



# CS427 FINAL REPORT

Team: FailedTest

Hiren Bhavsar, Asad Butt, Felicia Chandra, Kai Huang, Yu-Wen  
(Sundie) Jong, Yaning Liang, Souhayl Maronesy, Hao Zhang

# Table of Contents

1. Project Description .....	2
2. Architecture.....	2
2.1. Original Plugin .....	2
2.2. Changes .....	2
3. Design .....	3
3.1. Custom Data Structure .....	3
3.1.1. Pair.....	3
3.1.2. BuildInfo .....	3
3.1.3. TestInfo .....	3
3.2. TestBuddyHelper .....	3
3.3. RegressionReportNotifier .....	3
3.4. TestBuddyAction.....	4
3.5. Jelly Pages .....	4
3.6. CSS and Javascript Files .....	4
3.7. Tests.....	5
4. Usage .....	3
4.1. Email Extension .....	5
4.2. Test Buddy Interface.....	6
4.2.1. Accessing Test Buddy .....	6
4.2.2. Comparing the Last Two Builds .....	7
4.2.3. Viewing Builds .....	8
4.2.4. Viewing Tests .....	10
4.2.5. Comparing Tests .....	10
4.2.6. Generating Charts .....	11
5. Conclusion .....	15

# 1. Project Description

Our modifications to the Regression Report plugin are meant to make the user experience much richer when navigating through builds and tests. One of the challenges that arises as the number of builds and tests increase is that the developer begins having a hard time navigating through the project. For example, if the developer wants to refer back to a build that was made at the beginning of the project and compare it to a build created in the middle of the project, using the original plugin he or she would have to manually navigate through the builds and compare them; however, with the functionalities we added to the plugin, this comparison is a few clicks away. A similar functionality was added for comparing the results of a specific test across all builds, in addition to seeing the failing or passing rate of a particular test. We have also made it easier for the developer to stay up to date with the latest build status of the project through our extension of the email functionality. In addition to receiving emails about regressed tests, developers will also have the options to receive emails about progressed tests, failing new tests, passing new tests, and having the build log as an email attachment.

## 2. Architecture

### 2.1. Original Plugin

We used the Regression Report Plugin as our base. This plugin consists a core class (`RegressionReportNotifier`) as the back-end code to send an email regarding a regressed test, and a core jelly file (`config.jelly`) as the front-end interface to configure the email notification settings. It follows Jenkins convention, where the configuration values are stored in the project's `config.xml`. When a build is created, `RegressionReportNotifier` retrieves these settings and starts the email process.

### 2.2. Changes

We extended the email functionality to give developers more options to report tests upon their choice and added a user interface to view and compare builds and tests. These changes involve some updates to the original `RegressionReportNotifier` class, three new custom data structures: `pair`, `BuildInfo`, `TestInfo`, two additional core classes: `TestBuddyHelper`, `TestBuddyAction`, several jelly pages, and a few CSS and JavaScript files for front-end display. The details of these changes will be explained in the Design section below.

## 3. Design

### 3.1. Custom Data Structure

#### 3.1.1. Pair

`Pair.java` contains a custom class to pair two different objects together.

#### 3.1.2. BuildInfo

`BuildInfo` contains information about a build, for example build number, timestamp, status, a list of authors who contribute to the code changes, number of passed tests, and test passing rate in the build. It is created so that the jelly pages can easily access and display this information on the webpage.

#### 3.1.3. TestInfo

`TestInfo` contains information about a test, for example full name, short name, class name, package name, status, passing count, failing count, and skipped count in all builds. Similar to `BuildInfo`, this class is also created so that the jelly pages can access this information easily.

### 3.2. TestBuddyHelper

`TestBuddyHelper` is where all the main logic for this plugin resides. This class gets the necessary data from Jenkins and process it to generate information required by the plugin. Some notable functions in this class are `getAllCaseResultsForBuild` and `getChangedTestsBetweenBuilds`. `getAllCaseResultsForBuild` returns all tests for a build in the form of `CaseResult`, while `getChangedTestsBetweenBuilds` returns the tests which have different passing and failing status between two builds.

### 3.3. RegressionReportNotifier

`RegressionReportNotifier` is an existing class from the original plugin. Several new functions have been added to this class to extend the email functionality.

`attachLogFile` is created to attach the build log to the email notification. `perform` is also modified to account for three additional scenarios: progressed tests, passing new tests, and failing new tests.

### 3.4. TestBuddyAction

`TestBuddyAction` provides wrapper functions which convert the data returned by `TestBuddyHelper` into a form more suitable for the jelly pages. The return types of the functions inside this class are either simple objects like `String`, `BuildInfo`, `TestInfo`, or a collection of one of those types.

### 3.5. Jelly Pages

The code for Test Buddy front-end interface is located in the jelly files under `src/main/resources/jp/skypencil/jenkins/regression/TestBuddyAction`. When the user accesses the Test Buddy page, the request will be redirected to `index.jelly`. This file contains the navigation bar and the logic to include the main content jelly page based on the `displayType` parameter from the URL query string. The jelly pages are bound to `TestBuddyAction.java` so they can directly call any function in the java source using `it.functionName()`. If the function name starts `get` and does not have any parameters, we can omit the `get` prefix, change the next first character to be lowercase, and remove the parentheses. For example, for the function `getBuilds()`, we can call it in jelly using `it.builds`.

### 3.6. CSS and JavaScript Files

The CSS and JavaScript files are located under `src/main/webapp/css` and `src/main/webapp/js` respectively.

All styling for the Test Buddy jelly pages are located in `test-buddy.css`. The CSS classes are prefixed with `tb-` to ensure it does not have the same name as Jenkins CSS classes and thus interfering with Jenkins styling.

The JavaScript code for the Test Buddy jelly pages are located in `test-buddy.js`. The code mainly depends on jQuery. However, because Jenkins uses `prototype.js`, we have to set jQuery to use the no-conflict option.

```
jQuery.noConflict();
```

With this option, we are no longer able to use the dollar sign and need to spell out the word `jQuery` to use its functionalities.

To call a java function from JavaScript, we use the proxy function from jQuery:

```
remoteAction.javaFunctionName(parameter, jQuery.proxy(function(t)
{
    var results = t.responseObject();
    // Do something else
}, this));
```

`remoteAction` is bound to `TestBuddyAction.java` so we can call any function from it as long as the java source annotates them with `@JavaScriptMethod`.

We also use the Google Chart JavaScript library to create the graphs for one of our features.

### 3.7. Tests

The unit tests for `TestBuddyHelper` and `TestBuddyAction` are designed using `JenkinsRule`. The tests use a zip file containing a project with a build history which has been prepared in advance. To access this zip file, each test should be annotated with `@LocalData`.

The pre-made zip file is located under `src/test/resources/jp/skypencil/jenkins/regression` folder with a readme file which explains the different build results. In truth, `TestBuddyActionTest.zip` and `TestBuddyHelperTest.zip` have the same content. However, this has to be done because `@LocalData` can only find the zip file with the same test class name.

## 4. Usage

### 4.1. Email Extension

From the Jenkins homepage, select the project that needs the Email Extension Functionality. Click on **Configure** from the left sidebar to enter the configuration page. Under **Post-build Actions** section, select the option “*report regressed tests via E-mail*” from the **Add post-build action** drop-down menu. This option enables additional settings such as choosing to receive reports on regressed tests, progressed tests, failing new tests and/or passing new tests, and whether to attach logs to the email.

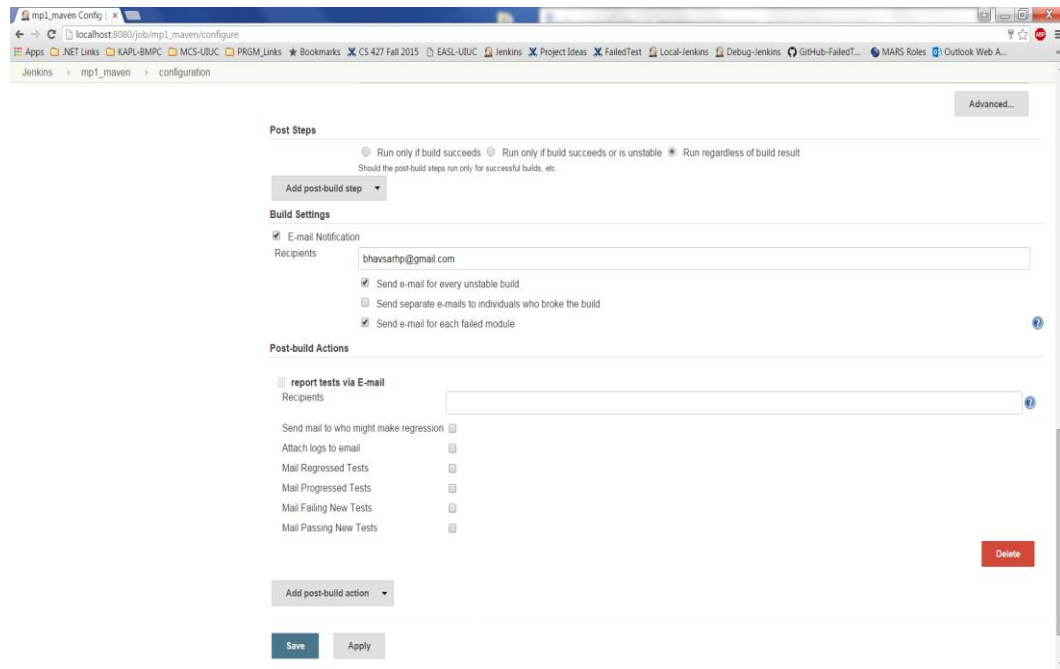


Figure 1: Email Extension

## 4.2. Test Buddy Interface

### 4.2.1. Accessing Test Buddy

From the project's main page, click on **Test Buddy** on the left sidebar. Test Buddy page is organized into five tabs, each tab providing different functionalities for test analysis, including comparing the last two builds, viewing all builds, viewing all tests, comparing any two tests, and generating charts.

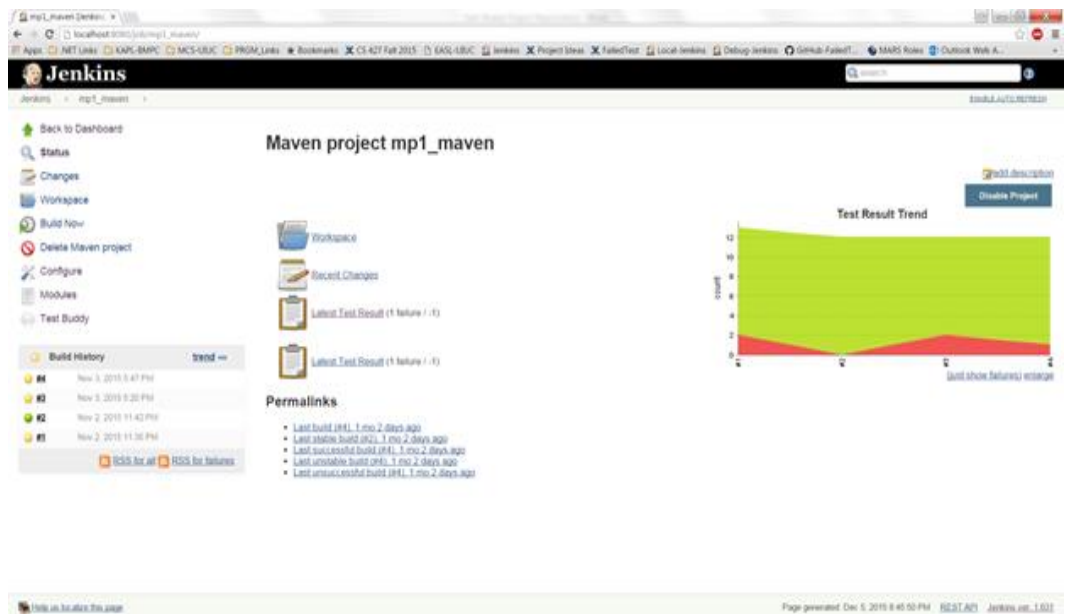


Figure 2: Accessing Test Buddy

#### 4.2.2. Comparing the Last Two Builds

From **Test Buddy** page, select the **Regression/Progression** tab. This feature displays the list of tests that had either a regression or a progression in the latest build. These are essentially all those tests that have a different result comparing to the previous build. The list of test results can be filtered using the checkboxes at the top.

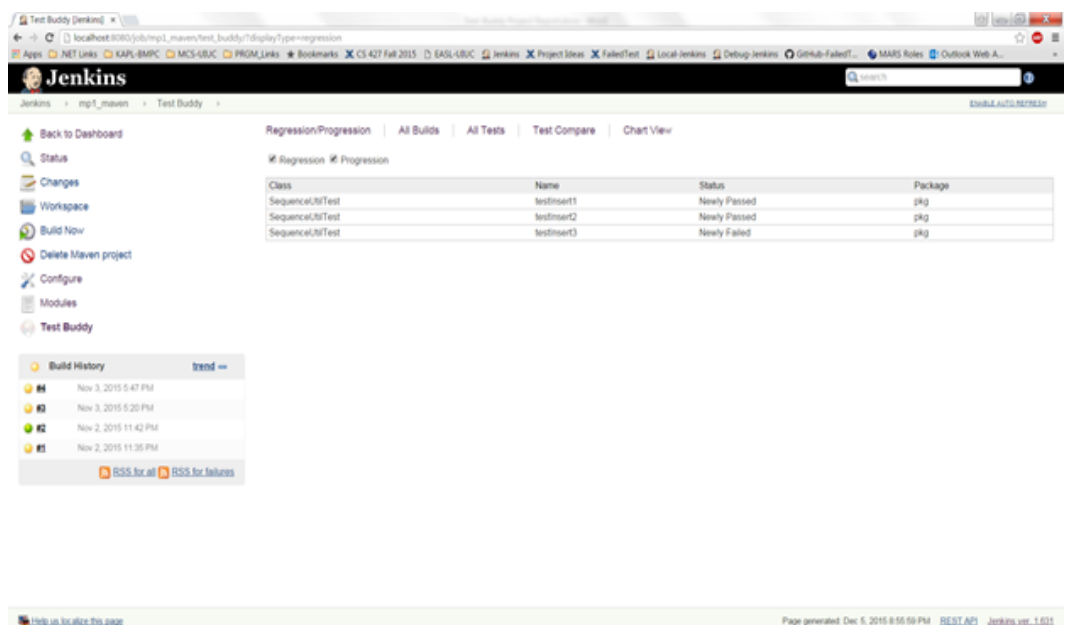


Figure 3: Comparing the Last Two Builds



### 4.2.3. Viewing Builds

From **Test Buddy** page, select the **All Builds** tab. This feature displays a list of all the builds of the current project in descending order, along with their build status, build time, commit authors, the passing rate and the number of passing tests.

The options at the top allows filtering the list of builds by its “Status” (ABORTED, FAILURE, NOT\_BUILD, SUCCESS, or UNSTABLE), “Authors” (all users that committed in the project), “Minimum Passing Rate” (a number between 0 and 1), and “Minimum Passed Tests”. **Clear** will remove all the filters and show the list of all builds again.

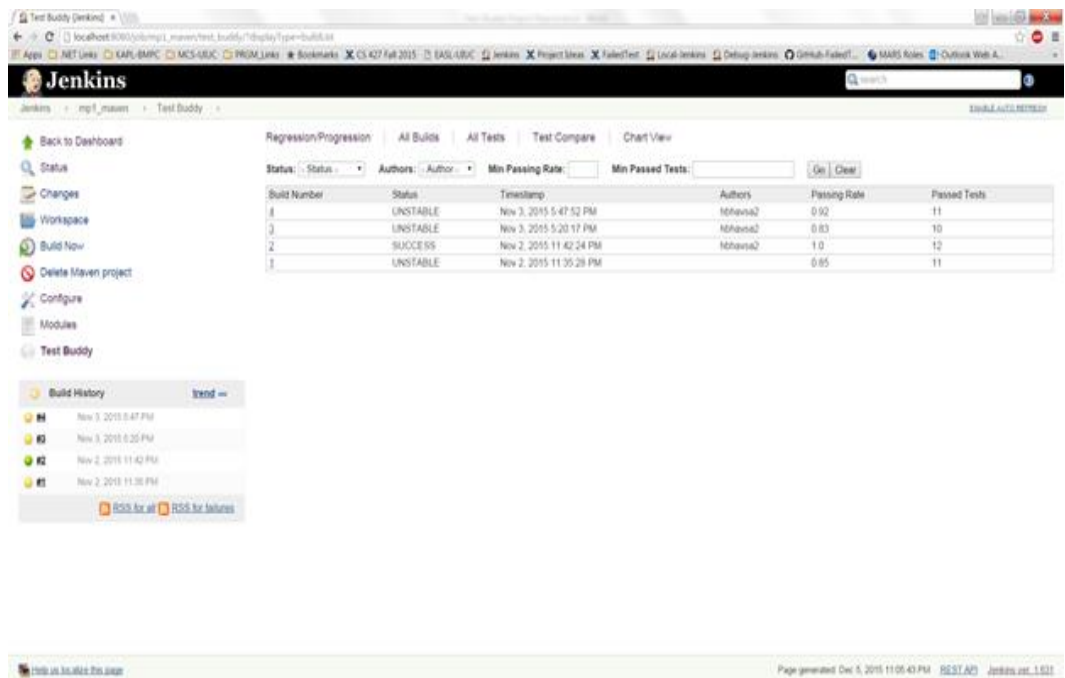


Figure 4: Viewing Builds

By clicking on a specific build number, a **Build Details** page will show a list of all the tests in that build, and their respective statuses.

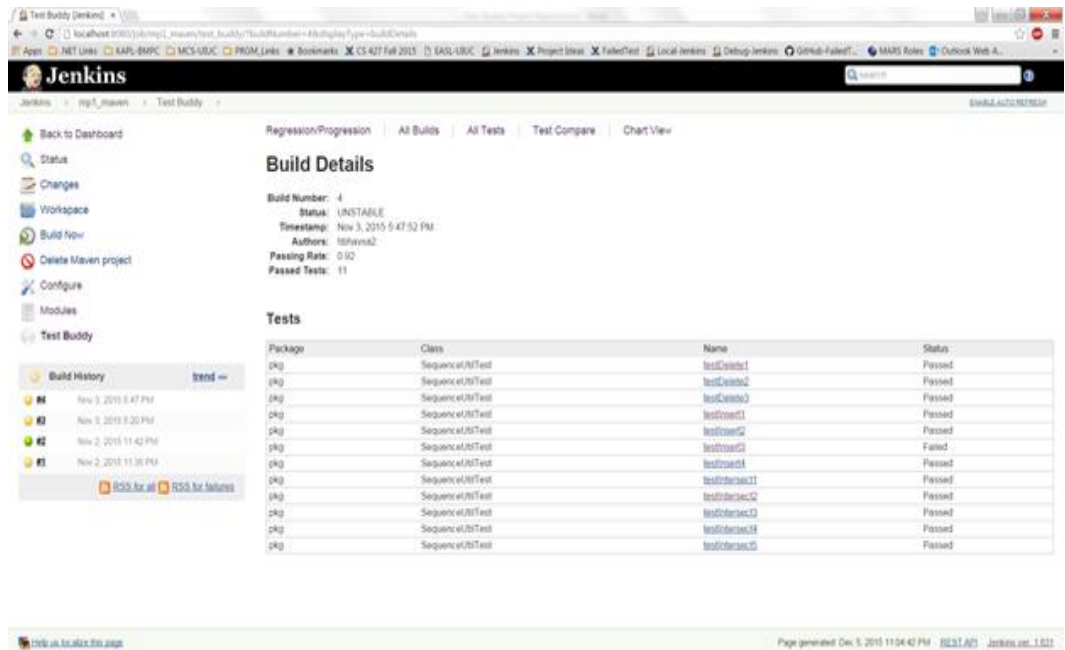


Figure 5: Build Details

By clicking on a specific test name, a **Test Details** page will display further information of the test, including its passing rate, it's passed, failed, and skipped count across all builds, and the respective status within each build.

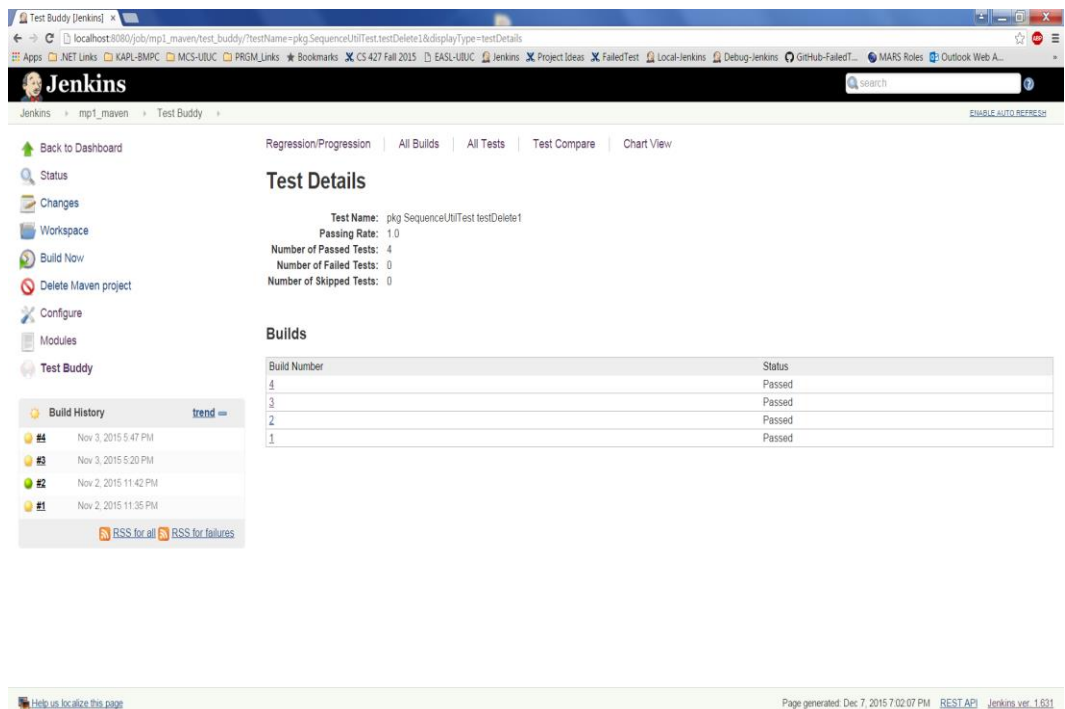
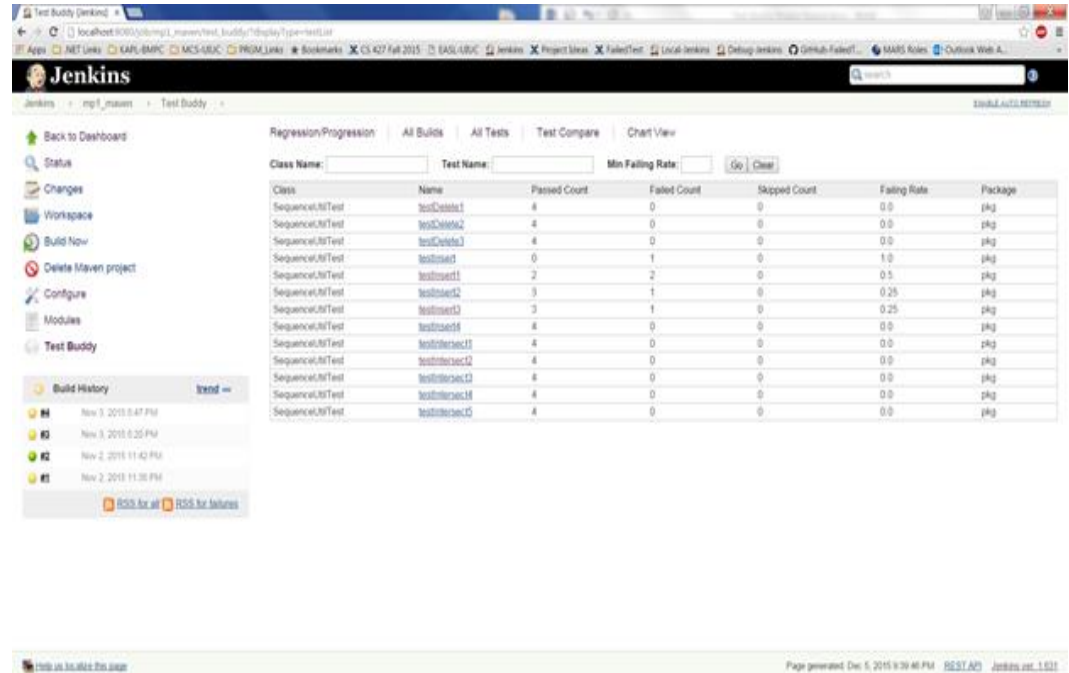


Figure 6: Test Details

## 4.2.4. Viewing Tests

From **Test Buddy** page, select the **All Tests** tab. This feature displays a list of all the tests of the current project, along with their passed, failed, skipped count.

The options at the top allows filtering the list of tests by “Class name”, “Test name” and “Minimum failing rate”.



Class	Name	Passed Count	Failed Count	Skipped Count	Failing Rate	Package
Sequence/NTest	testCreate1	4	0	0	0.0	pkg
Sequence/NTest	testCreate2	4	0	0	0.0	pkg
Sequence/NTest	testCreate3	4	0	0	0.0	pkg
Sequence/NTest	testCreate4	0	1	0	1.0	pkg
Sequence/NTest	testCreate5	2	2	0	0.5	pkg
Sequence/NTest	testCreate6	3	1	0	0.25	pkg
Sequence/NTest	testCreate7	3	1	0	0.25	pkg
Sequence/NTest	testCreate8	4	0	0	0.0	pkg
Sequence/NTest	testIntersect1	4	0	0	0.0	pkg
Sequence/NTest	testIntersect2	4	0	0	0.0	pkg
Sequence/NTest	testIntersect3	4	0	0	0.0	pkg
Sequence/NTest	testIntersect4	4	0	0	0.0	pkg
Sequence/NTest	testIntersect5	4	0	0	0.0	pkg

Figure 7: Viewing Tests

## 4.2.5. Comparing Tests

From **Test Buddy** page, select the **Test Compare** tab. This feature compares test results from any two builds of the current project. The drop-down menus at the top allows selecting any 2 build numbers. The table displays a list of all the tests that have a different status across these two builds.

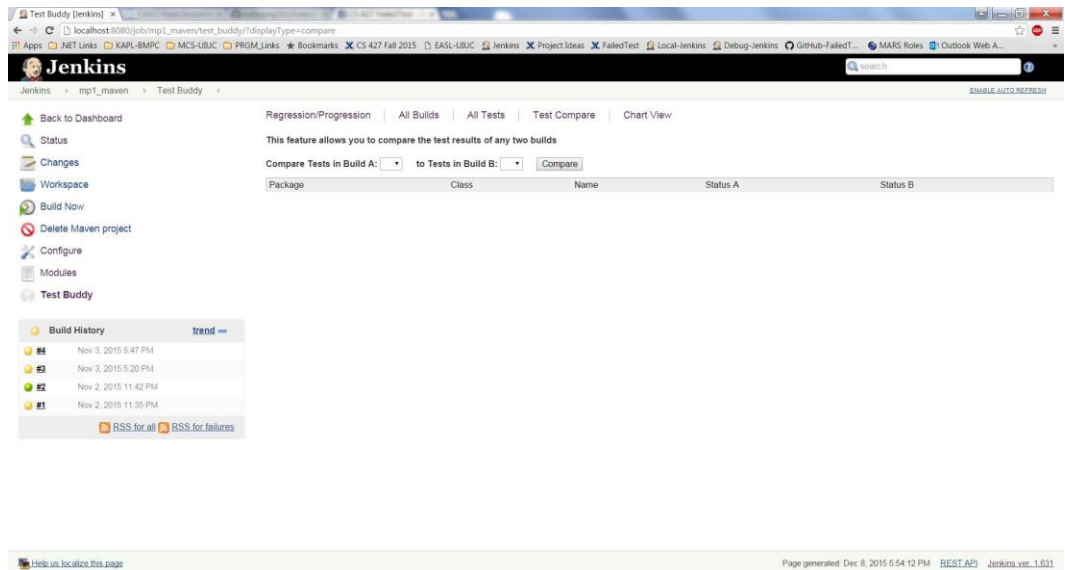


Figure 8: Comparing Tests – Select Builds

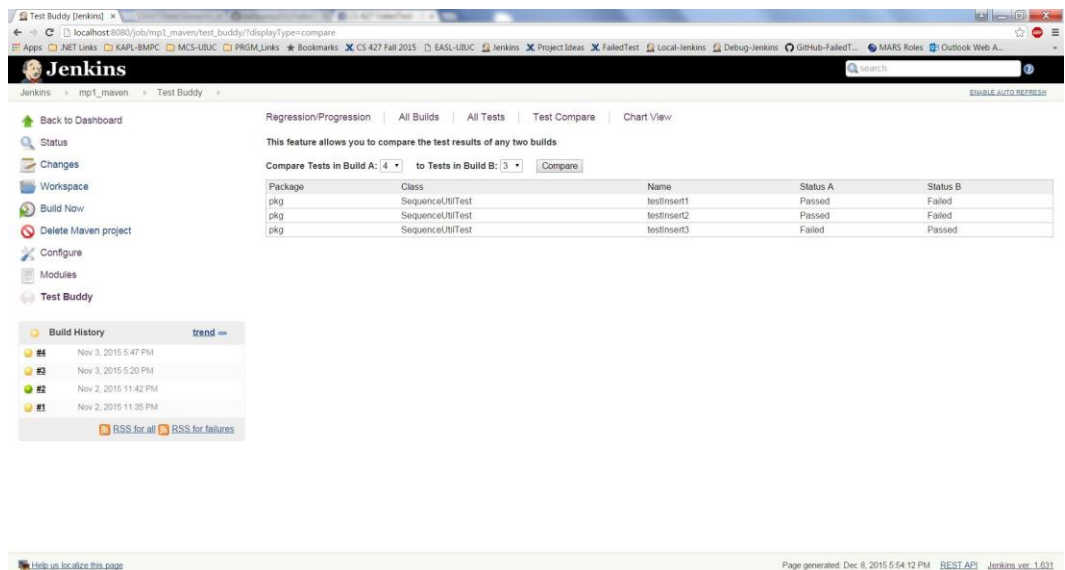


Figure 9: Comparing Test – Results

#### 4.2.6. Generating Charts

From **Test Buddy** page, select the **Chart View** tab. This feature allows searching a specific test by test name, class name, or package name. Once the search is complete, it allows to add up to 5 tests in the chart view list.

There are three types of charts supported: Bar Chart, Stacked Bar Chart, and Pie Chart. The chart representation allows an easy comparison of each test's passed, failed, and skipped count.

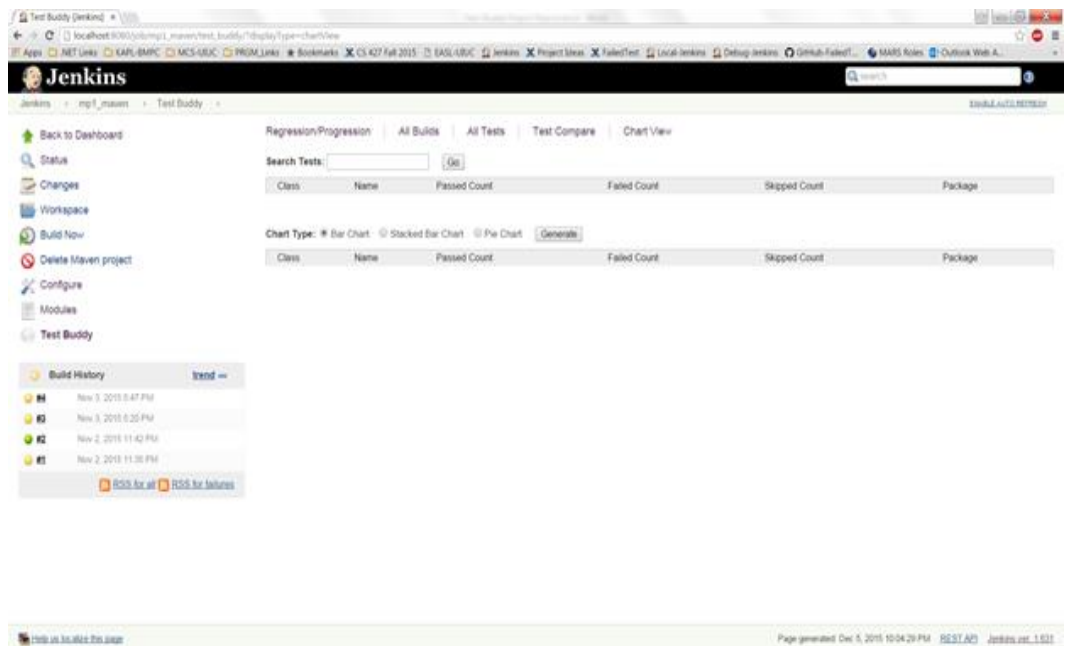


Figure 10: Generating Charts – Search Tests

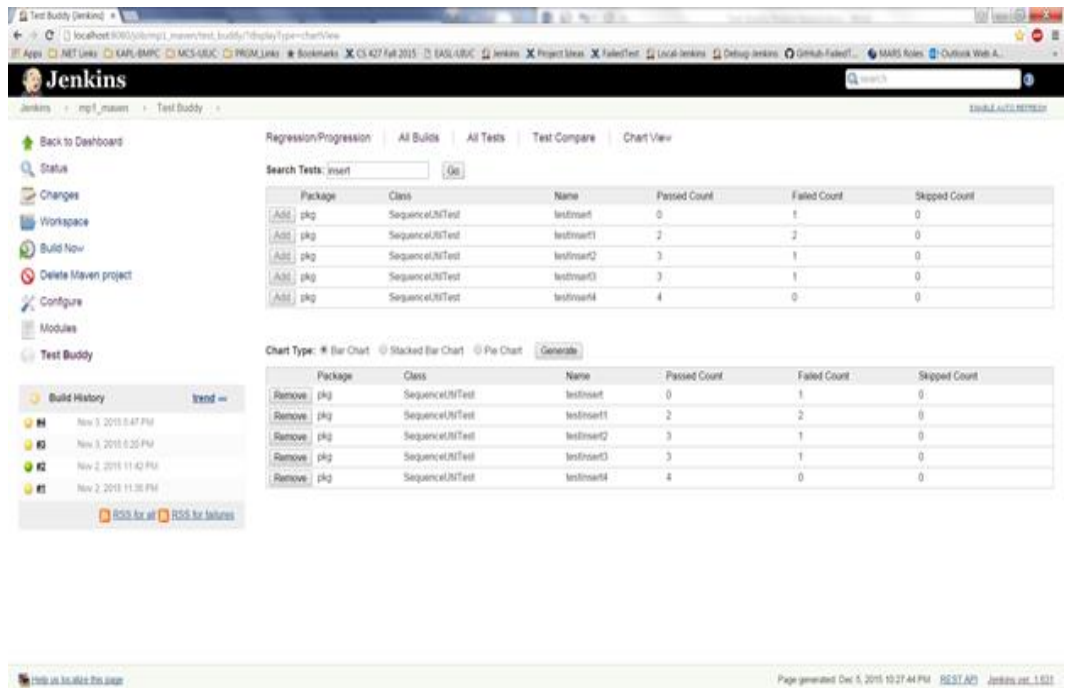


Figure 11: Generating Charts – Select Tests

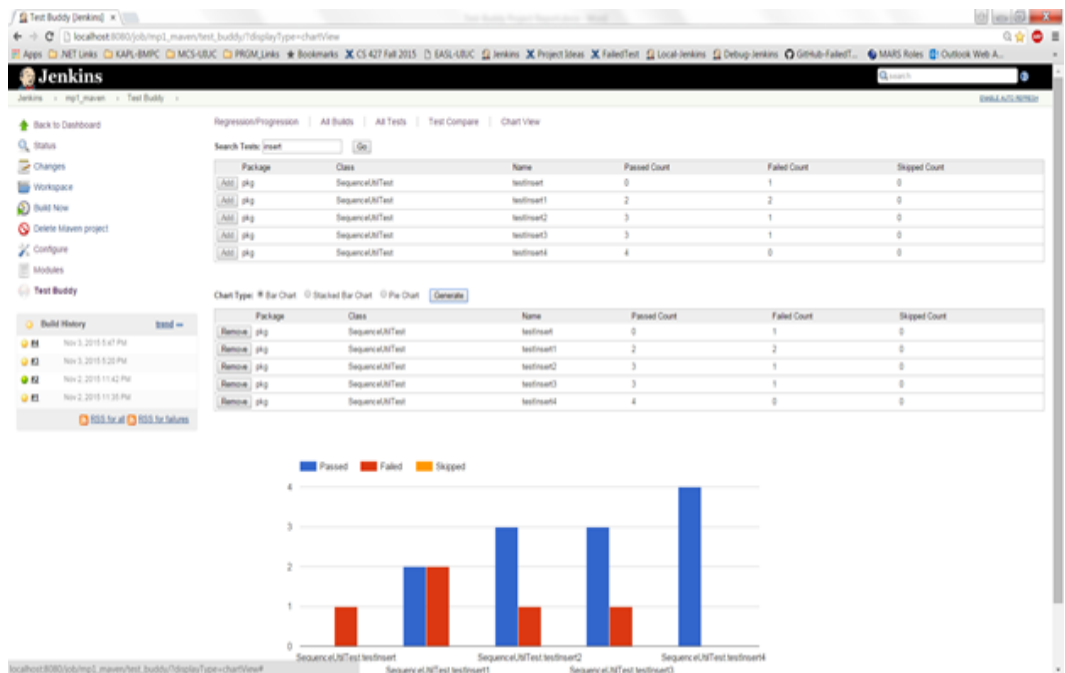


Figure 12: Generating Charts – Bar Chart

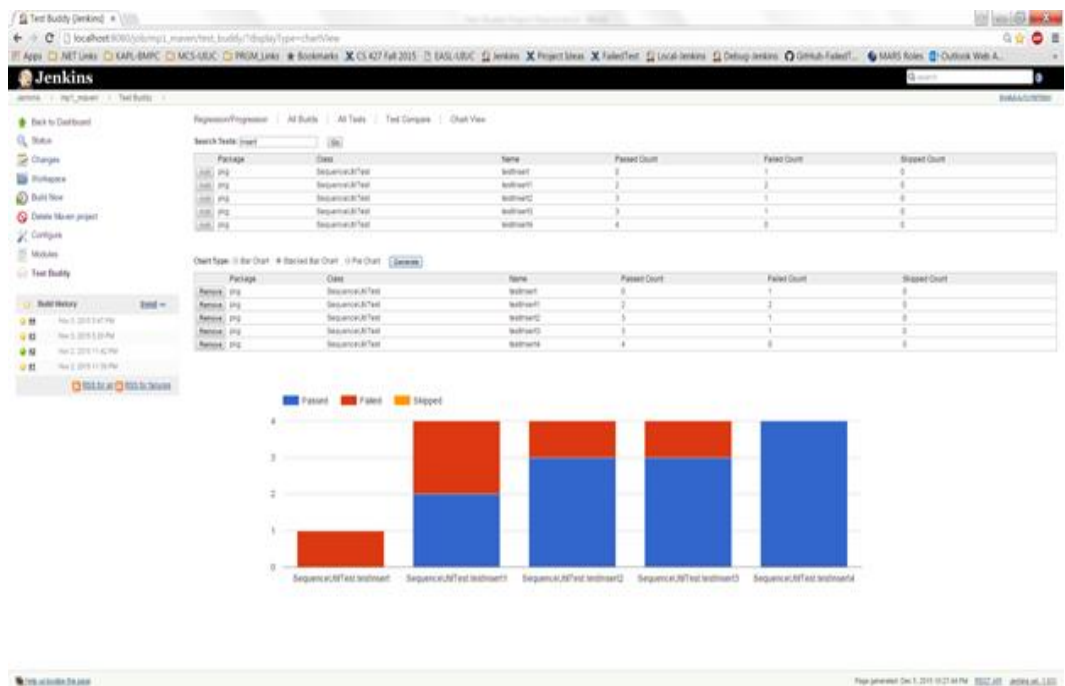


Figure 13: Generating Charts – Stacked Bar Chart

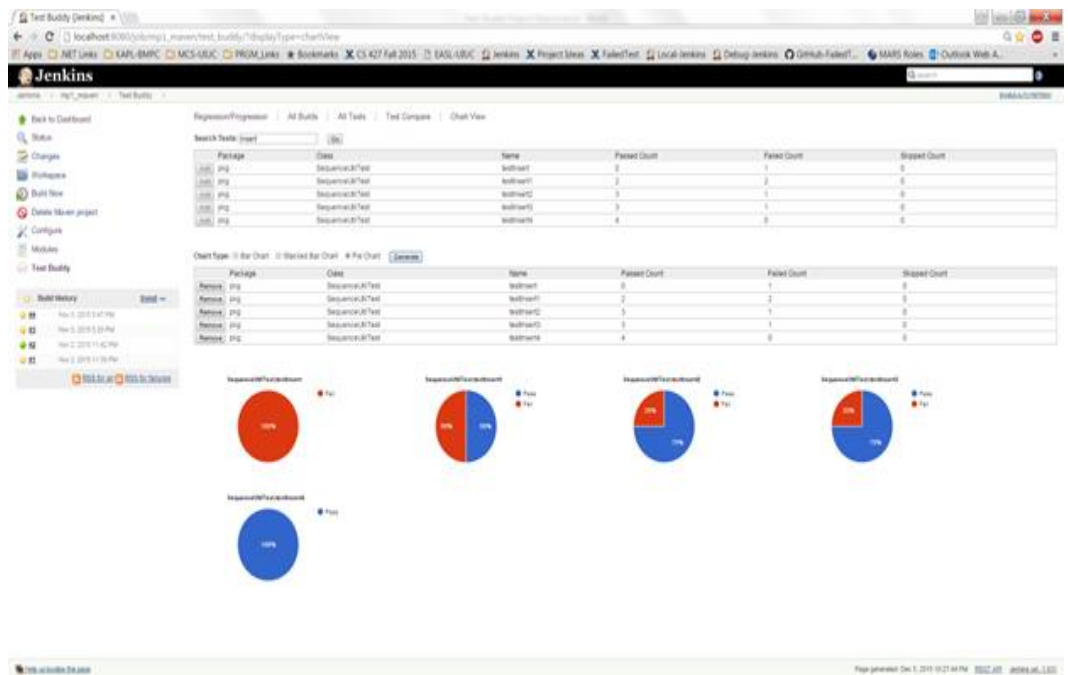


Figure 14: Generating Charts – Pie Chart

## 5. Conclusion

Through our extensive research regarding the initial plugin and personal experiences as developers, we have extended this Regression Report Plugin with features we consider will be beneficial to incorporate, including extending the email functionality to include progressed tests, failing new tests, and passing new tests, viewing all builds and tests, and comparing tests. We hope this plugin can better serve developers, testers, and upper management in the future. We would continue to work with the Regression Report Plugin team to make it publicly accessible.