in figure:

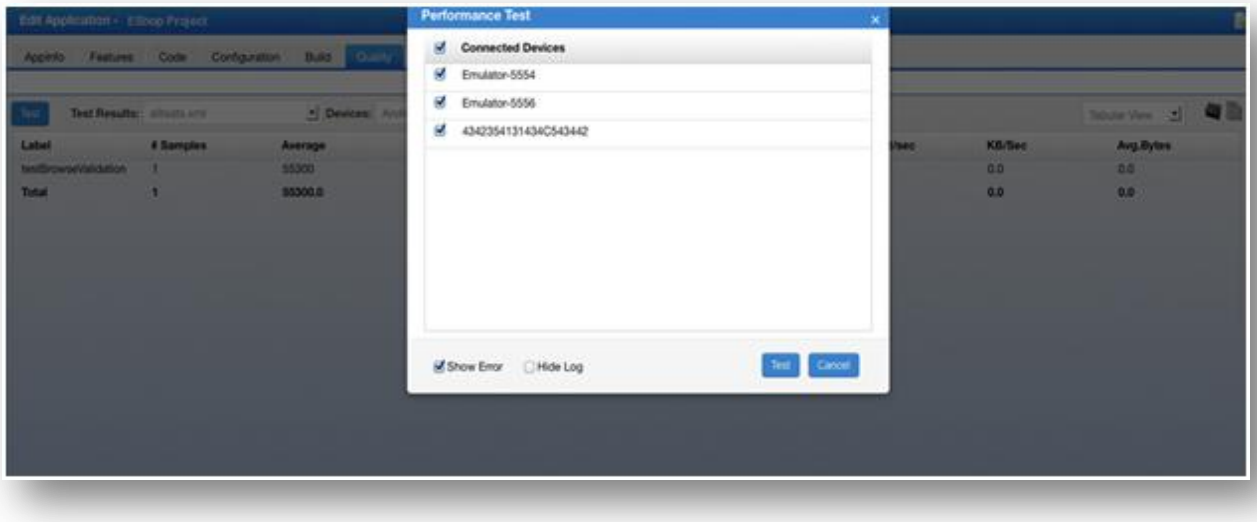


Figure 8-45: Choosing multiple devices for performance test.

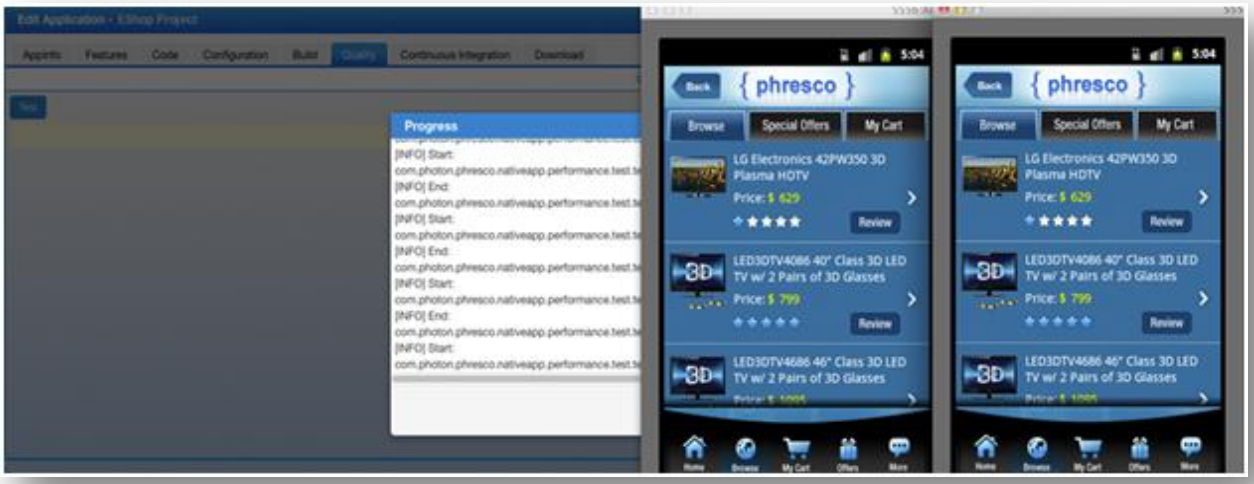


Figure 8-46: Performance test in multiple devices

8.5.4 Report generated after execution

Report for the performance testing is generated in both graphical and tabular form. The response time and the throughput are given as the report after the performance testing.

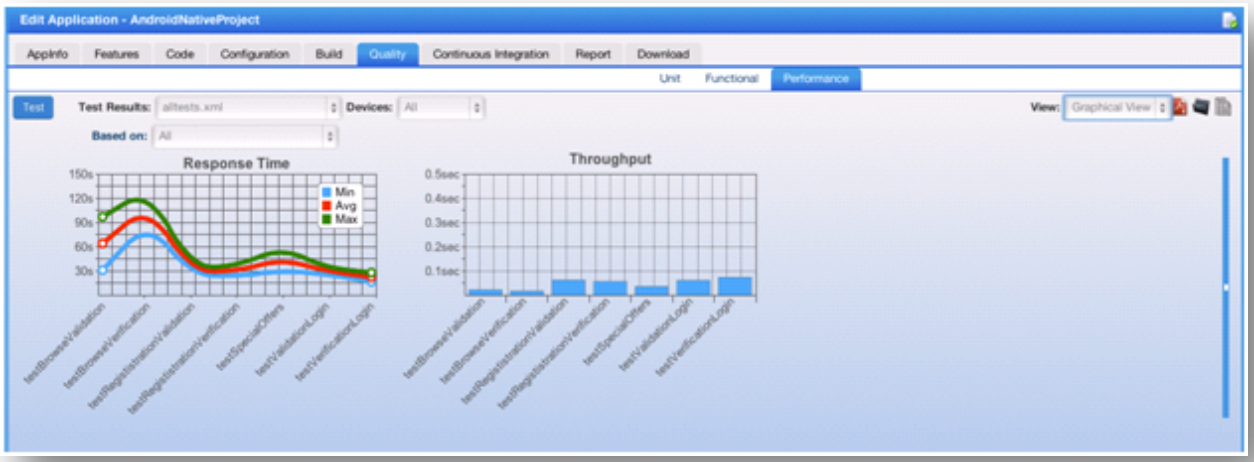


Figure 8-47: Android graphical report for performance test

8.6 iPhone Applications

8.6.1 Unit Test Case

8.6.1.1 Structure of Alltest and Test Cases

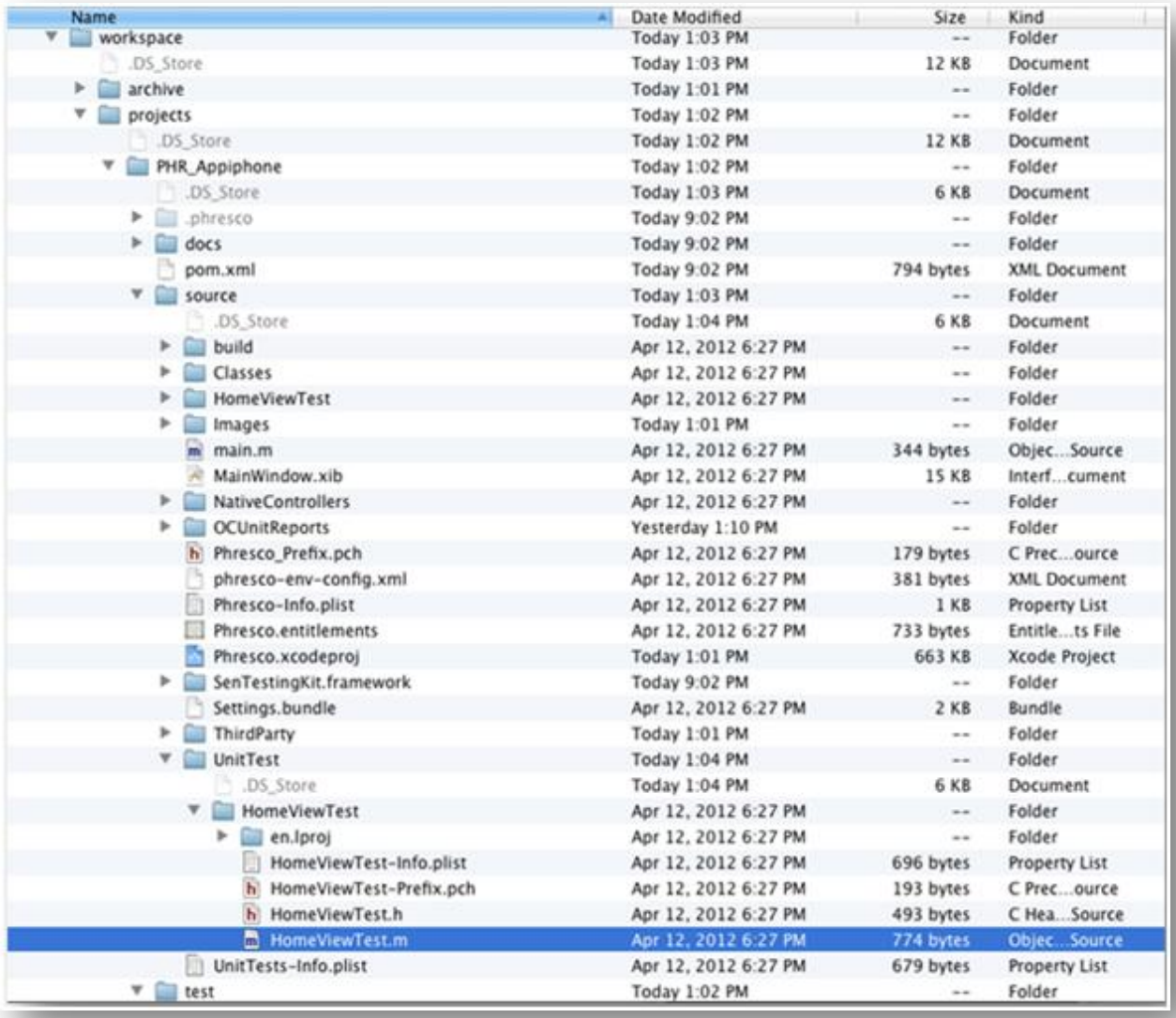


Figure 8-48: iPhone unit tests structure

a. **Home view test:** This is a test suite in which all other test cases are called.

b. **Test cases:** Test cases are present inside the test suites

Homeviewtest for iPhone Application

The testSuite contains a bunch of test cases, each of which performs a unique scenario on the application. Test developers can start writing new suite classes and test cases by following the syntax and structure of Phresco framework's out of the box class structure. HomeViewTest class calls all the suite classes and the test cases within it. Once suite classes and test cases are written developers can execute those from Phresco Framework and can see the report. Following is the HomeViewTest files which shows up how to add / include the class inside it.

```
#import <SenTestingKit/SenTestingKit.h>
#import <UIKit/UIkit.h>

#import "PhrescoAppDelegate.h"
#import "RootViewController.h"
#import "HomeViewController.h"
#import "BrowseViewController.h"
#import "ResultViewController.h"
#import "ProductDetailsViewController.h"
#import "AddToBagViewController.h"
#import "ViewCartController.h"
#import "CheckOutViewController.h"
#import "CheckOutOVerallViewController.h"
#import "ReviewViewController.h"
#import "ReviewCommentsViewController.h"
#import "LoginViewController.h"
#import "RegistrationViewController.h"
HomeViewTest.h

@interface HomeViewTest : SenTestCase{

@private
iShopAppDelegate *appDelegate;
RootViewController *rootController;
HomeViewController *homeController;
BrowseViewController *browseController;
ResultViewController *resultController;
ProductDetailsViewController *pdtDetailController;
AddToBagViewController *addCartController;
ViewCartController *viewCartController;
CheckOutViewController *checkViewController;
CheckOutOVerallViewController *checkOverallController;
ReviewViewController *reviewController;
ReviewCommentsViewController *reviewCommentsController;
LoginViewController *loginController;
RegistrationViewController *registerController;
```

```
    UITableView *tblView;
}
-(void) testAction;
@end

//////



- (void)testExample{

    [rootController tabBarButtonAction:@"0"];
  // STFail(@"Unit tests are not implemented yet in HomeViewTest");
}
-(void) testAction{

    [homeController buttonAction:@"0"];

}
-(void) testBrowse{


    //[browseController  tableView:tblView didSelectRowAtIndexPath:0];

    STAssertThrows([browseController tableView:tblView didSelectRowAtIndexPath:-1] ,
@"Invalid index");


}
-(void) testProductResult{

    [resultController  tableView:tblView didSelectRowAtIndexPath:0];
    [resultController reviewButtonSelected:@"0"];

}
-(void) testProductDetail{

    [pdtDetailController addToCart:@"0"];


}


-(void) testAddToCart{

   [addCartController removeIndex:@"0"];


}
-(void) testViewCart{

    [viewCartController browseButtonSelected:@"0"];
```

```
}
-(void) testCheckCart{

    [checkViewController cancelAction:@"o"];

}
-(void) testCheckOverall{

    [checkViewController reviewAction:@"o"];

}
-(void) testReview{

    [reviewController tableView:tblView didSelectRowAtIndexPath:o];

}
-(void) testComments{

    [reviewCommentsController goBack:o];

}

-(void) testLogin{

    [loginController registerButtonSelected:o];

}

-(void) testRegister{

    [registerController registerButtonSelected:o];

}
@end
```

Test Register Example for a Test Case

A test case is a set of conditions under which a tester will determine whether an application is meeting the requirement. This helps the user to verify a particular functionality during the testing process.

The following is the example of a test register

```
-(void) testRegister{

    [registerController registerButtonSelected:o];
 }
```

8.6.1.2 To Add New Test Case in Homeviewtest

You can add a new test case to the Homeviewtest using the following method. Here is an example for creating a new test case 'testsploffers'.

```
-(void) testSplOffers{

    [splOfferController tableView:tblView didSelectRowAtIndexPath:0];

}
```

8.6.1.3 Report generated after execution

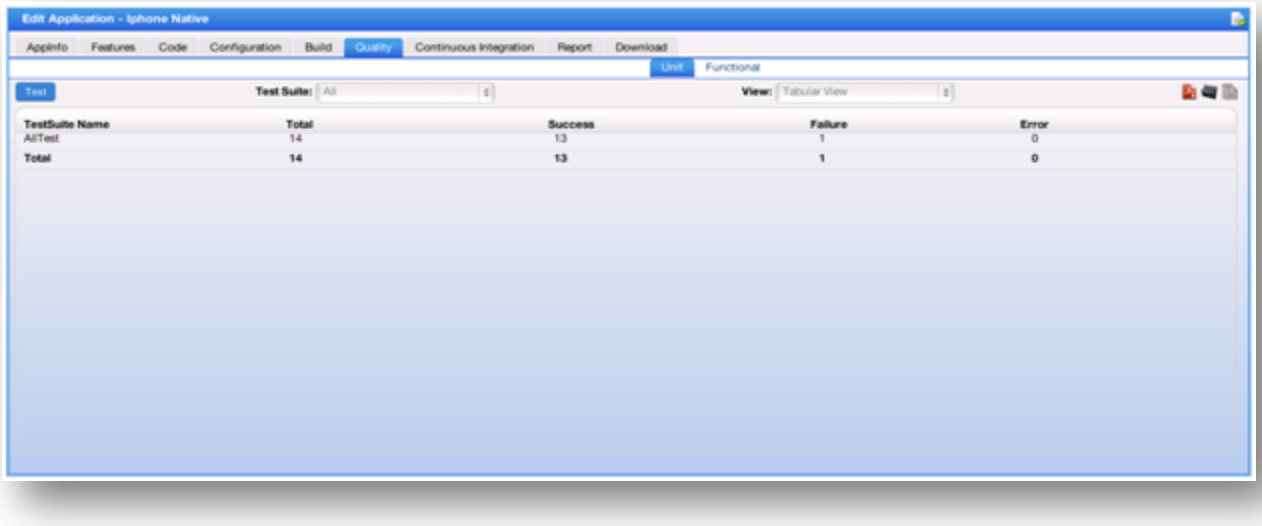


Figure 8-49: ***iPhone unit test report for single test case in tabular view***

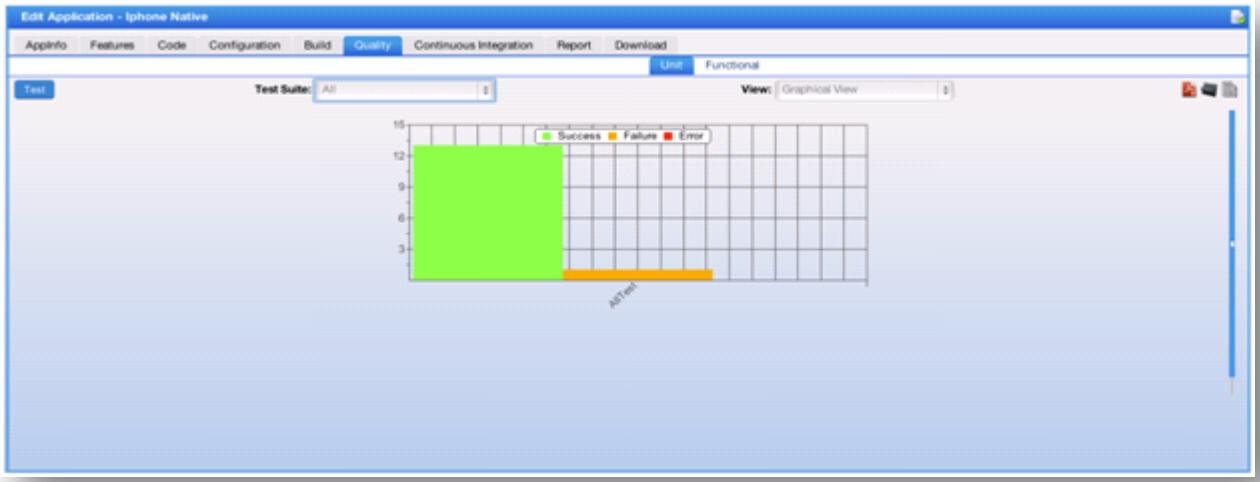


Figure 8-50: iPhone unit test report for all test cases in graphical view

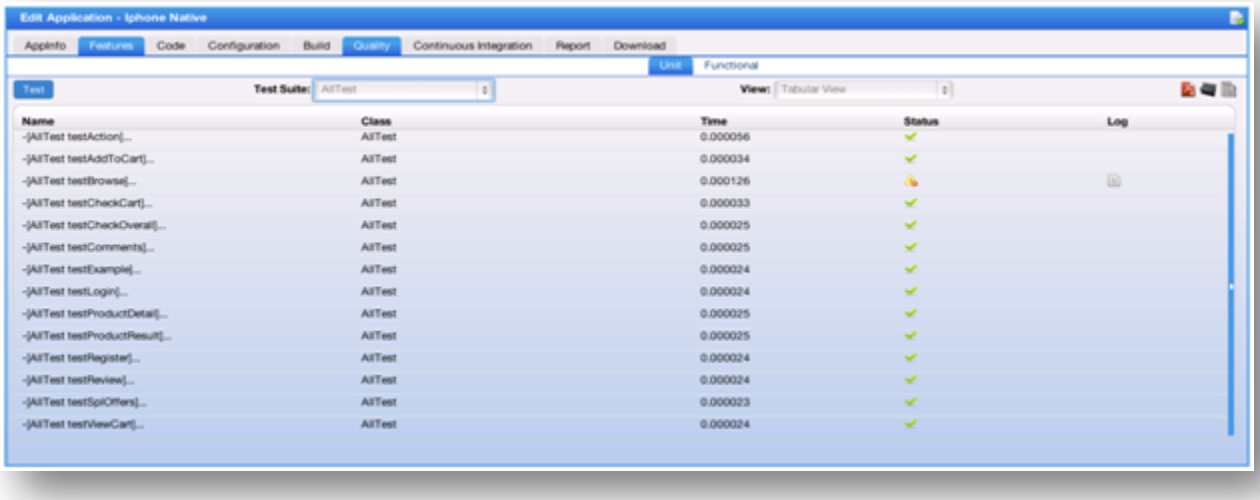


Figure 8-51: *iPhone unit test report for all test cases in tabular view*

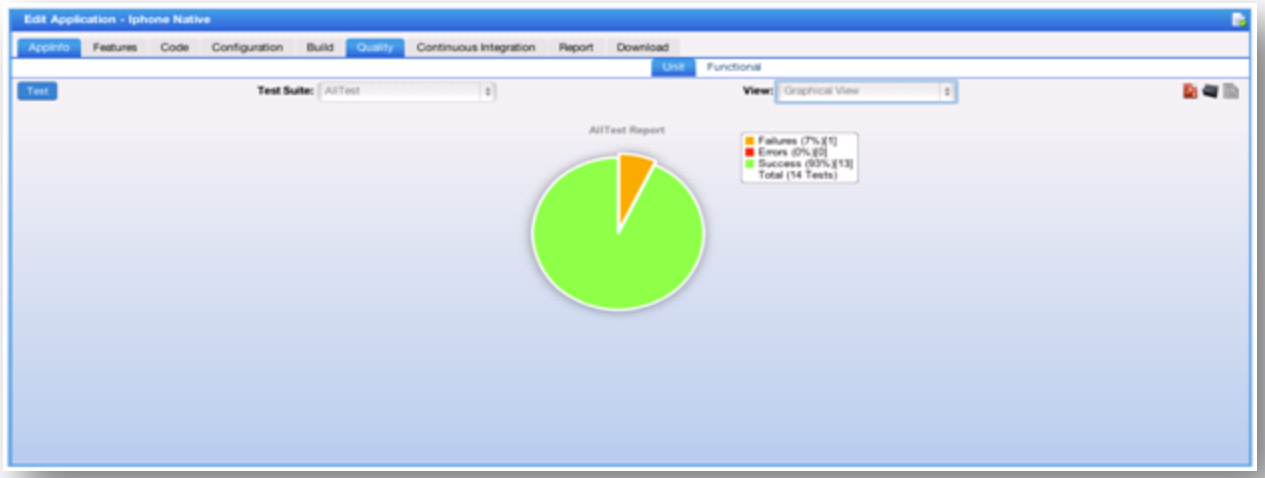


Figure 8-52: ***iPhone unit test report for single test case in graphical view***

8.6.2 Functional Test Case

8.6.2.1 Structure of Alltest and Test Case

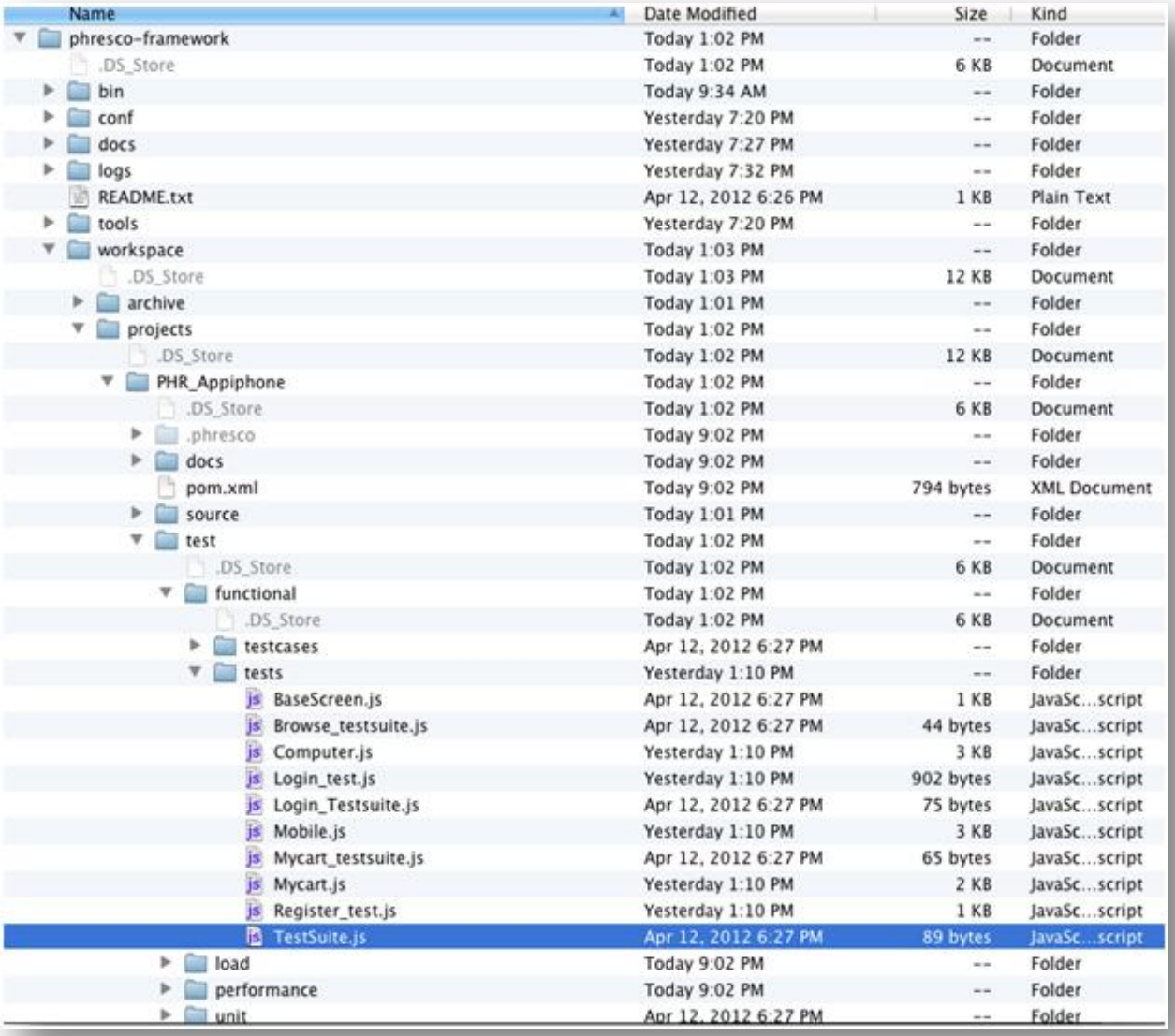


Figure 8-53: iPhone functional tests structure

a. **Test Suite:** This is the class which calls all other test suites and test cases.
b. **Test cases:** Test cases are called by the test suites and they are written separately.

8.6.2.2 Existing Test Cases and Suites Out Of the Box in Phresco

Test suite for iPhone

Testsuite is the class which holds all the other test suites. Functional test runs in the order by which the test suites are created. The testSuite contains a bunch of test cases, each of which performs a unique scenario on the application. Test developers can start writing new suite classes and test cases by following the syntax and structure of Phresco framework's out of the box class structure. Test suite calls all the suite classes and the test cases within it. Once suite classes and test cases are written testers can execute those from Phresco Framework and can see the report. Following is the Test suite files which shows up how to add / include the class inside it.

```
#import "Login_Testsuite.js"
#import "Mycart_testsuite.js"
```

Test Suite example for Login Test Suite

Test suite is a collection of test cases that are intended to be used to test a project code to show that it has some specified set of behaviors. A test suite contains detailed instructions or goals for each collection of test cases. New test cases can also be added by using the same syntax '#import.'

```
#import "BaseScreen.js"
#import "Register_test.js"
```

Test case example for Register_test

A test case is a set of conditions under which a tester will determine whether an application is meeting the requirement. This helps the user to verify a particular functionality during the testing process. Elements can be identified and screen shots can be captured when the test fails.

```
#import "BaseScreen.js"
function Register_Test(){

   var reg= "Register_Test";
   var app=UIATarget.localTarget();
```

```
    var target=app.frontMostApp();
    var main=target.mainWindow();
    var button1=main.buttons()[register].tap();
    target.logElementTree();
    main.scrollViews()[0].textFields()[0].setValue(first);
       app.delay(2);
    main.scrollViews()[0].textFields()[1].setValue(last);
       app.delay(2);
    main.scrollViews()[0].textFields()[2].setValue(Email_id);
       app.delay(2);
    main.scrollViews()[0].secureTextFields()[0].setValue(password);
       app.delay(2);
    main.scrollViews()[0].secureTextFields()[1].setValue(password);
       app.delay(2);
   var buttons = main.buttons();
    target.keyboard().buttons()[RETURN].tap();
    app.delay(2);
    buttons[register2].tap();
    app.delay(2);
    main.buttons()[1].tap();
    app.delay(5);
    UIALogger.logPass(reg);

}
UIALogger.logStart("Iphone Test");
Register_Test();
```

8.6.2.3 To Add New Test Suite in Main Test Suite

You can add a new test suite to the Main Test Suite using the following method.

```
#import "Login_Testsuite.js"
#import "Mycart_testsuite.js"
#import "Browse_testsuite.js"
```

8.6.2.4 To Add New Test Case in Test Suite

You can add a new test case to the Test suite using the following method.

```
#import "BaseScreen.js"
#import "Register_test.js"
#import "Login_test.js"
```

8.6.2.5 Report generated after execution

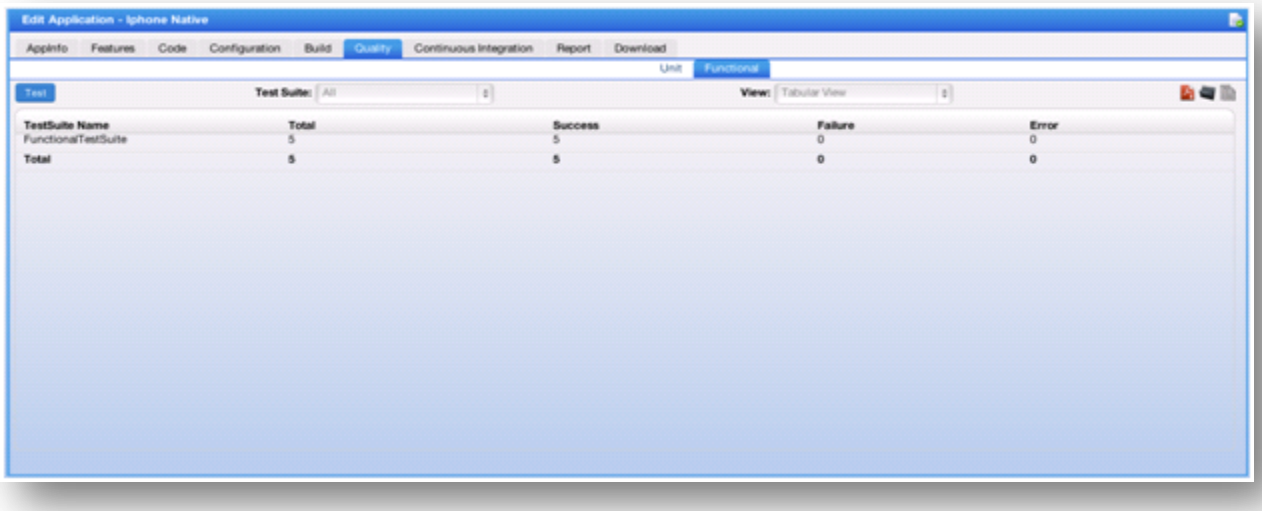


Figure 8-54: ***iPhone functional test report for single test case in tabular view***

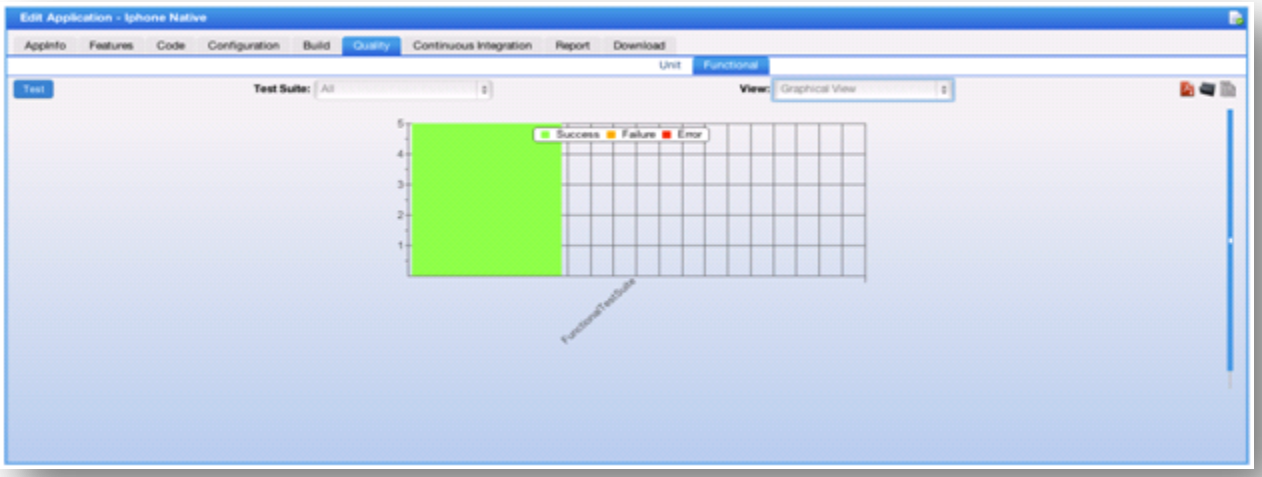


Figure 8-55: ***iPhone functional test report for all test cases in graphical view***

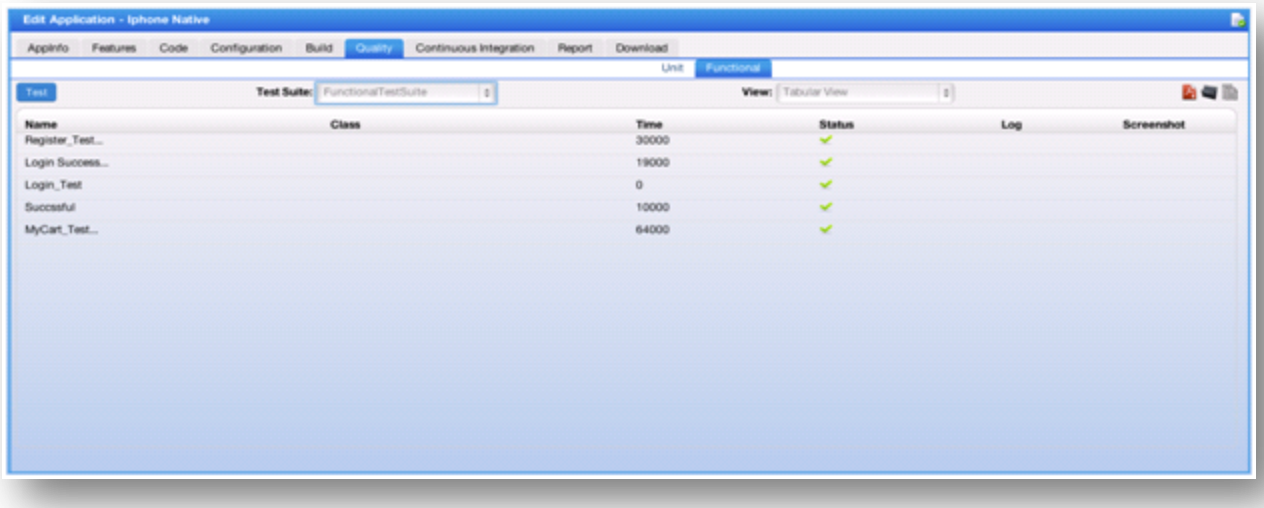


Figure 8-56: ***iPhone functional test report for all test cases in tabular view***

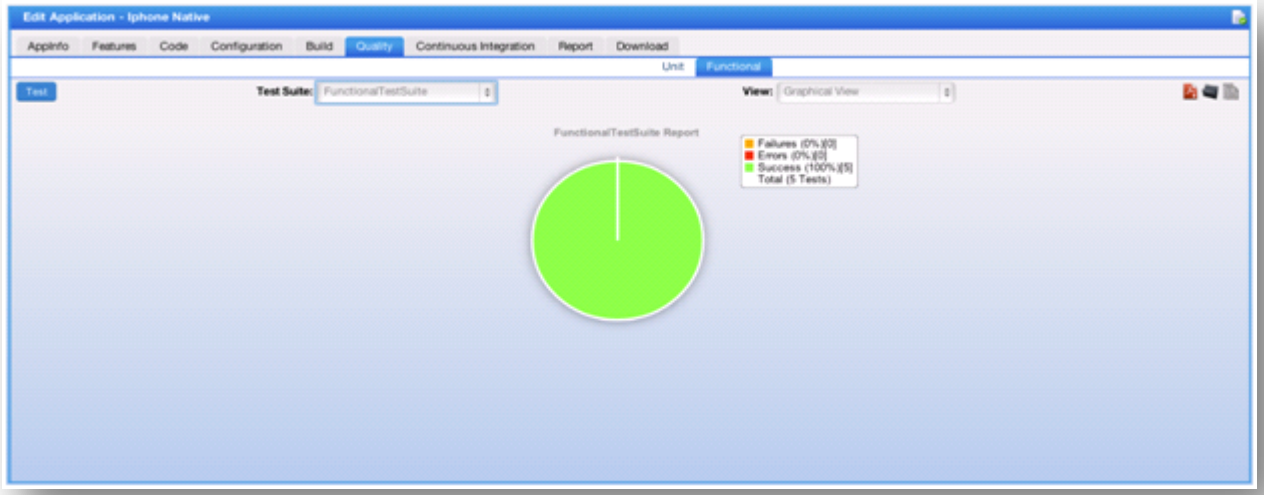


Figure 8-57: ***iPhone functional test report for single test case in graphical view***

8.7 Node js Test Cases

8.7.1 Unit Test Case

8.7.1.1 Structure of Test Suites and Test Case

a. **AllTest**: Alltest is the root file that carries the Test suite.

b. **Test suite**: Test suite is made to run through AllTest

c. **Test case**: All the Test cases should be added within the test suite.

8.7.1.2 Existing Test Cases and Suites Out Of the Box in Phresco

Alltest for Node js

AllTest is the root category which holds the test suite. AllTest contains a bunch of test suites, each of which performs a unique scenario on the application. Test developers can start writing new suite classes by following the syntax and structure of Phresco framework's out of the box class structure. Once test suite classes and test cases are written developers can execute those from Phresco Framework and can see the report. Following is the AllTest file which shows up how to add / include the class inside it.

```
        var nodeunit = require('nodeunit');
var reporter = nodeunit.reporters.junit;

var opts = {
    "error_prefix": "\u001B[31m",
    "error_suffix": "\u001B[39m",
    "ok_prefix": "\u001B[32m",
    "ok_suffix": "\u001B[39m",
    "bold_prefix": "\u001B[1m",
    "bold_suffix": "\u001B[22m",
    "assertion_prefix": "\u001B[35m",
    "assertion_suffix": "\u001B[39m"
}

opts.output = "target/surefire";

reporter.run(['source/test/eshop'], opts);
```

Test Suites example

Test suite is a collection of test cases that are intended to be used to test a project code to show that it has some specified set of behaviors. A test suite contains detailed instructions or goals for each collection of test cases. New test cases can also be added by using the following syntax.

Test case example

A test case is a set of conditions under which a tester will determine whether an application is meeting the requirement. This helps the user to verify a particular functionality during the testing process.

```
   exports.testSomething = function(test){
    test.expect(1);
    test.ok(true, "this assertion should pass");
    test.done();
};

exports.testSomethingElse = function(test){
    test.ok(true, "this assertion should fail");
    test.done();
};
```

8.7.1.3 To Add New Test Case in Test Suite

You can add a new test case to the Test suite using the following method.

```
 exports.testSomething = function(test){
    test.expect(1);
    test.ok(true, "this assertion should pass");
    test.done();
 };

 exports.testSomethingElse = function(test){
    test.ok(true, "this assertion should fail");
    test.done();
 };
```

8.7.1.4 Report generated after execution

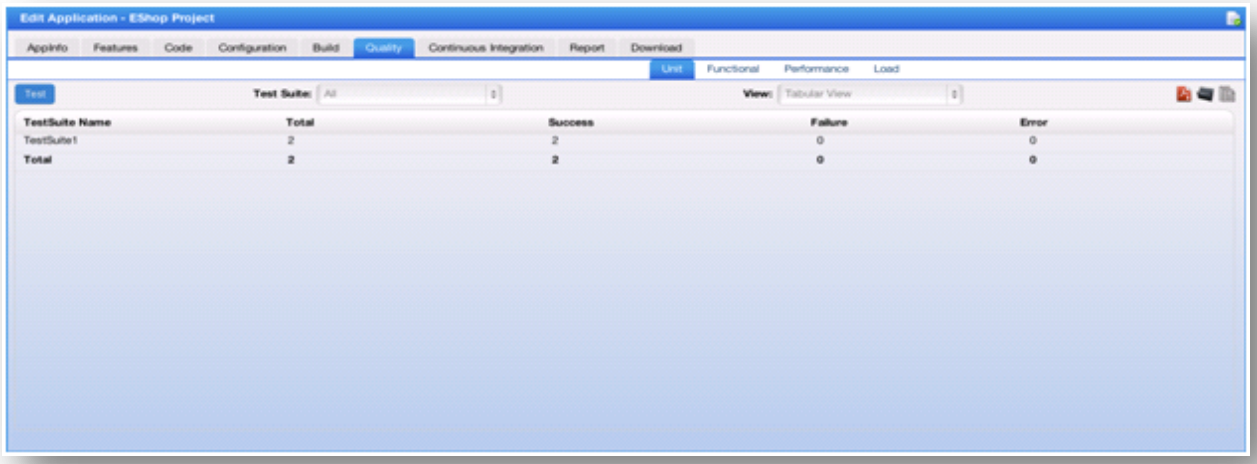


Figure 8-58: ***NodeJS unit test report for all test cases in tabular view***

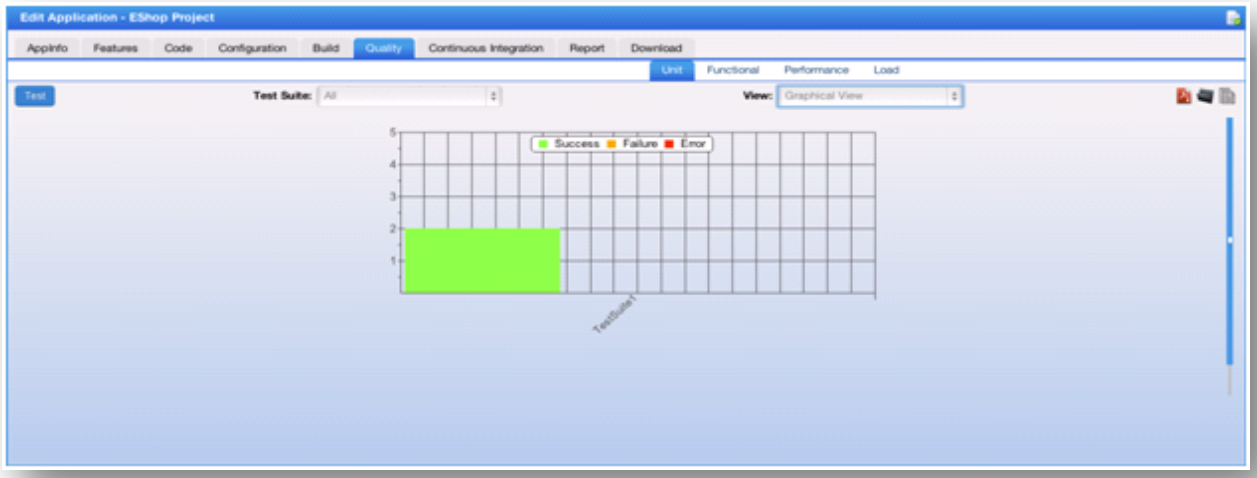


Figure 8-59: ***NodeJS unit test report for all test cases in graphical view***

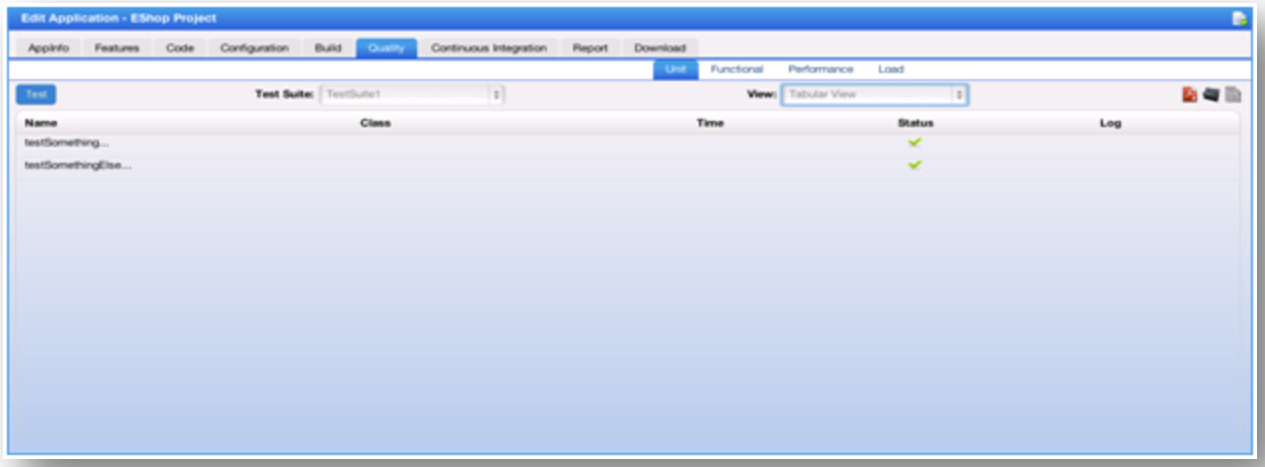


Figure 8-60: ***NodeJS unit test report for single test case in tabular view***

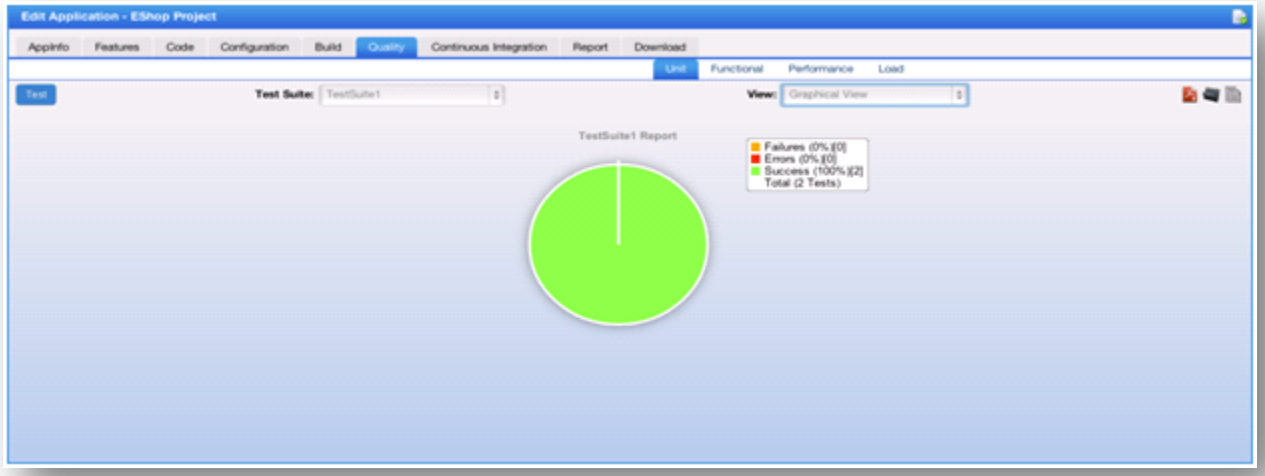


Figure 8-61: ***NodeJS unit test report for single test case in graphical view***

8.8 jQuery Test Cases

8.8.1 Unit Test Cases

8.8.1.1 Structure of Test Case

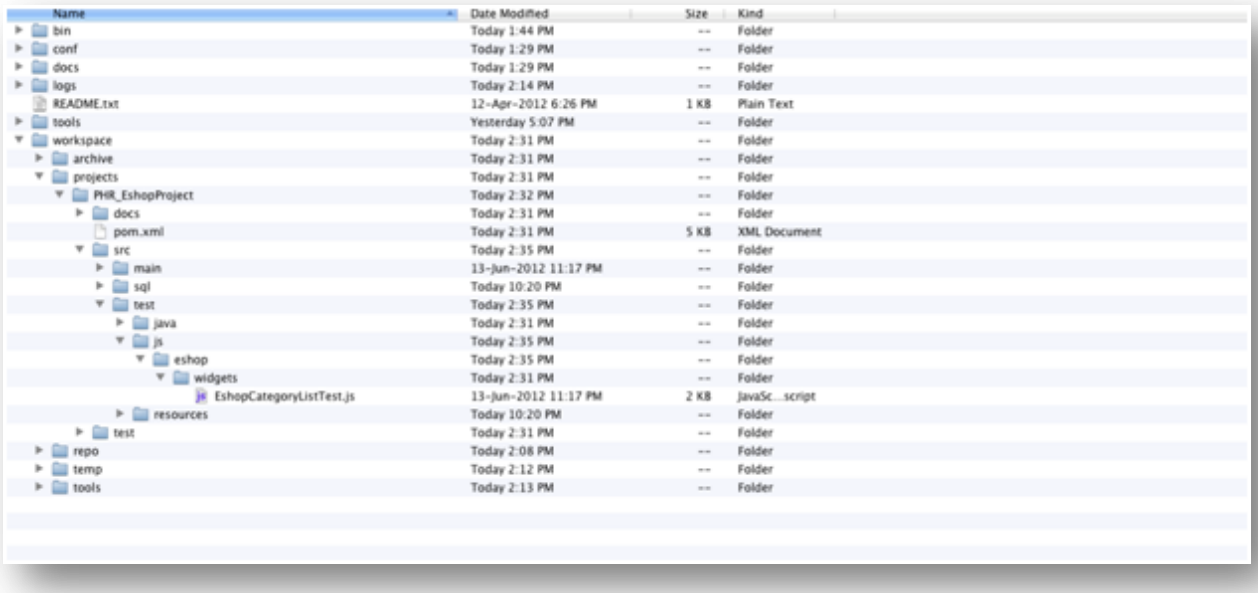


Figure 8-62: jQuery unit test case structure

a. **Test cases:** Test cases can be added directly in the js folder.

```
/*global require */

require([ "jQuery", "./Category", "./EShopAPI", "qunit" ], function($, Category, EShopAPI,
QUnit) {

        var equal = QUnit.equal, expect = QUnit.expect, test = QUnit.test;

        /**
         * Test that the setMainContent method sets the text in the category-widget
         */
        test("category-widget unite test case passed.", function() {

                var category, initeObj, listener, api, Phresco, mainContent, currentName,
currentID, navUL, output1, output2;

                // Setup view and call method under test
                category = new Category();
                api = new EShopAPI();

    //api.wsURL = "http://172.16.25.75:2020/eshop";
```

```
                api.getWsConfig();

        category.api = api;
                category.listener = undefined;
                category.Phrescoapi = undefined;

                output1 = category.renderUI();

                mainContent = $('<section id="submenu">');
        currentName = 'name';
        currentID = 'id';
        navUL = $('<ul></ul>');

                api.getCategories(function(jsonObject){

                        var productList = jsonObject,
                        totalCategories = productList.category.length,
                    i, category, categoryId, lis;

                        for (i = 0; i < totalCategories; i++) {
                                category = productList.category[i];
                    categoryId = category.id;
                    lis = $('<li><span><a href="javascript:void(0);">' + category[currentName]
 + '</a></span></li>');
                                navUL.append(lis);
                        }
                    });

        output2 = mainContent.append(navUL);

                // Expect that the text was set on the expected element
                equal(output1.html(), output2.html(),       "Expected text not set in
            category-widget");
            });
});
```

8.8.1.2 Report generated after execution

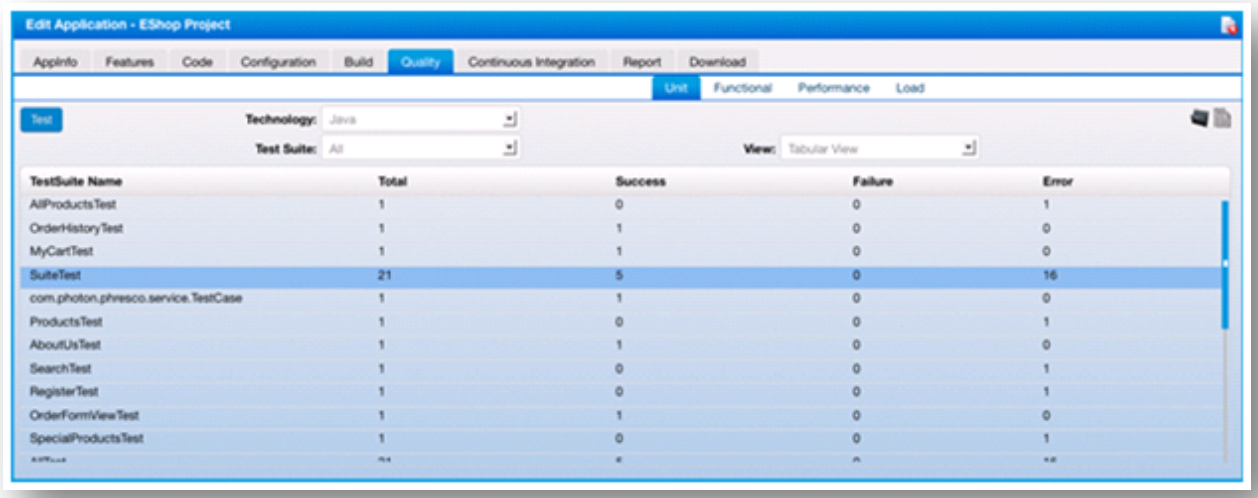


Figure 8-63: jQuery unit test report for all test cases in tabular view

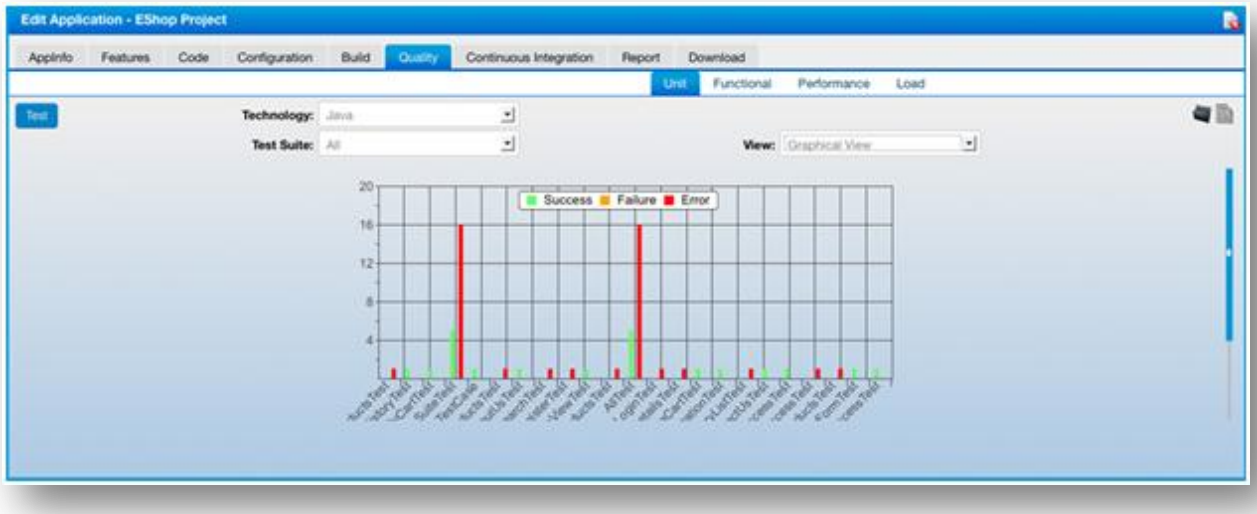


Figure 8-64: *jQuery unit test report for all test cases in Graphical view*

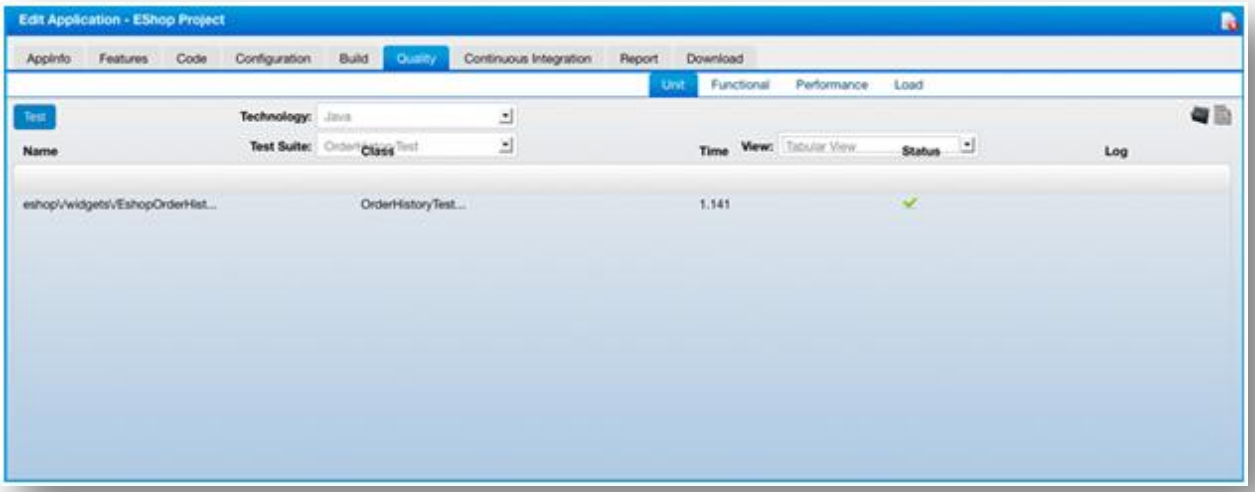


Figure 8-65: ***jQuery unit test report for single test case in tabular view***

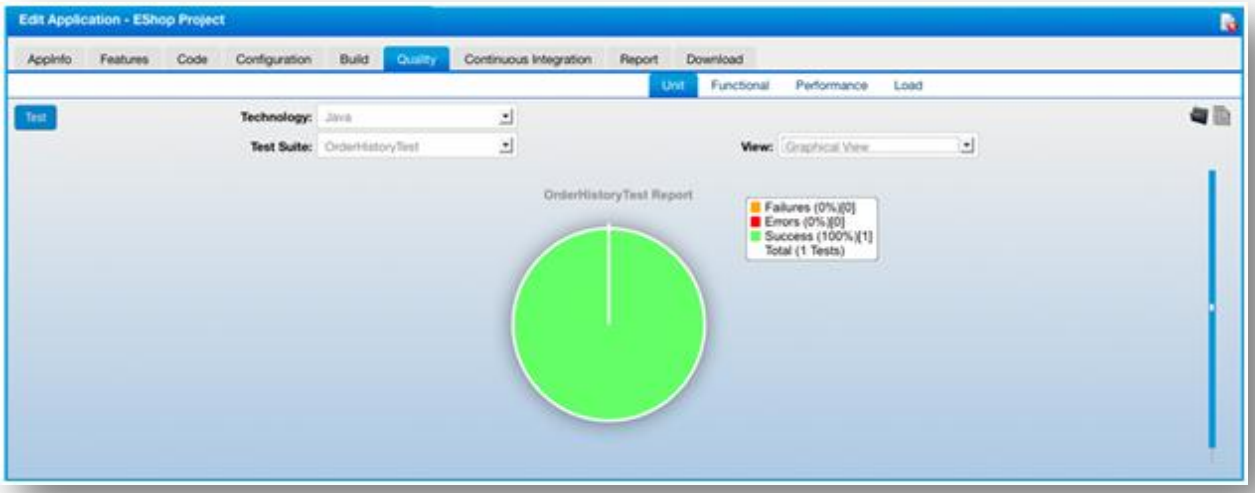


Figure 8-66: ***jQuery unit test report for single test case in graphical view***

Prerequisites for jQuery Unit Test Cases:

A prior installation of Phantomjs software is essential to run Test cases for jQuery project. This software is available in the Phresco framework->Downloads.

- Set environment variable for the phantomjs software.
- Extract the zip file and set the path {installation path}\phantomjs-1.5.0-win32-static in the system variables field.

8.8.2 Functional Test Cases

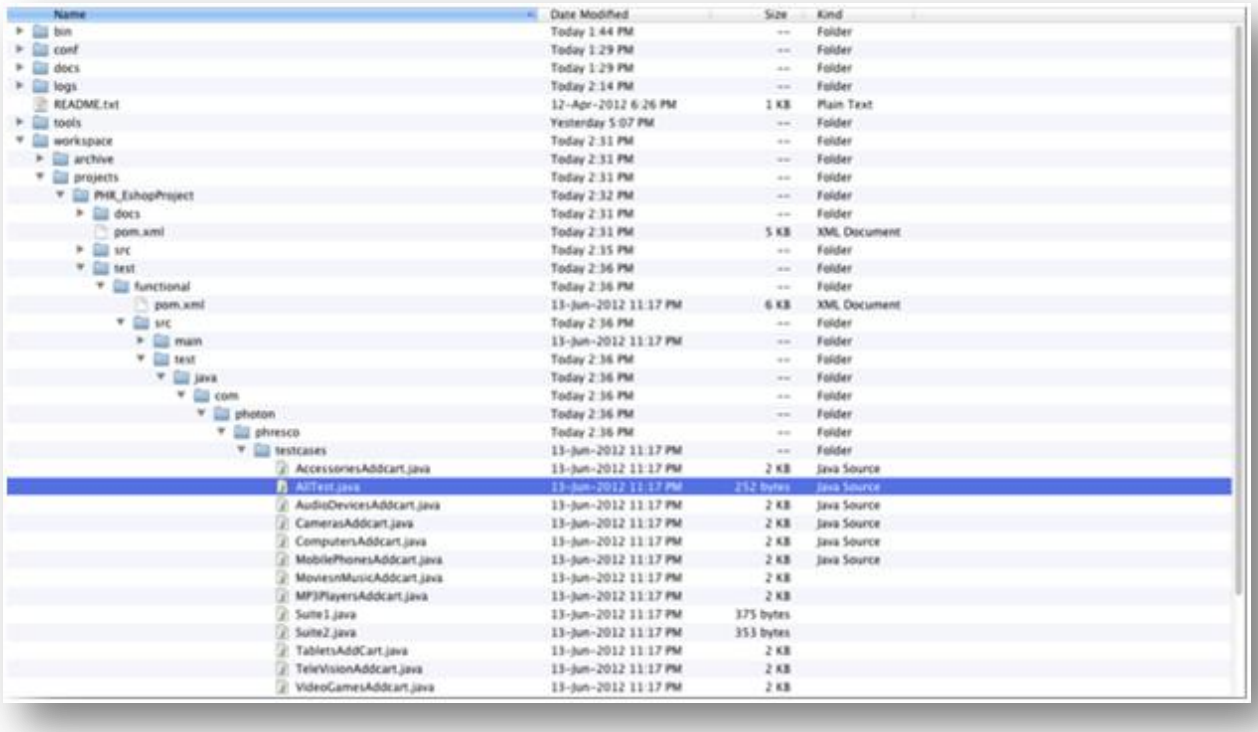


Figure 8-67: jQuery functional test case structure

a. AllTest: This is a root JUnit suite class that can either call a suite of classes or individual test cases.

b. Suite Class: This is also a JUnit suite class which calls the individual test cases.

c. Test Cases: These are individual java classes which perform unique testing scenarios against the application.

8.8.2.1 Existing Test Cases and Suites Out Of the Box in Phresco

In Phresco testing Framework in Junit suite class is named as “AllTest”.

The testSuite contains a bunch of test cases, each of which performs a unique scenario on the application. Test developers can start writing new suite classes and test cases by following the syntax and structure of Phresco frameworks out of the box class structure. AllTest class calls all the suite classes and the test cases within it. Once suite classes and test cases are written developers can execute those from Phresco Framework and can see the report. Following is the AllTest file which shows up how to add / include the class inside it.

```
Package com.photon.Phresco.testcases

import org.junit.runner.RunWith;
import org.junit.runners.Suite;
import org.junit.runners.Suite.SuiteClasses;

@RunWith(Suite.class)
@SuiteClasses({Suite1.class,Suite2.class})
public class AllTest {
}
```

Suite class

Suite class is a collection of test cases that are intended to be used to test a project code to show that it has some specified set of behaviors. A suite class contains detailed instructions or goals for each collection of test cases. New test cases can also be added by using the same syntax.

```
package com.photon.Phresco.testcases;
import org.junit.runner.RunWith;
import org.junit.runners.Suite;
import org.junit.runners.Suite.SuiteClasses;

@RunWith(Suite.class)
@SuiteClasses({ WelcomePage.class,TeleVisionAddcart.class,
                ComputersAddcart.class,
MobilePhonesAddcart.class,AudioDevicesAddcart.class, CamerasAddcart.class

})
public class Suite1 {

}
```

Test cases

A test case is a set of conditions under which a tester will determine whether an application is meeting the requirement. This helps the user to verify a particular functionality during the testing process. Elements can be identified and screen shots can be captured when the test fails.

```
package com.photon.Phresco.testcases;

import java.io.IOException;

import junit.framework.TestCase;

import org.junit.Test;
//import static org.testng.Assertjunit.*;
import org.openqa.selenium.server.SeleniumServer;

import com.photon.Phresco.Screens.MenuScreen;
import com.photon.Phresco.Screens.WelcomeScreen;
import com.photon.Phresco.selenium.report.Reporter;
import com.thoughtworks.selenium.Selenium;
import com.photon.Phresco.uiconstants.PhrescoUiConstants;

public class AWelcomePage extends TestCase {
```

```
private SeleniumServer serv;
protected Selenium selenium;
private PhrescoUiConstants phrsc;
private int SELENIUM_PORT;
private String browserAppends;

@Test
public void testWel() throws InterruptedException, IOException, Exception {
        try {
                phrsc = new PhrescoUiConstants();
                String serverURL = phrsc.PROTOCOL + "://"
                                + phrsc.HOST + ":"
                                + phrsc.PORT + "/";
                browserAppends = "*" + phrsc.BROWSER;
                assertNotNull("Browser name should not be null",browserAppends);
                SELENIUM_PORT = Integer.parseInt(phrsc.SERVER_PORT);
                assertNotNull("selenium-port number should not be null",
                                SELENIUM_PORT);
                WelcomeScreen wel=new WelcomeScreen(phrsc.SERVER_HOST,
SELENIUM_PORT,
                                browserAppends, serverURL, phrsc.SPEED,
                                phrsc.CONTEXT );
                assertNotNull(wel);
                MenuScreen menu = wel.menuScreen();
        assertNotNull(menu);

        } catch (Exception t) {
                t.printStackTrace();
                System.out.println("ScreenCaptured");
                selenium.captureEntirePageScreenshot("\\WelPageFails.png",
                                "background=#CCFFDD");
        }
}

@Override
public void setUp() throws Exception {

        serv = new SeleniumServer();
        try {
                serv.start();
        } catch (Exception e) {
                clean();
                throw e;
        }
}

@Override
public void tearDown() {
        clean();
}

private void clean() {
```

```
            if (serv != null) {
                serv.stop();
            }
            if (selenium != null) {
                selenium.stop();
            }
    }

}
```

8.8.2.2 To Add a New Test Suite in Alltest Class

You can add a new test suite to the AllTest class as follows.

Example

```
package com.photon.Phresco.testcases;

import org.junit.runner.RunWith;
import org.junit.runners.Suite;
import org.junit.runners.Suite.SuiteClasses;

@RunWith(Suite.class)
@SuiteClasses({Suite1.class,Suite2.class})
public class AllTest {
}
```

8.8.2.3 To Add New Test Case in Test Suite

You can add a new test case to the Test suite using the following method. Here is an example for creating a new test case in the name “CamerasAddcart”

```
package com.photon.Phresco.testcases;

import org.junit.runner.RunWith;
import org.junit.runners.Suite;
import org.junit.runners.Suite.SuiteClasses;

@RunWith(Suite.class)
@SuiteClasses({ WelcomePage.class,TeleVisionAddcart.class,ComputersAddcart.class,
                MobilePhonesAddcart.class,AudioDevicesAddcart.class,
                CamerasAddcart.class
                 })
public class Suite1 {

}
```

8.8.2.4 Report generated after execution

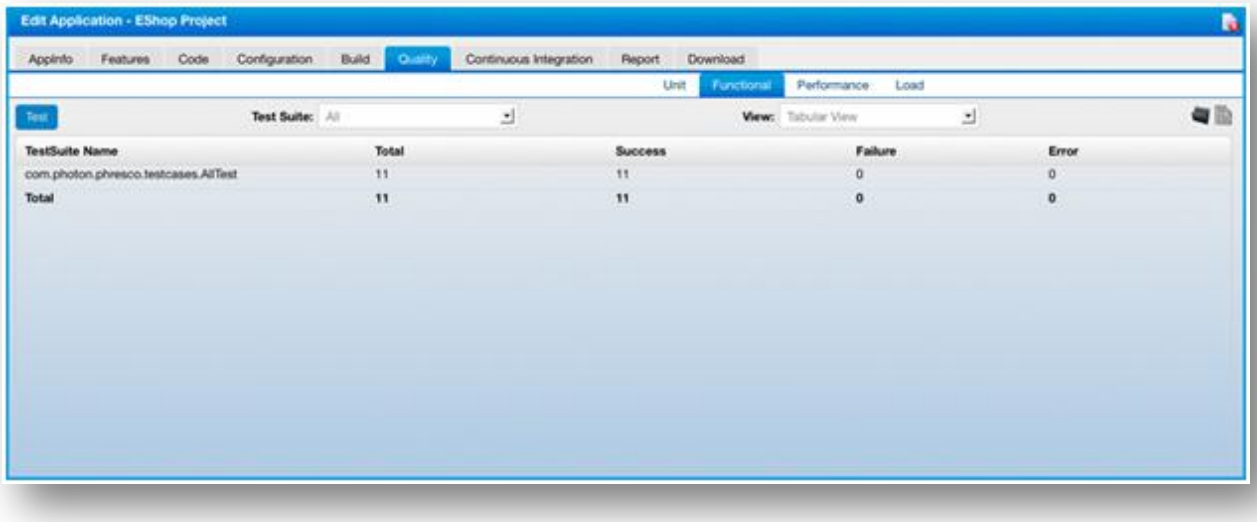


Figure 8-68: jQuery Functional test report for all test cases in graphical view

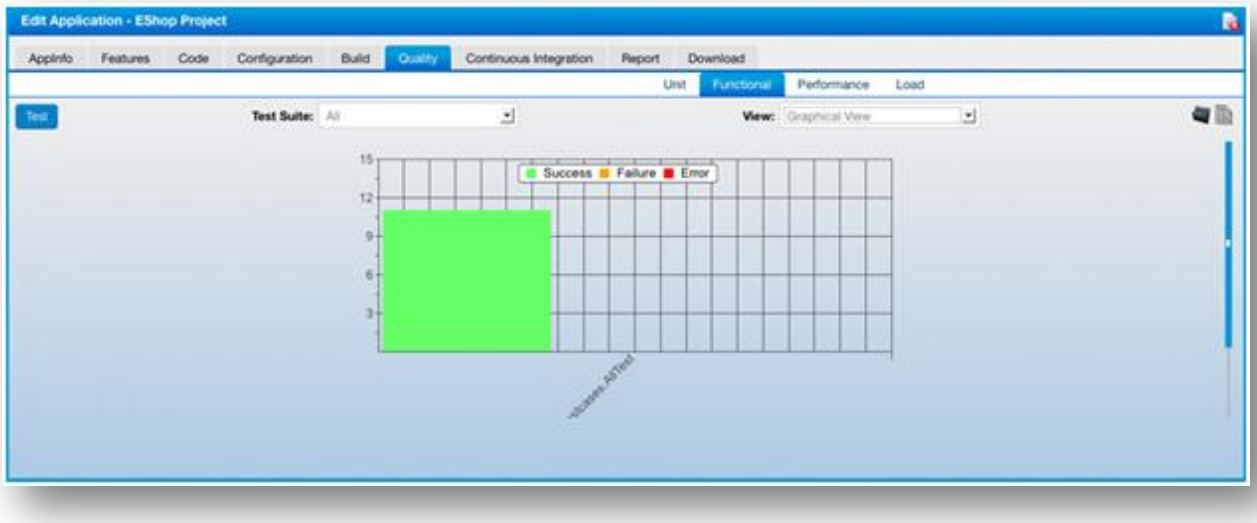


Figure 8-69: jQuery functional test report for all test cases in tabular view

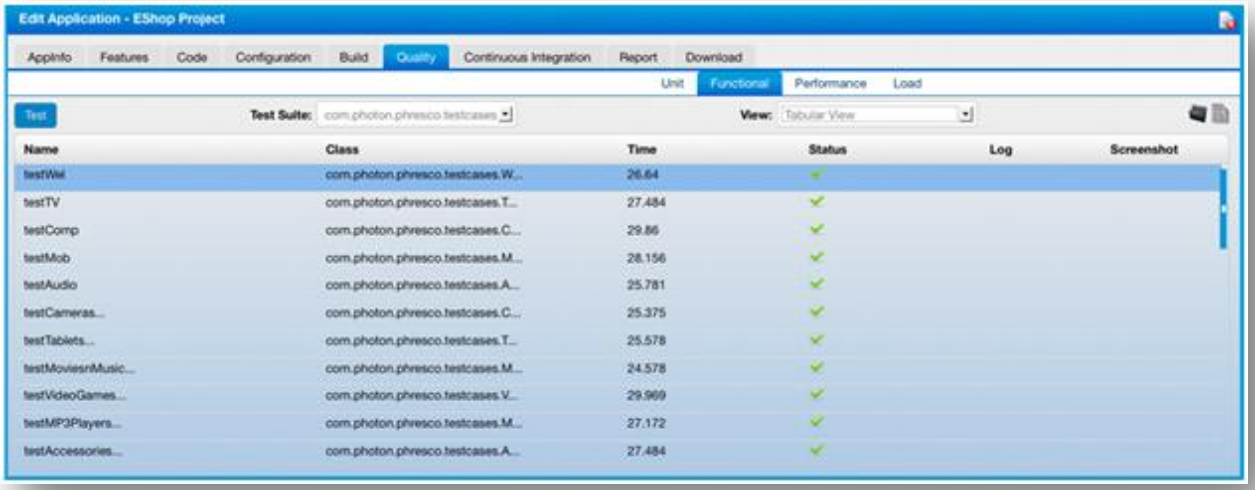


Figure 8-70: jQuery functional test case report for single test case in tabular view

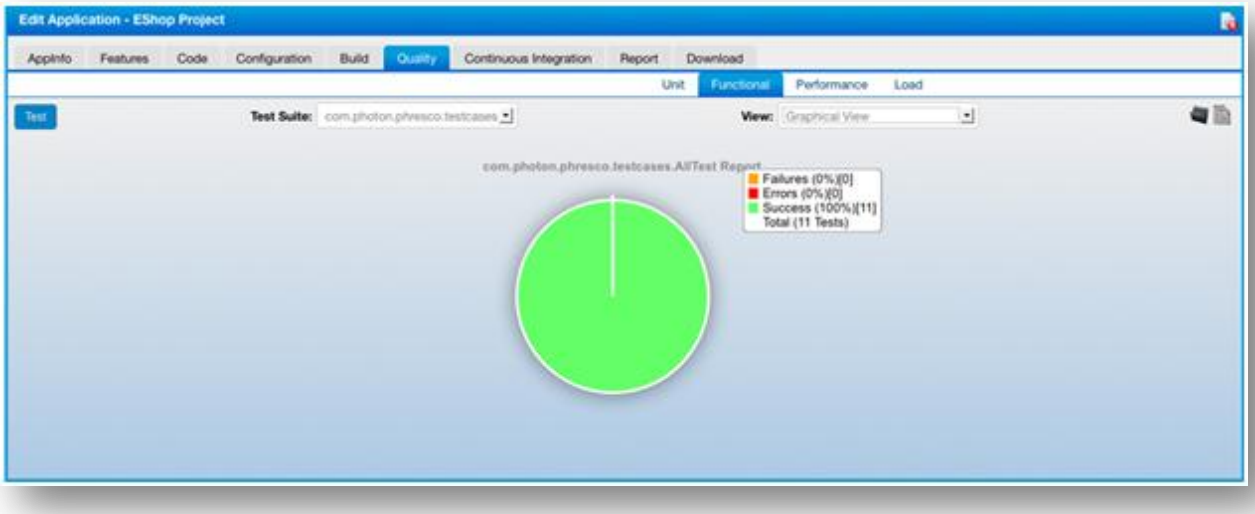


Figure 8-71: jQuery functional test case report for single test case in graphical view

8.9 Performance Testing

Phresco enables the users to test the performance of their project in an elementary way. It integrates JMeter as a tool and as a result generates a report that can be viewed by Phresco users. The Apache JMeter is a tool which is mainly used for testing both performance and load testing. It is fully comprised of java application and provides the result in tabular and graphical form. The inputs are entered in Phresco user interface which in turn is assigned as an input to the JMeter. The parameters for performance testing is calculated and given as a final report in Phresco framework.

Performance testing is testing the performance of a Server when certain number of users hits the URL at specific period of time. This helps in calculating the average response time, throughput, minimum responsive time and maximum responsive time.

Average response time:

Average response time is the Average time calculated when the server responds for any given input. This is calculated by using JMeter.

Throughput:

Throughput is the amount of capacity that a server can handle. In other words throughput is the amount of work that a server does in a specific time.

Minimum and Maximum Response time:

Minimum Response time is the minimum time calculated when the server responds for any given input.

Maximum Response time is the maximum time calculated when the server responds for any given input.

Performance testing can be done against server, web service and database.

Performance test against server by using Phresco Framework:

To run a performance test against a server, the server has to be selected along with the environment and the Test Result Name should be filled. To test the performance of any server, access to the server should be available. Add header tab is used to enter the header parameter which authenticates the server for testing. Few mandatory fields in Context URLs like Name, context, Type, Encoding and fields like No. of Users, Ramp-Up Period, and Loop count in Thread Group should be filled before Performance testing is done. GET or POST type can be selected in the type field. GET

is the method used when user needs to get the content from the server. POST is the method used when input is fetched. One or more number of servers can be tested by clicking the add button while the minus button is used to deselect the URLs.

Performance Test against Web Service using Phresco Framework:

To run a performance test against a Web service, the Web services has to be selected along with the environment and the Test Result Name should be filled. To test the performance of any WebService, access to the WebService should be available. Add header tab is used to enter the header parameter which authenticates the WebService for testing. Few mandatory fields in Context URLs like Name, context, Type, Encoding and fields like No. of Users, Ramp-Up Period, and Loop count in Thread Group should be filled before Performance testing is done. GET or POST type can be selected in the type field. GET is the method used when user needs to get the content from the server. POST is the method used when input is fetched. One or more number of web services/servers can be tested by clicking the add button while the minus button is used to deselect the URLs.

Performance Test against Database using Phresco Framework:

To run a performance test against a database, the database has to be selected along with the environment and the Test Result Name should be filled. To test the performance of any database, access to the database should be available. Add header tab is used to enter the header parameter which authenticates the database for testing. Few mandatory fields in Database Query like Name, Query Type, Query and fields like No. of Users, Ramp-Up Period, and Loop count in Thread Group should be filled before Performance testing is done. One or more number of database can be tested by clicking the add button while the minus button is used to deselect the database.

Report generated after performance testing

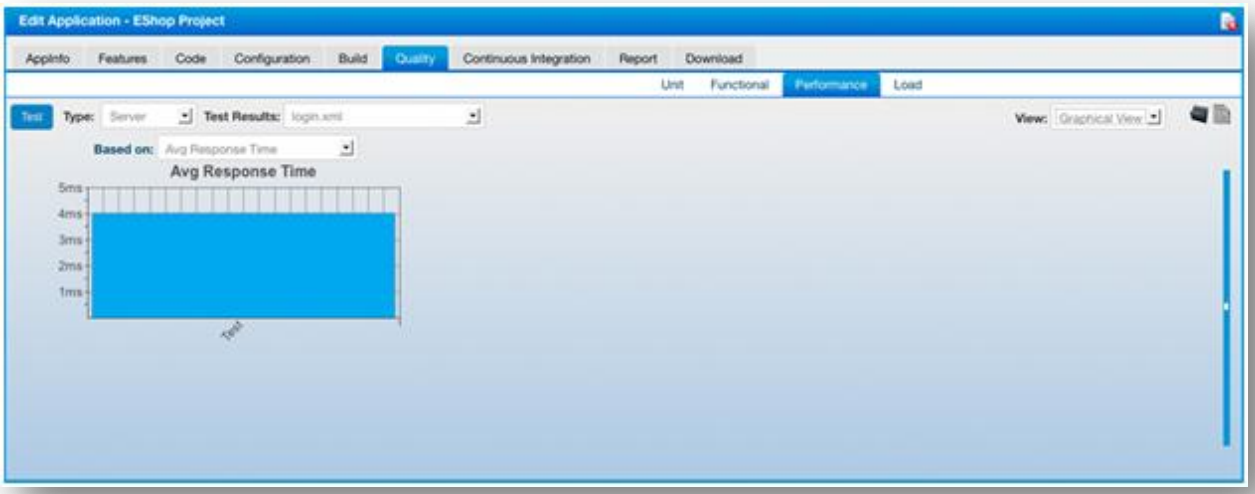


Figure 8-72: Performance test report graphical view

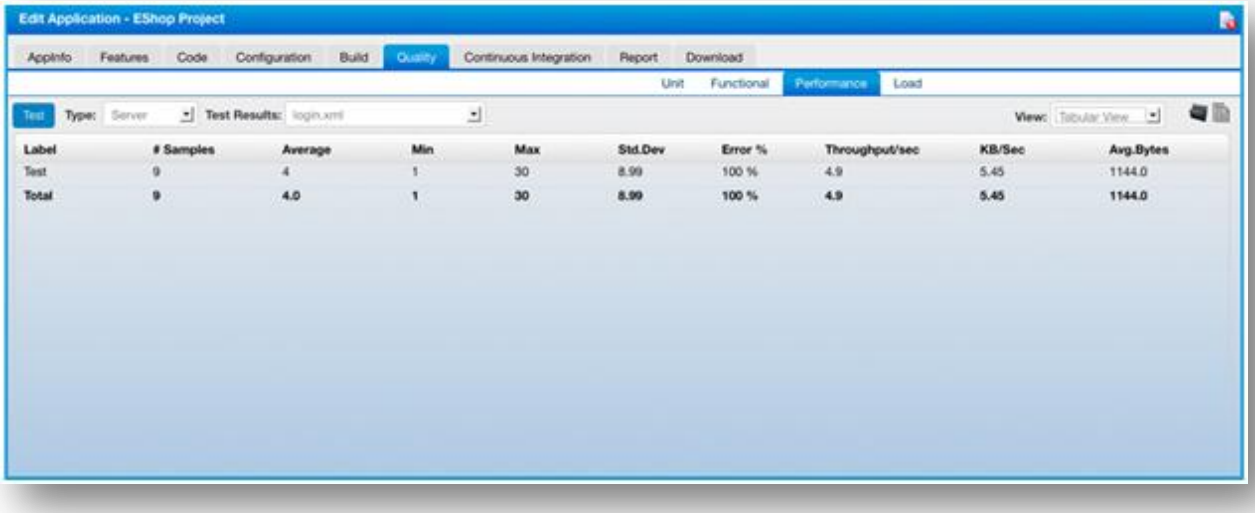


Figure 8-73: Performance test report tabular view

8.10 Load test

Load testing generally refers to the practice of modeling the expected usage of a server by simulating multiple users to access the same project concurrently. Project should be designed in such a way that when a maximum load is reached, a project should not crash and instead it should show a message saying the load has exceeded. Load and performance testing is usually conducted in a test environment identical to the production environment before a project is permitted for real time usage.

Phresco uses JMeter for Load testing. The result is given through static analysis and test cases. The test cases will point out the error and this will be very useful for the testing team.

Elapsed Time:

The amount of time that has passed since a particular process started especially compared with the amount of time that was calculated for it in a plan.

Load test against server using Phresco framework

To run a load test against a server, the server has to be selected along with the environment and the Test Result Name should be filled. Few mandatory like No. of Users, Ramp-Up Period, and Loop count in Thread Group should be filled before load testing is done. By clicking the test button, JMeter carries the inputs and provides the end report in both tabular and graphical form. The elapsed time and the status can be viewed from Phresco user interface.

Load test against Web service using Phresco framework

To run a load test against a Web service, the Web service has to be selected along with the environment and the Test Result Name should be filled. Few mandatory like No. of Users, Ramp-Up Period, and Loop count in Thread Group should be filled before load testing is done. By clicking the test button, JMeter carries the inputs and provides the end report in both tabular and graphical form. The elapsed time and the status can be viewed from Phresco user interface.

8.10.1 Report Generated After Load Testing

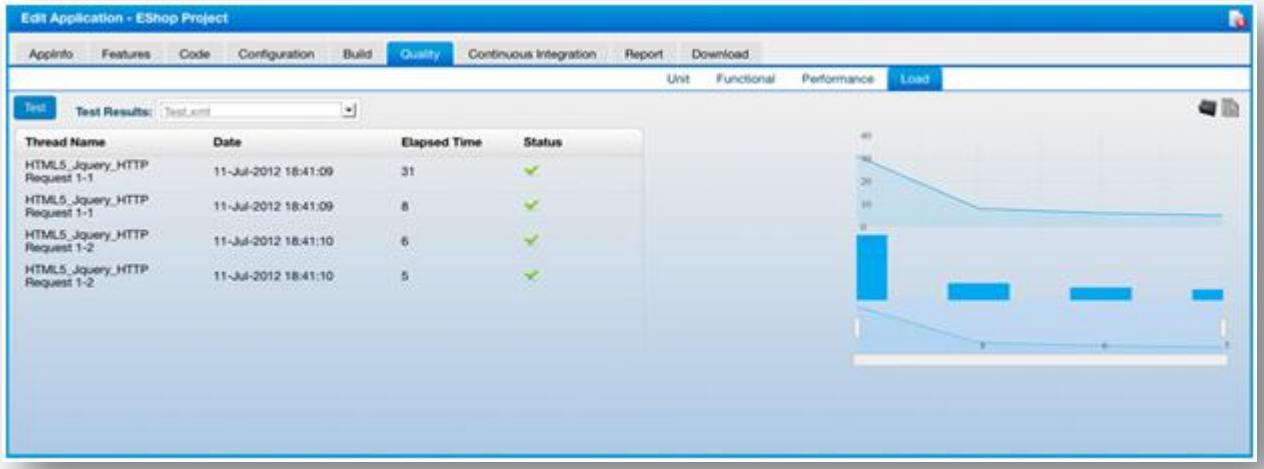


Figure 8-74: Load test report

9 Continuous Integration

Continuous Integration is the process where the builds are automatically generated and deployed by scheduling it to a particular time. Also testing process can be automated using continuous integration. This process reduces the integration problem and it gives the development team to produce a full quality project. This deducts the error that occurs during build, deployment and testing. It automatically gives the report through provided mail id. Continuous Integration uses the tool called Jenkins which is integrated within Phresco Framework. Jenkins server starts separately in the local host and the port number is 3579. When the codes are changed by the developers in the version controlling system, builds are automatically developed and the report is generated inside the Jenkins which also intimates about the build success or failure through email. If there is no change in the code, build will not start in the server.

Phresco also allow users to create multiple jobs by scheduling different project in different timings. This multiple job in continuous integration can be achieved by creating jobs after the completion of one job.

9.1 Performance of Continuous Integration

Setup button present in Phresco user interface loads the Jenkins and Jenkins can be started when the start button is clicked.

Note:

Once the Jenkins is started for one technology, it is not necessary to start the Jenkins again for other technology. When the stop button is clicked, the Jenkins stops running and the starting procedure should be done from setup and start.

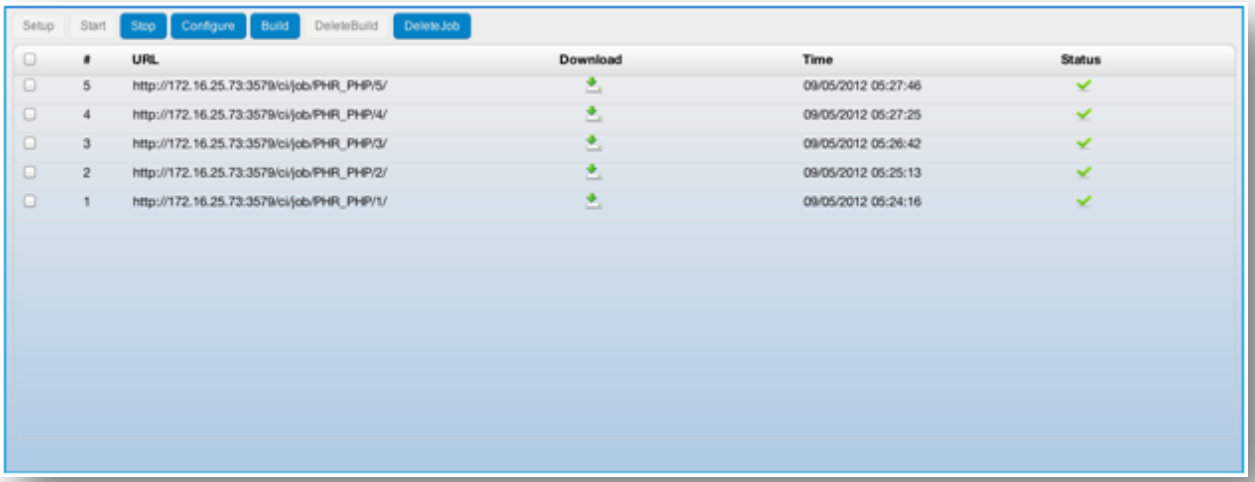


Figure 9-1: Continuous Integration auto generated builds

Fields in screenshot

Name

It denotes the name which identifies the Continuous Integration file and it is automatically filled with the code PHR_, followed by a project name.

SVN URL

It denotes the URL from which the projects are checked out can be copied here. Continuous Integration process is carried on to the copied link of a project.

Username

It denotes the SCM username.

Password

It denotes the SCM password.

Sender Email

It denotes the email id to which status of the build can be sent.

Sender Password

It denotes the password of the email id to which status of the build can be sent.

Recipients Email

The checkboxes in this field denotes, if the build failure or build success report should be sent through Email. When success checkbox sends the email when the build succeeds and when failure checkbox sends the email when the build fails.

Build triggers

- Build periodically:
  The option build periodically is to build the project in specific time.

- Poll scm:
  This option builds the project only when there is a commit in the source code.

Schedule

The date and time intervals can be selected which builds a project automatically in the given period of time and sends the report to the mail.

Cron expressions

A cron expression is a string consisting of few sub expressions that describe individual details of the schedule.

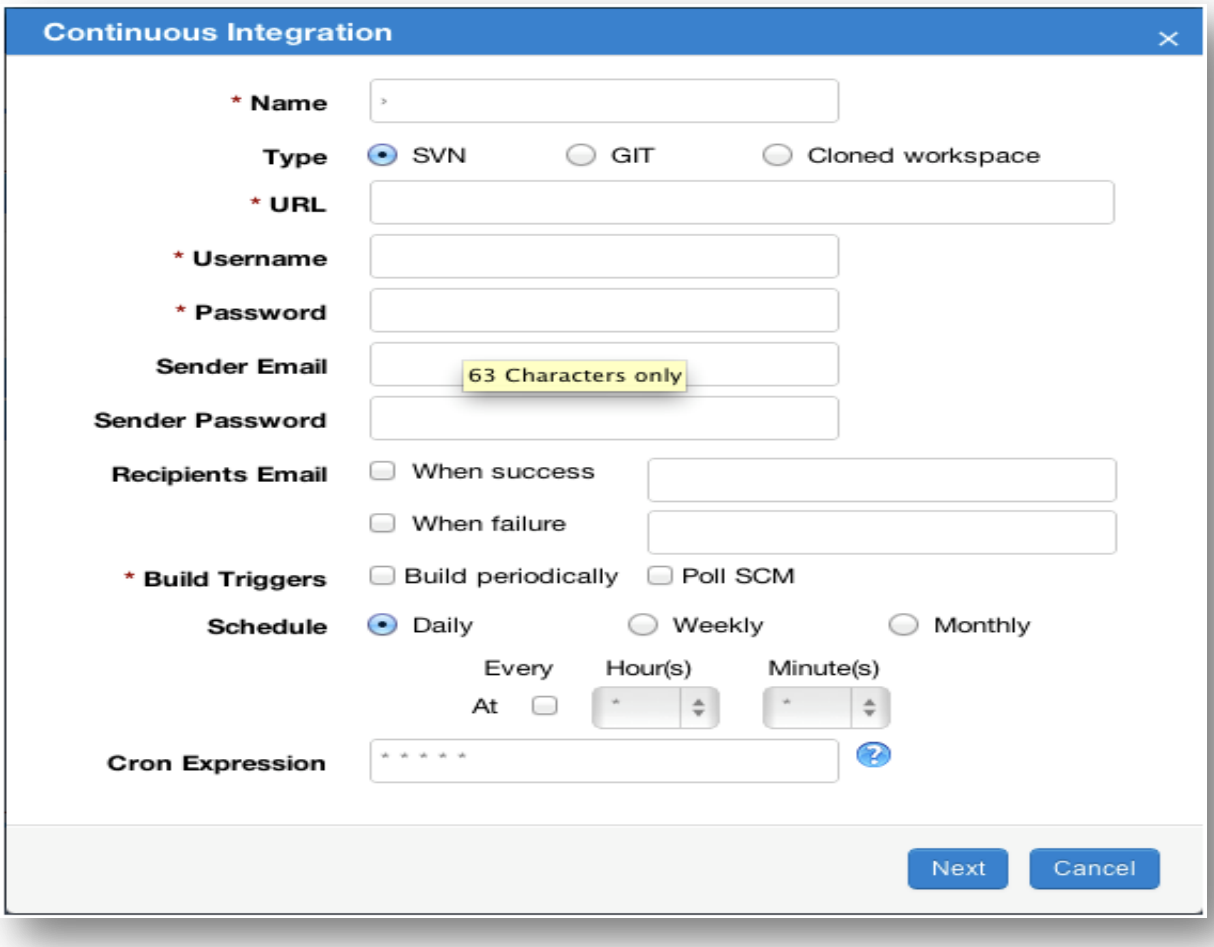


Figure 9-2: Continuous Integration

9.2 Continuous integration with Build configurations

Fields in the screenshot

Operation

Build, deploy and test operation can be selected from the dropdown field to perform the particular operation

Environment

Created environments can be selected from the drop down box. If the environment is not created Production environment is selected as default

Downstream Project

It creates link within the multiple jobs created so that when one operation of the job is completed, the next operation starts automatically depending on the order selected

Clone the workspace

This can be enabled to use the same source code across all the operations and can be selected as 'no' to check out the source code for each and every operation

Show settings

The global configurations for server, database, web service and email can be selected using the show settings option.

Skip unit tests

Phresco enables skip unit tests where the unit tests will be skipped on building a project. This consumes less time on building a project.

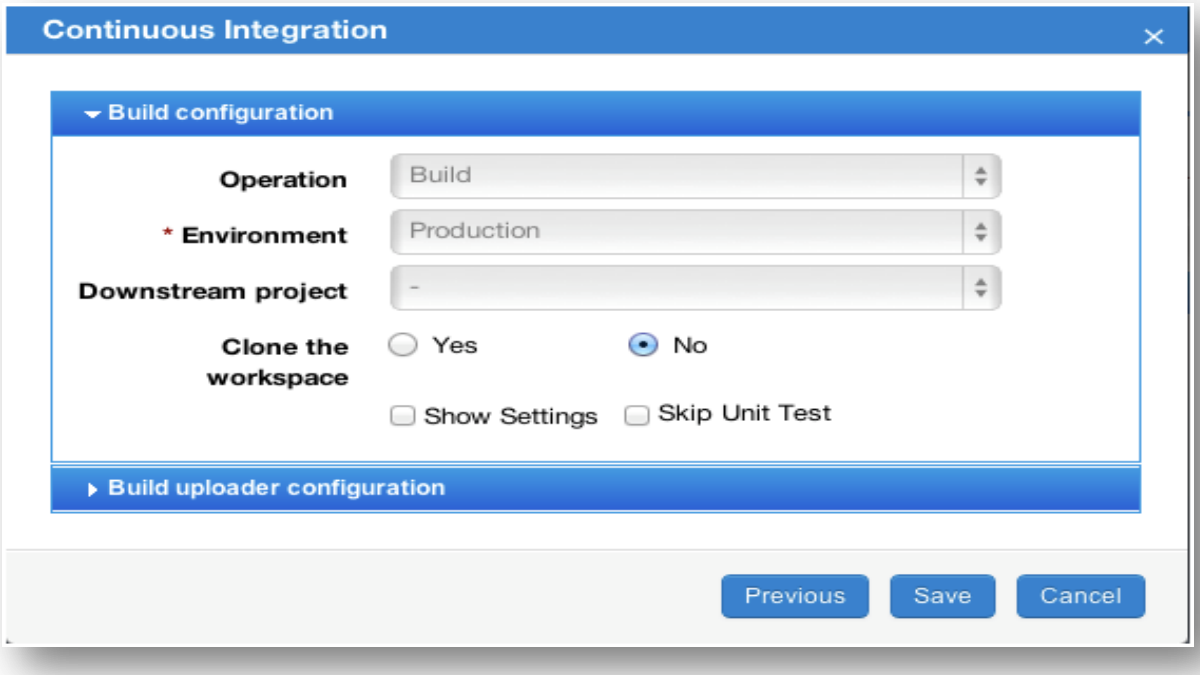


Figure 9-3: Build configuration fields

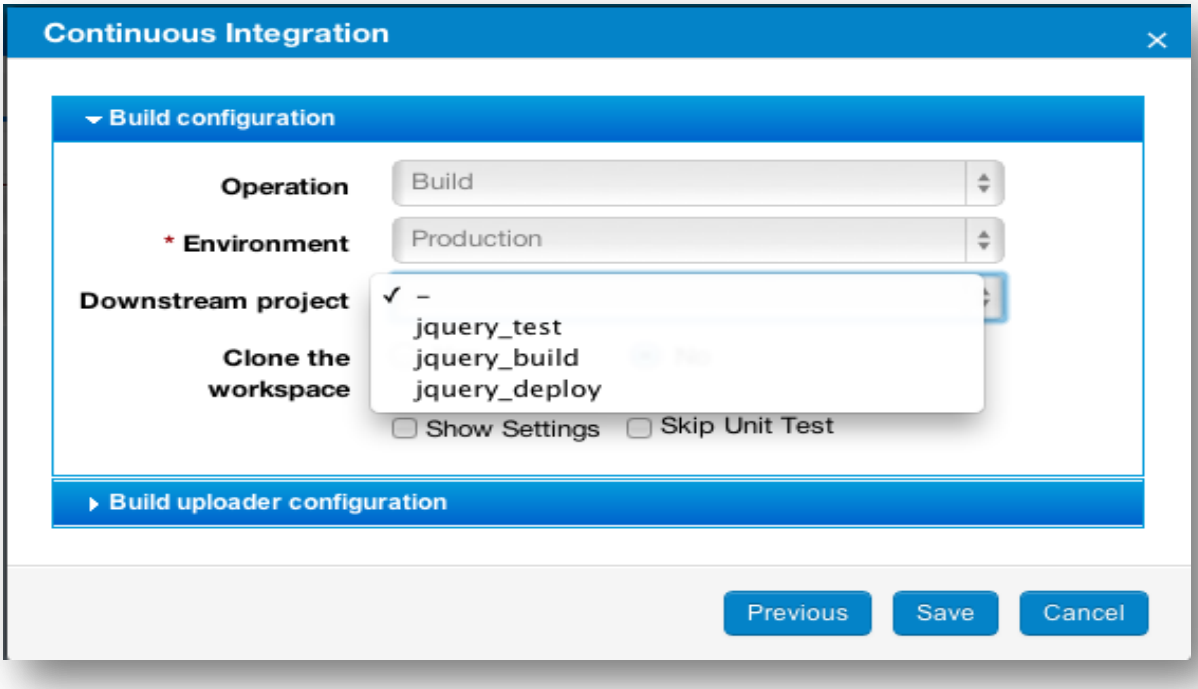


Figure 9-4: the created project in downstream

9.3 Continuous integration with collabnet plugin

Continuous integration process builds the project automatically and also uploads the build in insight using the collabnet plugin.

Field in the screenshot

Enable collabnet file release

When the build is to be uploaded in insight automatically, collabnet file releases can be selected as yes otherwise no option should be selected.

URL

It denotes the collabnet WebService URL to which the build should be uploaded.

Username

It denotes the user name of the insight credentials for accessing the insight.

Password

It denotes the password of the insight credentials.

Project

It denotes project name in the collabnet to which the project is to be uploaded.

Package

It denotes the Package name of a project where the build is to be uploaded.

Release

It denotes the folder releases name of a project where the build should is to be uploaded.

Overwrite existing files

When the existing build in insight is to be overwritten, yes option can be selected otherwise no option should be selected.

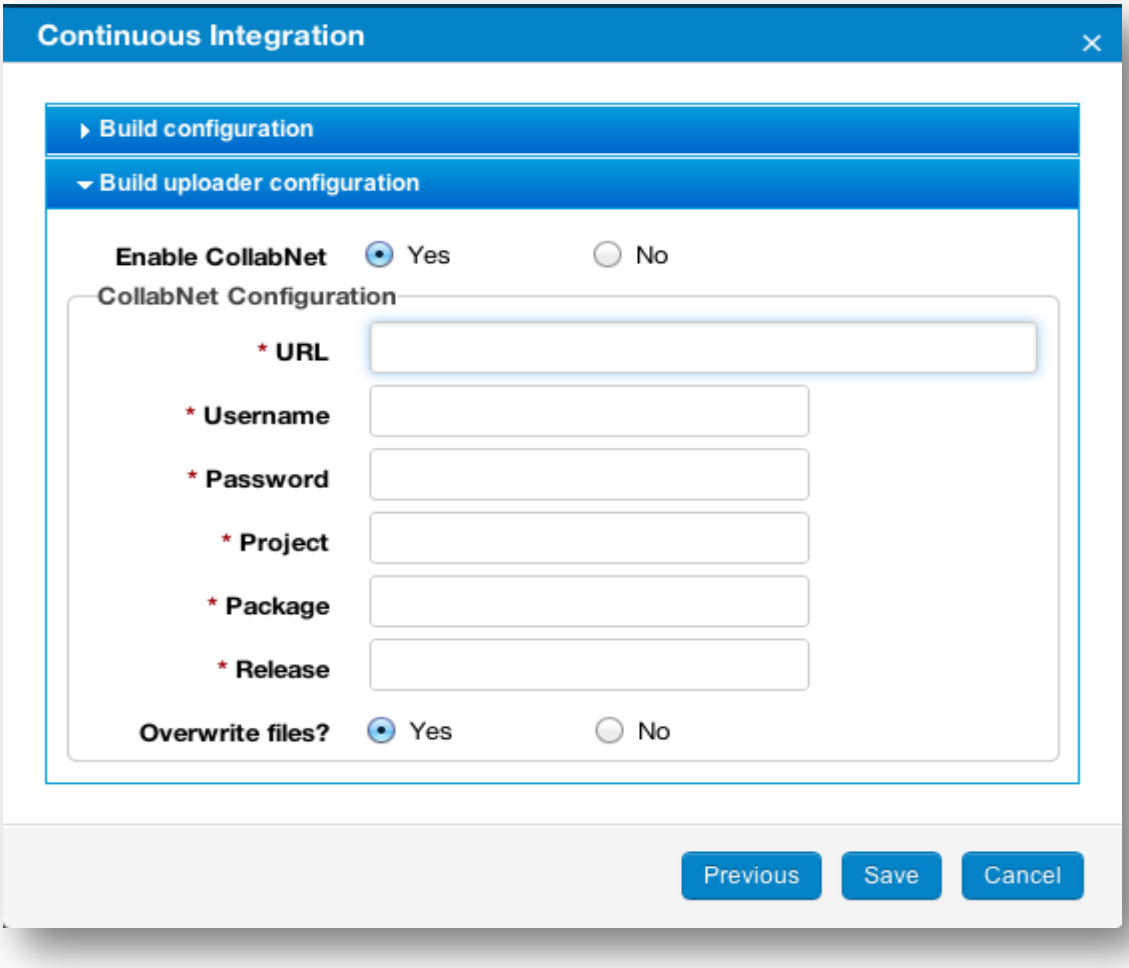


Figure 9-5: build uploader configuration fields