Name: Udit Damle
Seat Number: 31011021025
Class: TYIT Hons (2023-34)

# Software Testing

Practical Journal

## Index

## **Practical 1:** Install Selenium IDE. Write a test suite containing a minimum of 4 test cases for different formats.

**Create a New Maven Java Application Project**
To create a new Maven project, click File > New Project and select Maven then Java Application;

**Configure Maven Project and Location**
Next, you need to configure some details for your project. For the purpose of simplicity, we're going to call this project HelloSelenium. You'll notice that when you enter the Group ID field, set this to the canonical name of your package which you generally want to set to your primary domain name in reverse, i.e. com.contradodigital, which will then automatically populate the Package name at the bottom to be com.companyname.helloselenium. This is the industry's best practice for naming your packages so that they have a unique reference.

**Open & Configure Your Pom.xml File**
Next, we need to configure your pom.xml file which is used for Maven projects to manage your dependencies. Out of the box within NetBeans, when you create a Maven project, a very basic pom.xml file is created for you.
Dependencies:

- Selenium Java
- Selenium API
- Selenium Server
- Selenium Chrome Driver
- Selenium Remote Driver
- JUnit

**pom.xml**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="https://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.contradodigital</groupId>
    <artifactId>HelloSelenium</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
    </properties>
    <pluginRepositories>
        <pluginRepository>
            <id>central</id>
            <name>Central Repository</name>
            <url>https://repo.maven.apache.org/maven2</url>
            <layout>default</layout>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
            <releases>
                <updatePolicy>never</updatePolicy>
            </releases>
        </pluginRepository>
    </pluginRepositories>
    <repositories>
        <repository>
            <id>central</id>
            <name>Central Repository</name>
            <url>https://repo.maven.apache.org/maven2</url>
            <layout>default</layout>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </repository>
    </repositories>
    <dependencies>
        <!--
https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
        <dependency>
            <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-java</artifactId>
            <version>3.141.59</version>
        </dependency>
        <!--
https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-api -->
        <dependency>
            <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-api</artifactId>
            <version>3.141.59</version>
        </dependency>
        <!--
https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-server -->
        <dependency>
            <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-server</artifactId>
            <version>3.141.59</version>
        </dependency>
        <!--
https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-chrome-driver
-->
```

```
        <dependency>
            <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-chrome-driver</artifactId>
            <version>3.141.59</version>
        </dependency>
        <!--
https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-remote-driver
-->
        <dependency>
            <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-remote-driver</artifactId>
            <version>3.141.59</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/junit/junit -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.13.2</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

Right click on project name and click on “Build with dependencies”.

**Install & set up Selenium IDE.**
Go to the following link and click on “Add to Chrome”:
https://chrome.google.com/webstore/detail/selenium-ide/mooikfkahbdckldjjndioackbalphokd/related?hl=en
Click the Selenium IDE icon in Chrome to open it up. Once it is open for the first time you will notice a basic welcome screen;
Click on Create a New Project to get started. Give your project a name so it’s clear what you are testing. In this example, we’re going to be doing some testing on YumInfo.

**Create Your First Automated Web Browser Test in Selenium IDE**
We’ve got Selenium installed and a new project created, let’s get onto creating your first automated browser test.

- Firstly click on the + button to add a new test;
- This will then open up the popup which allows you to give your new test a name. In this example, we’re going to test if we can use the search functionality on the YumInfo site to easily find a useful package.
- Once you’ve done this, you’ll notice that a new Test Case has been created for you which is in the left section of the screenshot below, but you’ll notice there are no steps that have been created yet which is why the section on the right of the screenshot below is still all blank.

What you will notice in the above screenshot is there are two core sections that we are going to look at next;

- Playback base URL – This is the landing page that you are going to start your tests from. Generally speaking, this is so you can test in the same way that your users would use the website.
- Record Button – This is in the top right coloured in red. This allows you to start the process running for recording your automated test scripts within Selenium IDE.

To get started, enter the base URL you want to work with. In our case, we're going to enter https://yum-info.contradodigital.com as that is the website we are doing the automated browser-based testing on.
This step will open a brand new Chrome window and it will inform you that recording has started.
Now all you need to do is to click around your website and use it like a user would. In this case, as this specific Test Case, we are looking to search for a package and then view the package information we're going to do just that.
Once you are done clicking around, simply navigate back to your Selenium IDE that is open and click on Stop Recording. Once you have done that you will notice that the specific steps that you have just taken within the web browser have been recorded within Selenium IDE.

What the above steps are saying is that I followed these actions;

- Open the Base URL https://yum-info.contradodigital.com
- Set the browser window size to the default of your computer setup
- Click on the HTML Element that has an ID of 'YumSearch', which in this case is the search box that allows users to search for packages
- Type into the search box "sftp" without the quotes
- Then click Enter to trigger the search
- And finally, click on the link titled FileZilla which is a relevant package that can handle SFTP based communications

What all this has shown us is that as a user doing these steps, this all works as expected on the website. Hopefully, this isn't an unexpected result that basic functionality on your website is working. But this is just a simple example we are using to get you up and running.
Save this Test Case so you can reference back to it later down the line.

**Re-Run Your First Automated Web Browser Test Case**
Now that you have recorded your first test, you want to replay it so that you are confident that it has been recorded accurately. For traditionally built websites that use a single Request/Response, you'll find that these tests generally record perfectly the first time around. Whereas for websites built using more Single Page Applications / Front End Frameworks that load content dynamically into the page past the initial page load, you'll find you will likely have a few issues with the default recordings and that the automated recording will need some manual intervention to get them to work properly.
To re-run the test you have just created, simply click on the Play button;
Once you click that button, you will notice that magic starts to happen. Your web browser will open and the exact steps that you just took will be replicated in real-time right in front of your eyes. Most importantly, once it is complete, you will see that it has been completed successfully.

**Do so for four tests:**

- Search for a package and view its information.
- Browse packages by number(0-9).
- Browse packages by alphabet.
- Click on the login button.

Click on tests, and select test suites. Click on the “+” button. Name the test suite appropriately. Click on Run all tests.

**Install Chrome Web Driver**
To do this, go into Google Chrome > Settings > About Chrome and you will see your version number there;
Head over to the Chromium Chrome Driver Downloads page and find the version that applies to you.

**Export Test Case from Selenium IDE to JUnit Format**
Next we need to export the Test Case that we created in Selenium IDE so that we can then import that into NetBeans. To do this go back to Selenium IDE and right-click the Test Suite you created then click on Export.

Then select the language we want to export the file to. The beauty of the Selenium IDE is that it is cross-language compatible which means that you can import the Test Case into any number of your preferred automated web browser testing setups you use, in this case, we’re using JUnit in Java

Once this has been exported, this will save a .java file in our example to your local file system which will look as follows;

```java
// Generated by Selenium IDE
import org.junit.Test;
import org.junit.Before;
import org.junit.After;
import static org.junit.Assert.*;
import static org.hamcrest.CoreMatchers.is;
import static org.hamcrest.core.IsNot.not;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.Dimension;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.Alert;
import org.openqa.selenium.Keys;
import java.util.*;
import java.net.MalformedURLException;
import java.net.URL;
public class PracticalOneTest {
  private WebDriver driver;
  private Map<String, Object> vars;
  JavascriptExecutor js;
  @Before
  public void setUp() {
    System.setProperty("webdriver.chrome.driver", "C:/chromedriver.exe");
    driver = new ChromeDriver();
    js = (JavascriptExecutor) driver;
    vars = new HashMap<String, Object>();
  }
  @After
  public void tearDown() {
    driver.quit();
  }
  @Test
  public void clickonloginbutton() {
    driver.get("https://yum-info.contradodigital.com/");
    driver.manage().window().setSize(new Dimension(1050, 708));
    driver.findElement(By.linkText("Login")).click();
  }
  @Test
  public void browsepackagesbynumbers() {
    driver.get("https://yum-info.contradodigital.com/");
    driver.manage().window().setSize(new Dimension(1050, 708));
    driver.findElement(By.linkText("A")).click();
    driver.findElement(By.linkText("< Home")).click();
    driver.findElement(By.linkText("Z")).click();
  }
  @Test
  public void browsepackagesbyalphabets() {
    driver.get("https://yum-info.contradodigital.com/");
    driver.manage().window().setSize(new Dimension(1050, 708));
    driver.findElement(By.linkText("0-9")).click();
    driver.findElement(By.linkText("< Home")).click();
    driver.findElement(By.linkText("0-9")).click();
    driver.findElement(By.linkText("0-9")).click();
    driver.findElement(By.linkText("< Home")).click();
    {
      WebElement element = driver.findElement(By.linkText("0-9"));
      Actions builder = new Actions(driver);
      builder.doubleClick(element).perform();
    }
  }
  @Test
  public void searchforapackageandviewitsinformation() {
    driver.get("https://yum-info.contradodigital.com/");
    driver.manage().window().setSize(new Dimension(1050, 708));
    driver.findElement(By.id("YumSearch")).click();
    driver.findElement(By.id("YumSearch")).sendKeys("sftp");
    driver.findElement(By.cssSelector(".input-group-btn > .btn")).click();
    driver.findElement(By.linkText("filezilla")).click();
  }
}
```

Make modifications to the code to include Chrome Web Driver Path:

```
System.setProperty("webdriver.chrome.driver", "C:/chromedriver.exe");
```

**Create a New JUnit Test File in NetBeans**
Ok, so back over to NetBeans. We want to import the Exported JUnit file from Selenium IDE into NetBeans so that we can manage the lifecycle of this Test Case better and work collaboratively with our colleagues. We'll jump into more around the team collaboration elements of Selenium a little later. For now, let's first jump in and get the JUnit Test added to NetBeans. To do this, right-click on a folder in your project in NetBeans and select New File, then search for JUnit, then select JUnit Test and click Next.

**Merge Your Exported Selenium IDE Test Case Into Your NetBeans JUnit Test Class**
The next step is generally best done as a copy and paste to fit the automatically generated Selenium IDE Test Case code into the standardised approach you use for your JUnit Test Classes within NetBeans. Don't just blindly copy and paste the code as while the automatically generated code is handy, you need to manage this code to fit with your specific needs and use cases.

**Run Your JUnit Test Class**
Excellent, now we're at a point where we can actually run our JUnit Test Class to confirm everything is still working as expected. To do this simply right-click and select Test File within your JUnit Test Class. If everything has merged successfully you should see your Google Chrome Web Browser kick into action, run the test and the test should pass.

## **Practical 2:** Write a test plan for any two websites.

### 1. E-commerce Website Testing Test Plan

#### Test Plan Identifier

E-COM Testing TP_1.0

#### Test Deliverables

Test cases, test reports

#### Introduction

This test plan outlines the testing approach and strategy for the e-commerce website "YourEcomSite.com."

1. To define the tools to be used throughout the testing process.
2. To communicate to the responsible parties the items to be tested, set expectations around the schedule, and define environmental needs.
3. To define how the tests will be conducted.

#### Test Tasks

- Functional Testing
- Security Testing
- Performance Testing
- Accessibility Testing
- User Acceptance Testing

#### Test Items

- Website URLs (e.g., home page, product pages, checkout)
- User accounts and profiles
- Search functionality
- Product listings and details
- Shopping cart and checkout process
- Payment gateway integration
- User reviews and ratings
- Security features (e.g., SSL, user data protection)
- Performance (load times, server responsiveness)
- Mobile device compatibility
- Accessibility for users with disabilities

#### Environmental Needs

- Test Environment: A test environment mirroring the production setup. Hardware, software, and network configurations identical to production.
- Test Data: Realistic test data for valid and invalid scenarios.

- Tools and Software: Testing tools (e.g., JMeter, Selenium, browsers) for comprehensive testing.
- Development and debugging tools for troubleshooting. Network Conditions: Testing under various network conditions (3G, 4G, broadband) to assess website performance.
- Mobile Devices: Testing on smartphones and tablets with various screen sizes and operating systems.

## Features To Be Tested

Features to be tested include the following:

- As a shopper, logging into the website as a shopper
- As a shopper, navigating the store
- As a shopper, adding items to a shopping cart
- As a shopper, removing items from a shopping cart
- As a shopper, purchasing multiple units of the same item
- As a shopper, initiating a return
- As a shopper, contacting support
- As a shopper, completing an order cycle
- As a shopper, cancelling an order
- As a shopper, leaving a review
- As an admin, granting a refund
- As an admin, fulfilling an order
- As an admin, answering a support inquiry
- As an admin, moderating reviews
- As an admin, validating in-stock/out-of-stock

## Features Not To Be Tested

Third-party services and APIs not controlled by the website (e.g., payment gateway services)

## Responsibilities

- Testers: Test case execution, reporting defects
- Developers: Bug fixes and code improvements
- Product Owners: Defining acceptance criteria

## Staffing and Training Needs:

- Testers should have experience in e-commerce testing.
- Training on accessibility testing may be required.

## Approach

Tests will be conducted per the documented test cases stored in TestLodge. The test manager will create test runs for each tester. The tester will execute the tests in TestLodge and mark each case

as Pass / Fail / Skip. The tester should leave notes on actual results and any other relevant details when possible.
When tests are marked as Fail, bug reports will automatically be created in the issue tracker integrated with TestLodge.
Once complete, the test manager should review the test run reports in TestLodge and report back to the team accordingly.

## Schedule

Test planning and test case creation: Weeks 1-2
Functional, usability, and security testing: Weeks 3-4
Performance testing: Weeks 5-6
Accessibility and user acceptance testing: Weeks 7-8
Test report generation: Week 9

## Item Pass/Fail Criteria

All core functionality of the systems should function as expected and outlined in the individual test cases. There must be no critical defects found and an end user must be able to complete a purchase cycle successfully and initiate a refund without any errors. 95% of all test cases should pass and no failed cases should be crucial to the end user's ability to use the application.

## Risks and Contingencies

Risks include unexpected load on the website, third-party service outages, and security vulnerabilities.
Contingencies include load balancing, monitoring, and timely security patches.

## Suspension & Resmuption Criteria

Testing should be paused immediately if either system experiences login issues or failure in any basic CRUD (Create, Read, Update and Delete) actions.

## Approvals

The test manager and product manager both must agree on the completion of the testing project and determine when it's ready to proceed to the next step.

# 2. ATM Test Plan

## Test Plan Identifier

TP-ATM-001

## Test Deliverables

The following deliverables will be produced during testing:

- Test plan
- Test cases
- Test execution reports
- Defect reports

## Introduction

The purpose of this test plan is to outline the testing approach and activities for the ATM System. This document provides an overview of the scope, objectives, resources, and schedule for the testing phase.

## Testing Tasks

The following activities must be completed:

- Test plan prepared.
- Functional specifications written and delivered to the testing team
- The environment should be ready for testing (test data, test logins, test payment information, etc).
- Perform the tests.
- Prepare test summary report.

## Test Items

The test items include the entire ATM System, including hardware and software components.

## Environmental Needs:

- Physical ATM machines
- Test data (bank cards)
- Network connectivity
- Power supply

## Features To Be Tested

Features to be tested include the following:

- Cash withdrawal
- Balance inquiry
- Deposit function
- Card insertion and removal
- PIN entry
- Receipt printing

- Screen navigation
- Error handling

## Features Not To Be Tested

The following features will not be tested in this phase:

- Network connectivity (external systems)
- Hardware maintenance and servicing

## Responsibilities

Testers (Responsible for test case execution and reporting)
Stakeholders (Reviewing and approving test results)

## Staffing And Training Needs

Testers will be trained in ATM operation and testing procedures.

## Approach

Testing will consist of manual testing using physical ATM machines. Test cases will cover a range of scenarios, including normal and exceptional conditions.

## Schedule

Testing will start on January 25, 2023, and conclude on February 15, 2023.

## Item Pass/Fail Criteria

Cash withdrawal: Pass if successful, fail if any error occurs. - Balance inquiry: Pass if an accurate balance is displayed, fail if inaccurate. - Deposit function: Pass if successful deposit, fail if any error occurs.

## Risks And Contingencies

Risk: ATM hardware malfunctions
Contingency: Immediate repairs and resumption of testing after resolution.

## Suspension and Resumption Criteria

Testing may be suspended if critical hardware issues arise. Testing will resume once the hardware issues are resolved.

## Approvals

The test manager and product manager both must agree on the completion of the testing project and determine when it’s ready to proceed to the next step.

## **Practical 3:** Study ATM System specifications and report various bugs.

### System Specifications:

The proposed software will control a simulated ATM with hardware components like a card reader, customer console, cash dispenser, printer, and operator control. It communicates with the bank's computer system over a secure link.

Customers insert their ATM card and enter a PIN for validation before conducting transactions, which include cash withdrawals (subject to bank approval), deposits in envelopes (verified by an operator), money transfers, and balance inquiries. Transactions are sent to the bank for verification and approval.

If a customer enters an invalid PIN three times, the card is permanently retained, requiring the customer to contact the bank for retrieval.

The ATM provides printed receipts for successful transactions, displaying transaction details. An operator, who starts and stops the ATM's operation using a key-operated switch, verifies and inputs the total cash on hand during startup.

To ensure auditability and recovery from hardware failures, the ATM maintains an internal transaction log. It records startup and shutdown events, communication with the bank, cash dispensing, and envelope receipts. Log entries omit PINs for security.

### Bugs:

**Security Concerns:** The description mentions that the ATM retains a customer's card. While this is common practice, it raises potential security issues if not handled properly. Ensuring the security of retained cards, including encryption, storage, and retrieval procedures, is crucial.
**Operator Control:** Operator control through a key-operated switch can be a potential security concern. Proper authentication and authorization mechanisms for operators are vital to prevent unauthorized access.
**Transaction Logging:** The internal transaction log is essential for auditing and recovery. However, ensuring that log entries are protected against tampering and unauthorized access is critical.
**Error Handling:** The description mentions that in case of a transaction failure, the ATM will display an explanation. Proper error handling is crucial to provide informative and user-friendly messages to customers.
**Network Security:** While the description focuses on the ATM's interaction with the bank's computer, ensuring the security of data transmission is vital. This includes encryption and secure communication protocols.
**PIN Security:** PIN handling should adhere to industry standards to ensure that PINs are securely managed and not stored in plain text.

**Timeouts and State Management:** The description mentions timeout periods for deposit envelopes. Proper management of timeouts, retries, and system states is crucial for ensuring the reliability of the system.

## **Practical 4:** Write a Test Incident Report for a case study.

Hi there!

Well, I nearly caused a panic today because I thought I had found a mega showstopper on the trading system we are testing. The test manager and others got involved in examining databases first on the server and then on the gateway that feeds the clients, checking update logs from processes that ran overnight as well as checking data passed to the client. Eventually, I found the problem. I had mis-clicked on a .bat file when running up a client and had run up the wrong client environment. By that time the test manager was ready to say a few short words in my ear, particularly as the development people had started to get involved and they have zero tolerance for mistakes made by testers. The only saving grace was that I found the mistake and not one of the developers.

It was, objectively, an interesting mistake. When you log into the server test environments, the panels always show the environment to which you are connected. In our case, we have two test environments called Systest14 and Systest15 and my tests were set up in Systest15. To run up the clients, we have to run .bat files for either a 14 or 15 client. I had started two clients, that is two exchange participants, so I could do some trading between them.
It appears I started the first client OK in environment 15 but when I started the second, I accidentally moved the mouse a fraction so it ran the 14 .bat file that is next to it in the Explorer file list. To make matters worse, the client screens do not show the environment to which you are attached.

At first, I felt a bit stupid having caused much hectic and wasted activity. On reflection, I thought that if I, as a reasonably competent person, can make a mistake like this then something is wrong. On the server side when I log on to a test environment, I have to enter the environment name and it's shown on all the panels. On the client side, I run a client test environment by selecting a .bat file from a list of many and have to ensure I click on the right file. There is neither a display nor the ability to determine the client environment in which I am working.
So I am going to log this as a high priority, or even showstopper, error - the client does not show the environment. In real-life terms, it means a real user could be connected to the production system and think he is connected to a test system and screw up trading. I know this happened once on the equities trading system when a trader entered a load of test transactions into the production system by mistake and caused mayhem.

As an addendum to this story, a couple of days later one of the testers found what appeared to be another mega showstopper. He and the test manager spent three hours crawling all over the system before they discovered the 'error'. A new filter had been added to the client software to filter transactions displayed in panels by geographical market. Unknown to them, it was set to a default of the German market, whereas they thought they were in the UK market. Consequently, at first sight, it appeared there were fundamental problems with the network transaction bus and the message-broadcasting systems. Apart from the issue that they should have been informed of this change, it raised a similar problem to the one I had experienced -the client system does not display the market in which you are trading.
Well - I'm off for another happy day at the office! All the best

## Test Incident Report:

**Incident Report Identifier:** TIR_01

**Summary:**
The incident involved a tester who mistakenly ran the wrong client environment while testing a trading system, causing confusion and unnecessary investigations by the test manager and development team. The incident brought to light a critical issue related to the lack of clear identification of the client environment during testing.

**Incident Description(inputs, expected results, actual results, anomalies, date and time, procedure step, environment, attempts to repeat, testers and observers):**
The incident occurred when the tester attempted to run two exchange participants (clients) for trading on the system. The testing environment had two test environments called Systest14 and Systest15. The tester's tests were set up in Systest15. When attempting to start the second client, a slight mouse movement caused the tester to execute the Systest14 environment's .bat file instead of Systest15. Unfortunately, the client screens did not display the environment to which they were connected, causing confusion. As a result of this confusion, extensive investigations were conducted on the server and gateway, involving the examination of databases, update logs, and data passed to clients. Ultimately, the tester realized the error and rectified it. The incident raised concerns about the lack of clear identification of the client environment during testing, as well as the potential for similar mistakes in the future. It was noted that this issue could lead to real users mistakenly connecting to the production system when they intend to use the test system, potentially causing significant problems, as seen in a past incident on the equities trading system.

**Impact:** The incident had several impacts: The incident led to wasted time and resources as the test manager and development team became involved in extensive investigations. It highlighted the critical issue of the client system not displaying the connected environment, potentially leading to confusion and incorrect trading actions. The incident served as a reminder of the importance of clear environmental identification in a trading system, as similar issues could lead to severe problems in real-life trading scenarios.

**Practical 5:** Create an ATM System and prepare the test case and test using black box testing

```python
class Account:
    def __init__(self, account_number, name, pin, balance):
        self.account_number = account_number
        self.name = name
        self.pin = pin
        self.balance = balance

    def check_balance(self):
        return self.balance

    def withdraw(self, amount):
        if self.balance < amount:
            print("Insufficient Balance")
        else:
            self.balance -= amount

class ATM:
    def __init__(self):
        self.accounts = {}

    def add_account(self, account):
        self.accounts[account.account_number] = account

    def authenticate_account(self, account_number, pin):
        if account_number in self.accounts:
            account = self.accounts[account_number]
            if str(pin) == account.pin:
                return account
        return None

    def display_welcome_message(self):
        print("Welcome to Bank of XYZ ATM")
        print("--------------------------")

    def perform_action(self, account):
        print(f"Welcome, {account.name}")
        option = int(input("Select the action you wish to perform\n1:
Check Balance\n2: Withdrawal\n"))
        if option == 1:
            balance = account.check_balance()
            print(f"Your balance is: {balance}")
        elif option == 2:
            amount = int(input("Please enter the amount to withdraw:
"))
            account.withdraw(amount)
            print("Withdrawal successful")

def main():
    atm = ATM()

    account1 = Account(101, "ABC", "1234", 10000)
    account2 = Account(102, "DEF", "4678", 7000)
    account3 = Account(103, "GHI", "9101", 3000)

    atm.add_account(account1)
    atm.add_account(account2)
    atm.add_account(account3)

    atm.display_welcome_message()
    account_number = int(input("Please enter account number: "))
    pin = int(input("Please enter your PIN to proceed: "))

    authenticated_account = atm.authenticate_account(account_number,
pin)
    if authenticated_account:
        atm.perform_action(authenticated_account)
    else:
        print("Incorrect account number or PIN. Please try again.")

if __name__ == "__main__":
    main()
```

To perform black box testing for the provided ATM System, you need to create test cases that cover various aspects of the system's functionality. Black box testing focuses on testing the system's functionality without looking into its internal structure. Below are some test cases for the ATM System you provided:

**Valid Account Login and Check Balance:**

- Test Case: Authenticate with a valid account number and PIN, then choose to check the balance.
- Expected Result: The system should display the account balance.

**Valid Account Login and Withdrawal:**

- Test Case: Authenticate with a valid account number and PIN, then choose to withdraw an amount within the account balance.
- Expected Result: The system should complete the withdrawal, and the account balance should be updated accordingly.

**Invalid Account Login:**

- Test Case: Authenticate with an invalid account number and PIN.
- Expected Result: The system should display an error message indicating that the account number or PIN is incorrect.

**Insufficient Balance for Withdrawal:**

- Test Case: Authenticate with a valid account, attempt to withdraw an amount greater than the account balance.
- Expected Result: The system should display an "Insufficient Balance" error message.

**Invalid Input for Withdrawal:**

- Test Case: Authenticate with a valid account, attempt to withdraw with an invalid input (e.g., a negative amount or a non-numeric input).
- Expected Result: The system should display an error message indicating that the input is invalid.

**Account Not Found:**

- Test Case: Attempt to authenticate with an account number that does not exist.
- Expected Result: The system should display an error message indicating that the account number is not found.

**Correct Withdrawal and Balance Update:**

- Test Case: Authenticate with a valid account, perform a withdrawal, and then check the updated balance.
- Expected Result: The system should correctly update the balance after the withdrawal.

**Menu Selection - Invalid Option:**

- Test Case: Authenticate with a valid account and select an invalid option in the menu (e.g., an option other than 1 or 2).
- Expected Result: The system should display an error message indicating that the option is invalid.

**Valid Account Login and Cancel Transaction:**

- Test Case: Authenticate with a valid account, start a transaction, but choose to cancel it.
- Expected Result: The system should abort the transaction and return to the main menu.

**Valid Account Login and Withdrawal with Maximum Balance:**

- Test Case: Authenticate with a valid account, and attempt to withdraw an amount equal to the account's current balance.
- Expected Result: The system should complete the withdrawal, and the account balance should become zero.

These test cases cover various scenarios, including valid and invalid inputs, different transaction types, and error conditions.

## **Practical 6:** Install the Selenium server (Selenium RC) and demonstrate it using a script in Java.

Create a new Java project add the respective jars and do the following configurations.

Selenium-RC Installation
Assuming Java is installed and the PATH environmental variable is configured, follow the steps below:

1. Download the Selenium RC Zip file from: https://www.seleniumhq.org/download/
2. Unzip the file. There will be a folder per supported language and selenium-server folder which contains the selenium-server-standalone.jar which is the Selenium-RC server

Add all the external jars to your project in Eclipse IDE

To configure the RC server:
Select run - External Tools - External Tools configuration
Double click on Program on the left pane
On the right pane, add the values as following:

1) Name: RC_Server
2) Location: yourJDKlocation\java.exe
3) Working Directory: your directory location where jar is downloaded or stored.
4) Arguments: -jar selenium-server-standalone-3.13.0.jar

Then click on Apply and Run

The server starts and we can see the port number on the Console:

Add a new java class and type the following code..

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.remote.DesiredCapabilities;

public class Hello {
 static String driverPath = "D:\\Udit\\software
testing\\selenium\\GeckoDriver.exe";
    public static WebDriver driver;


 public static void main(String[] args) {
  int a=10,b=20;
  System.out.println("testing the RC SERVER");
  System.out.println(a+b);
  System.out.println("Selenium demo");

  System.setProperty("webdriver.gecko.driver",driverPath);
  DesiredCapabilities capabilities = DesiredCapabilities.firefox();
  capabilities.setCapability("marionette",true);
  driver= new FirefoxDriver();
  driver.get("https://www.facebook.com/");
  driver.manage().window().maximize();
  driver.quit();
}

}
```

**Practical 7:** Write and test a program to count the number of checkboxes on the page checked and unchecked count.

**index.html**

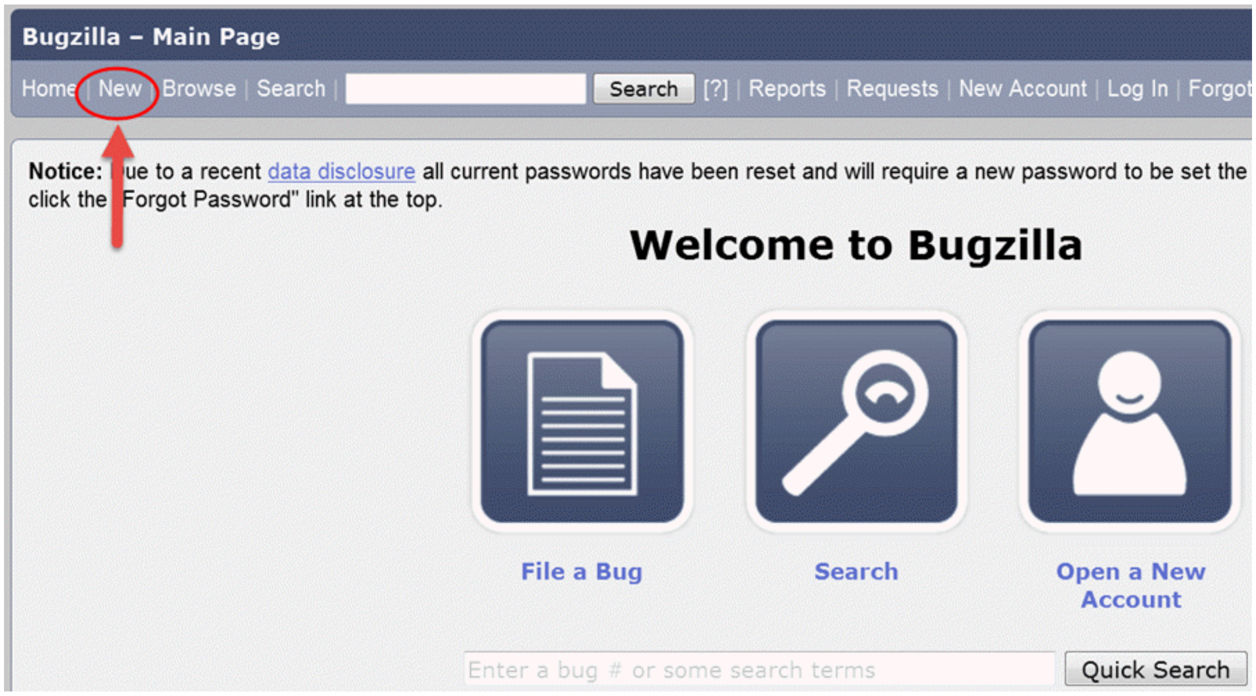
```html
<!DOCTYPE html>
<html>
<body>
<form>
<h2>Text Input</h2>
  First name:<br>
<input type="text" name="firstname">
<br>
  Last name:<br>
<input type="text" name="lastname">
<br>
<h2>Select Gender</h2>

<input type="radio" name="gender" value="male" checked> Male<br>
<input type="radio" name="gender" value="female"> Female<br>
<input type="radio" name="gender" value="other"> Other<br>

<h2>Select Languages Known</h2>
<input type="checkbox" name="lang" value="Java"> Java<br>
<input type="checkbox" name="lang" value="PHP" checked="checked"> PHP<br>
<input type="checkbox" name="lang" value="C#"> C#<br>
<input type="checkbox" name="lang" value="Python" checked="checked">
Python<br>
<input type="checkbox" name="lang" value="Ruby"> Ruby<br>
<br><br>
<input type="submit" value="Submit">
</form>
</body>
</html>
```

TestCheckBoxes.java

```java
import java.util.*;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;

 class TestCheckBoxes
 {
      public static void main(String args[])
      {
        System.setProperty("webdriver.gecko.driver","E:\\soft\\software
testing\\Mozilla Driver\\geckodriver.exe");
        WebDriver driver = new FirefoxDriver();
        driver.get("file:///C:/Users/Admin/Desktop/reg.html");
        int chk = 0;
        int unchk = 0;
        List <WebElement> els =
driver.findElements(By.xpath("//input[@type='radio']"));
        for(WebElement el : els)
        {
            if(el.isSelected()) {
                  chk++;
            }
            else{
                  unchk++;
            }

        }
        System.out.println("Total checked items" + chk);
      System.out.println("Total unchecked items" + unchk);
      }
 }
```

## **Practical 8:** Identify a Bug and create a bug report on Bugzilla.

Step 1: To create a new bug in Bugzilla, visit the home page of Bugzilla and click on the NEW tab from the main menu


Bugzilla – Main Page
Home | New | Browse | Search | Search [?] | Reports | Requests | New Account | Log In | Forgot
File a Bug
Search
Open a New Account
Enter a bug # or some search terms
Quick Search

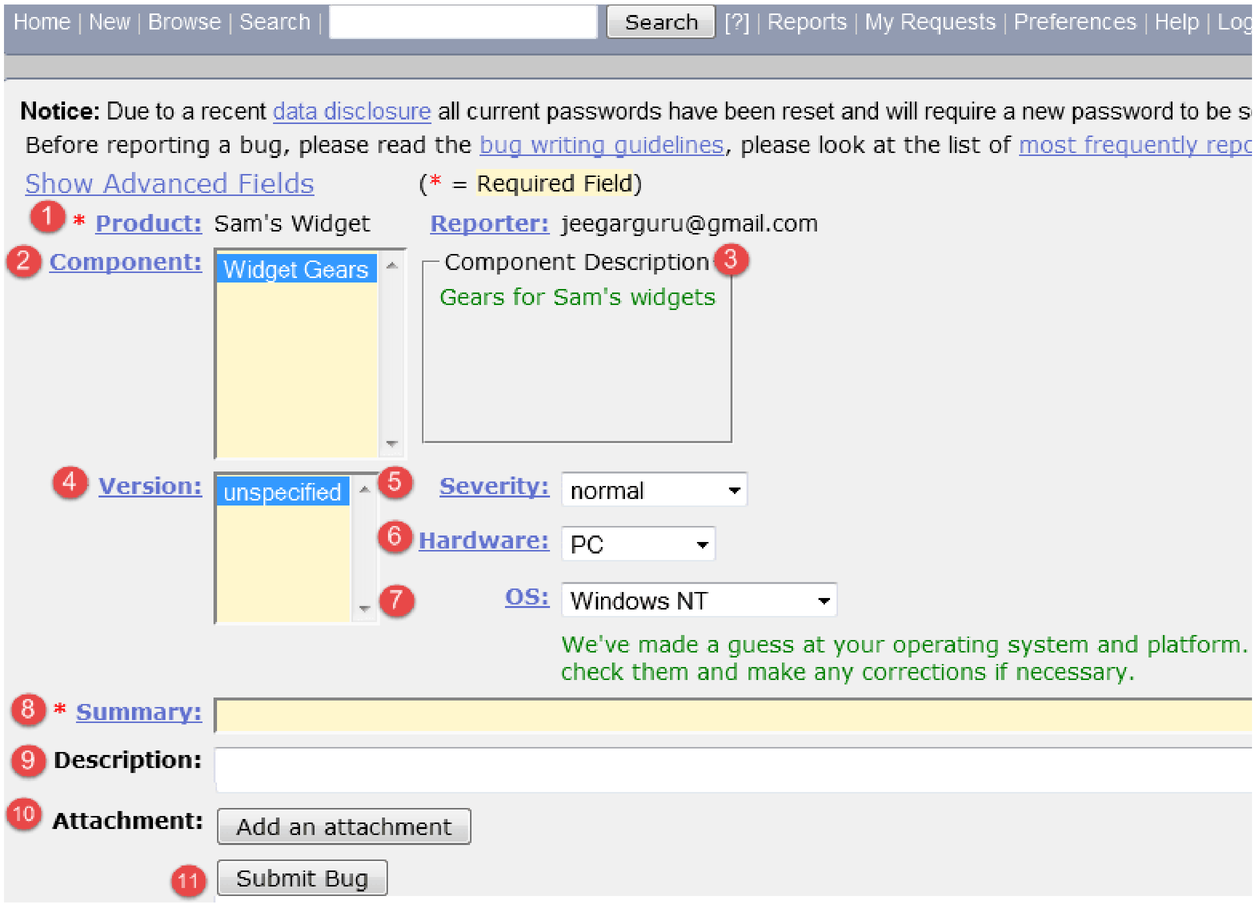
**Notice:** Due to a recent data disclosure all current passwords have been reset and will require a new password to be set the click the "Forgot Password" link at the top.

**Welcome to Bugzilla**

Step 2: In the next window

1. Enter Product
2. Enter Component
3. Give Component description
4. Select version,
5. Select severity
6. Select Hardware
7. Select OS
8. Enter Summary
9. Enter Description\
10. Attach Attachment
11. Submit

NOTE: The above fields will vary as per your customization of Bugzilla


Home | New | Browse | Search | Search [?] | Reports | My Requests | Preferences | Help | Log

**Notice:** Due to a recent data disclosure all current passwords have been reset and will require a new password to be s
Before reporting a bug, please read the bug writing guidelines, please look at the list of most frequently repo

Show Advanced Fields (* = Required Field)

1 * **Product:** Sam's Widget **Reporter:** jeegarguru@gmail.com

2 **Component:** Widget Gears

Component Description 3
Gears for Sam's widgets

4 **Version:** unspecified

5 **Severity:** normal

6 **Hardware:** PC

7 **OS:** Windows NT

We've made a guess at your operating system and platform.
check them and make any corrections if necessary.

8 * **Summary:**

9 **Description:**

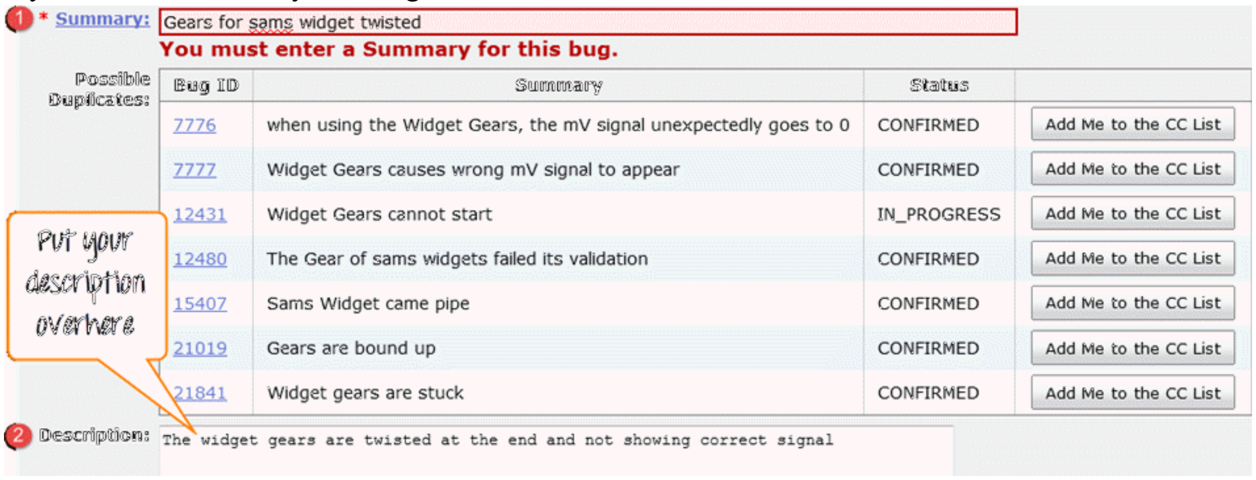
10 **Attachment:** Add an attachment

11 Submit Bug

NOTE: The mandatory fields are marked with *.
In our case field's

- Summary
- Description

Are mandatory
If you do not fill them you will get a screen like below


1 * **Summary:** Gears for sams widget twisted
**You must enter a Summary for this bug.**

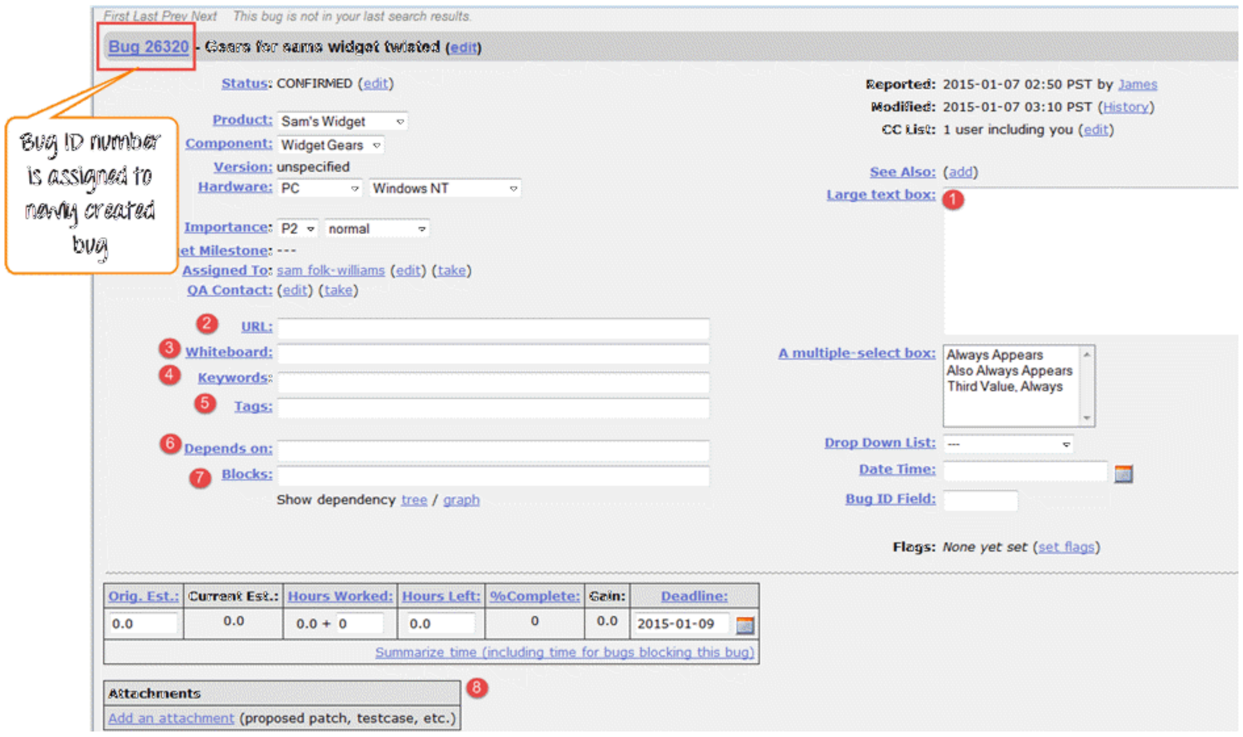
Possible Duplicates:

| Bug ID | Summary | Status | |
|---|---|---|---|
| 7776 | when using the Widget Gears, the mV signal unexpectedly goes to 0 | CONFIRMED | Add Me to the CC List |
| 7777 | Widget Gears causes wrong mV signal to appear | CONFIRMED | Add Me to the CC List |
| 12431 | Widget Gears cannot start | IN_PROGRESS | Add Me to the CC List |
| 12480 | The Gear of sams widgets failed its validation | CONFIRMED | Add Me to the CC List |
| 15407 | Sams Widget came pipe | CONFIRMED | Add Me to the CC List |
| 21019 | Gears are bound up | CONFIRMED | Add Me to the CC List |
| 21841 | Widget gears are stuck | CONFIRMED | Add Me to the CC List |

2 Description: The widget gears are twisted at the end and not showing correct signal

Step 4) Bug is created ID# 26320 is assigned to our Bug. You can also add additional information to the assigned bug like URL, keywords, whiteboard, tags, etc. This extra-information is helpful to give more detail about the Bug you have created.

1. Large text box
2. URL
3. Whiteboard
4. Keywords
5. Tags
6. Depends on
7. Blocks
8. Attachments


First Last Prev Next This bug is not in your last search results.

Bug 26320 - Gears for same widget twisted (edit)

Status: CONFIRMED (edit)
Product: Sam's Widget
Component: Widget Gears
Version: unspecified
Hardware: PC Windows NT
Importance: P2 normal
Target Milestone: ---
Assigned To: sam folk-williams (edit) (take)
QA Contact: (edit) (take)
URL:
Whiteboard:
Keywords:
Tags:
Depends on:
Blocks:
Show dependency tree / graph

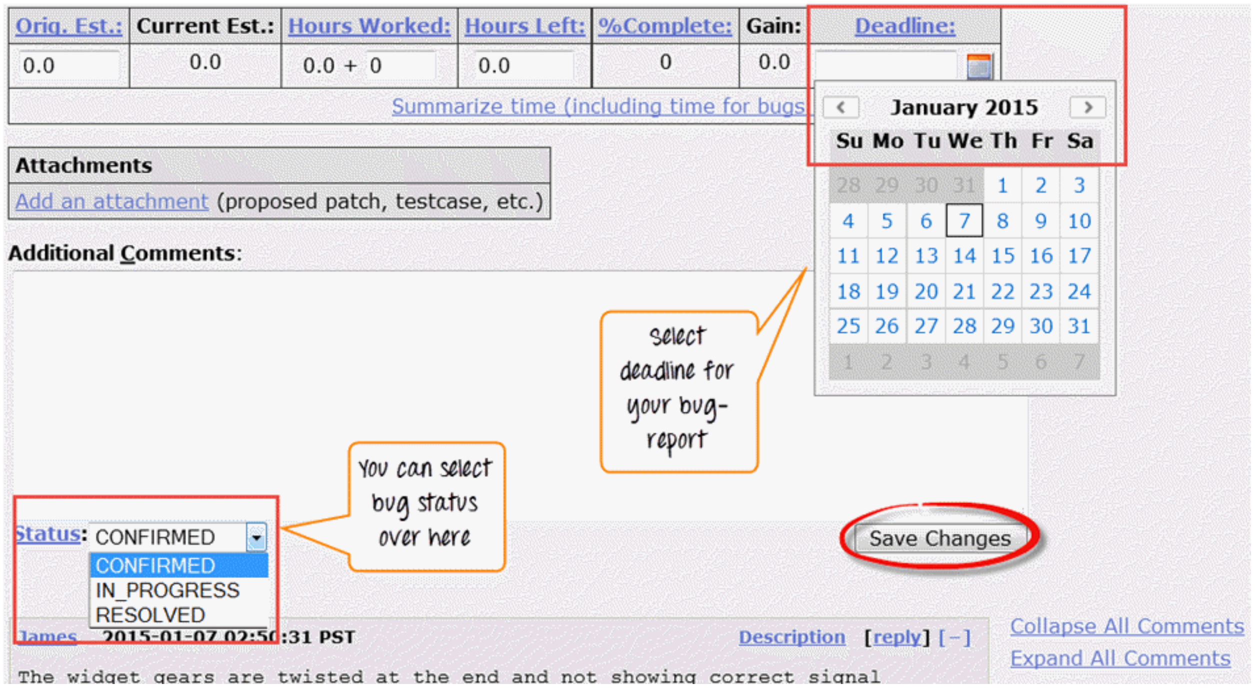
Reported: 2015-01-07 02:50 PST by James
Modified: 2015-01-07 03:10 PST (History)
CC List: 1 user including you (edit)
See Also: (add)
Large text box:
A multiple-select box: Always Appears / Also Always Appears / Third Value, Always
Drop Down List: ---
Date Time:
Bug ID Field:
Flags: None yet set (set flags)

| Orig. Est.: | Current Est.: | Hours Worked: | Hours Left: | %Complete: | Gain: | Deadline: |
|---|---|---|---|---|---|---|
| 0.0 | 0.0 | 0.0 + 0 | 0.0 | 0 | 0.0 | 2015-01-09 |

Summarize time (including time for bugs blocking this bug)

Attachments
Add an attachment (proposed patch, testcase, etc.)

Step 5: In the same window if you scroll down further. You can select deadline date and also status of the bug. Deadline in Bugzilla usually gives the time-limit to resolve the bug in given time frame.



| Orig. Est.: | Current Est.: | Hours Worked: | Hours Left: | %Complete: | Gain: | Deadline: |
|---|---|---|---|---|---|---|
| 0.0 | 0.0 | 0.0 + 0 | 0.0 | 0 | 0.0 | |

Summarize time (including time for bugs

| Attachments |
|---|
| Add an attachment (proposed patch, testcase, etc.) |

Additional Comments:

Status: CONFIRMED

James 2015-01-07 02:50:31 PST

```
The widget gears are twisted at the end and not showing correct signal
```

## **Practical 9:** Report of Performance and Stress Testing on a web application- “Offee”.

### Step 1: Add Thread Group

- Start JMeter.
- Select the "Test Plan" on the tree.
- Add a Thread Group.
- Right-click on the "Test Plan" and choose "Add -> Threads (Users) -> Thread Group."

### Step 2: Adding JMeter Elements

Now, let's add JMeter elements for "offee.in" testing:

**HTTP Request Defaults:**

1. Add an HTTP Request Defaults element to set the website name.
   - Right-click on the Thread Group and choose "Add -> Config Element -> HTTP Request Defaults."
   - In the HTTP Request Defaults control panel, enter the website name under test (e.g., "https://www.offee.in").

**HTTP Request:**

2. Add an HTTP Request to specify the URL you want to test.
   - Right-click on the Thread Group and select "Add -> Sampler -> HTTP Request."
   - In the HTTP Request Control Panel, enter the specific Path field for "offee.in" (e.g., "/"). This indicates the URL request you want to send to the "offee.in" server.

### Step 3: Adding Graph Result

3. You can add a listener to view the test results in a graph format.
   - Right-click on the "Test Plan," then choose "Add -> Listener -> Graph Results."

### Step 4: Run the Test and Get the Results

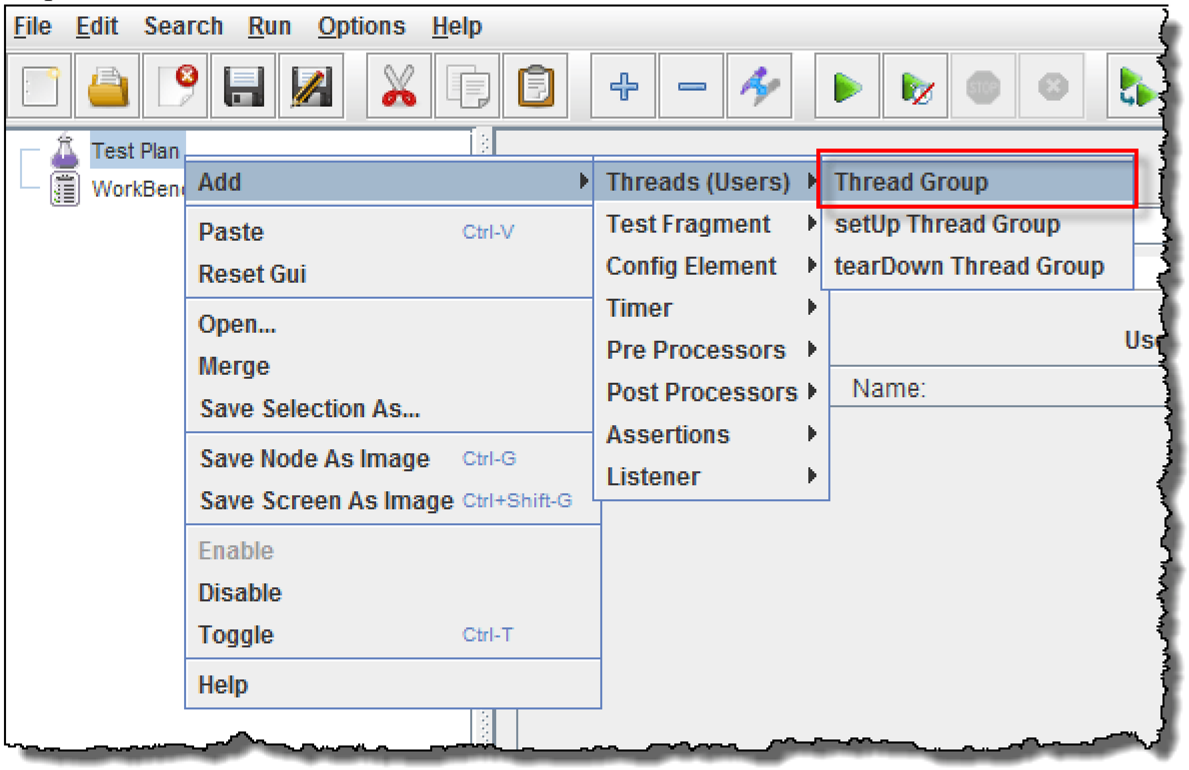
4. Press the "Run" button (Ctrl + R) on the toolbar to start the testing process.
5. You will see the test results displayed on the graph in real-time, showing the performance of the "offee.in" website.

## **Practical 10:** Create 10 threads(users) and perform load testing on website with JMeter.

**Step 1: Add Thread Group**

1. Start JMeter
2. Select Test Plan on the tree
3. Add Thread Group

Right click on the "Test Plan" and add a new thread group: Add -> Threads (Users) -> Thread Group

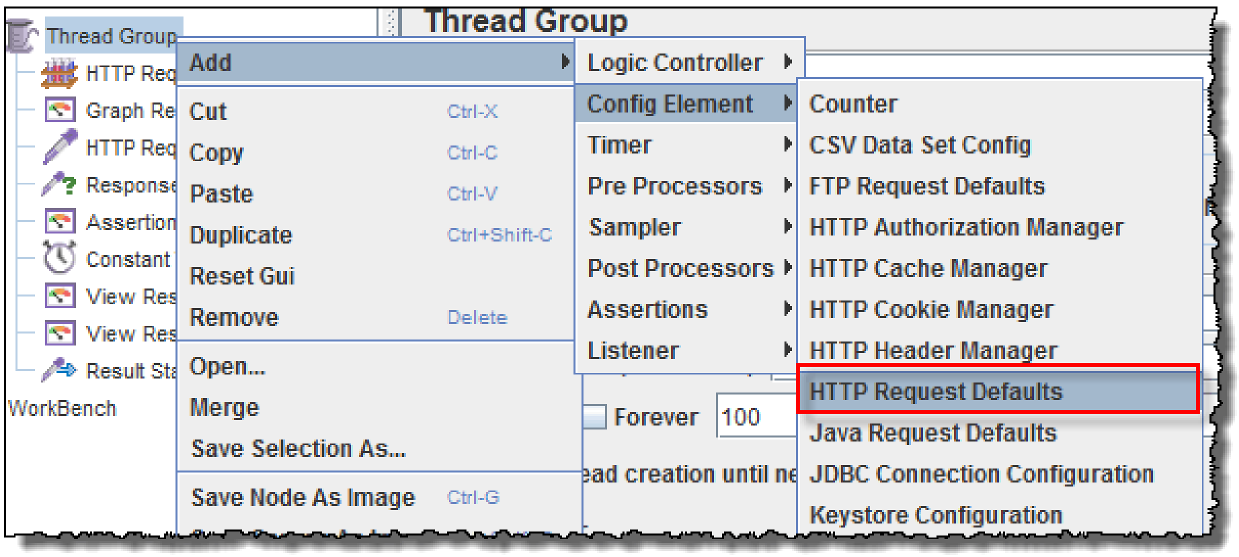


**Step 2: Adding JMeter elements**

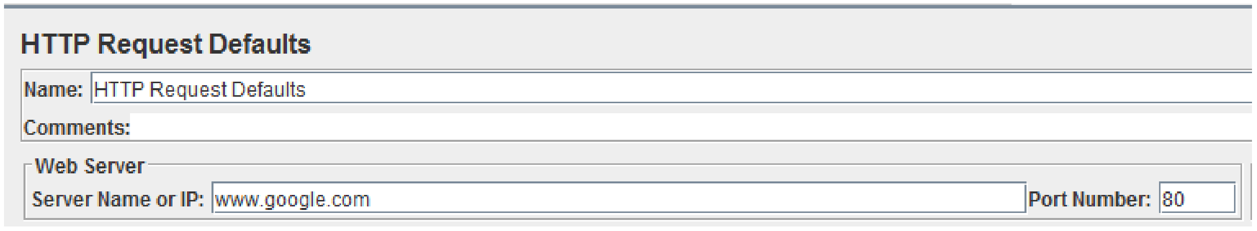
Now we determine what JMeter elements in this test. The elements are

- HTTP request Default

This element can be added by right-clicking on the Thread Group and selecting: Add -> Config Element -> HTTP Request Defaults.



In the HTTP Request Defaults control panel, enter the Website name under test ([http://www.google.com](http://www.google.com))



HTTP Request Defaults

Name: HTTP Request Defaults

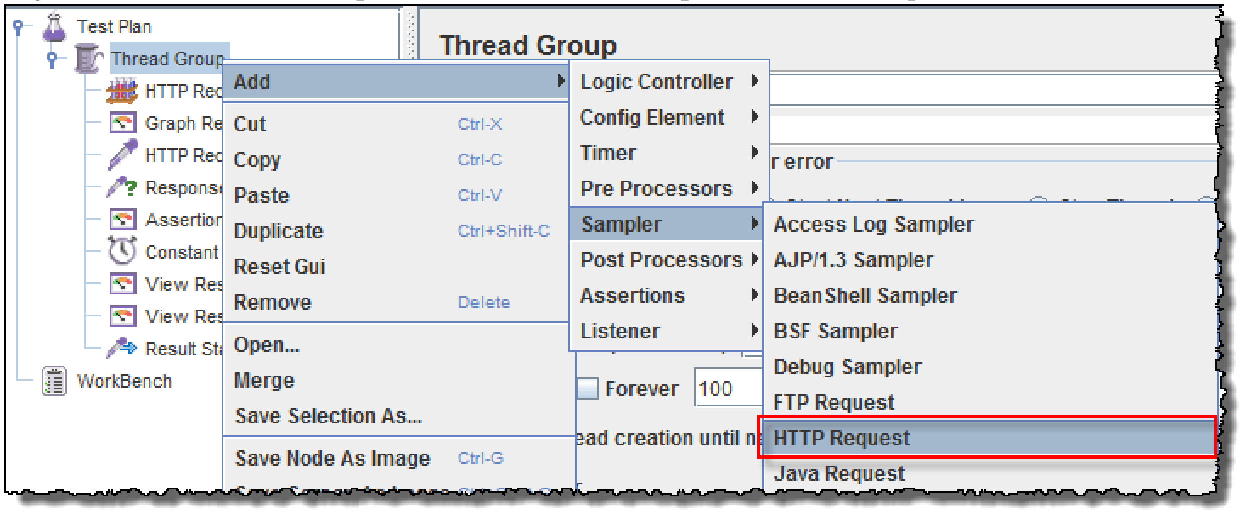
Comments:

Web Server

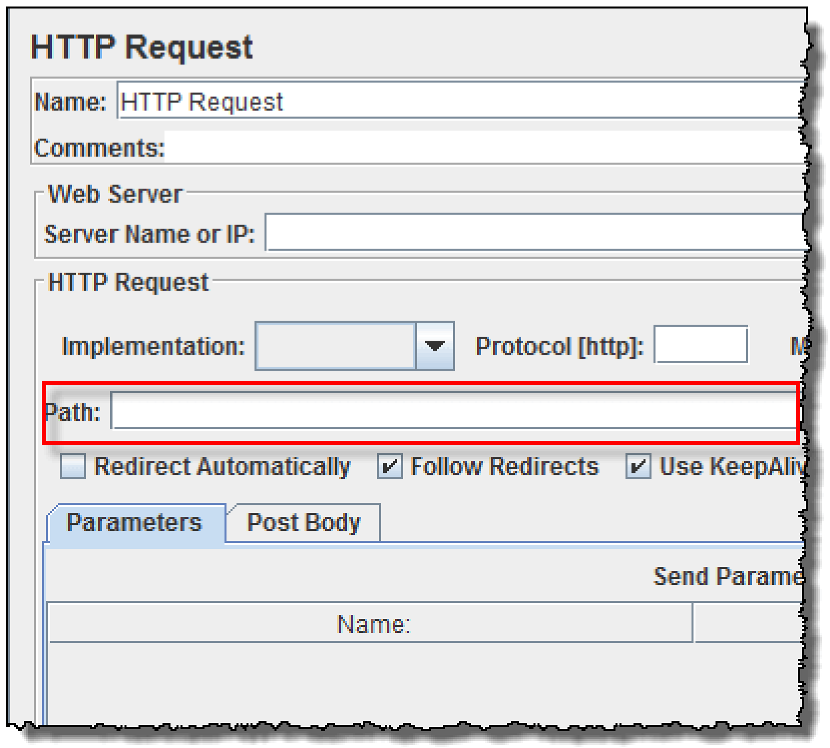
Server Name or IP: www.google.com Port Number: 80

- HTTP Request

Right-click on Thread Group and select: Add -> Sampler -> HTTP Request.



In HTTP Request Control Panel, the Path field indicates which URL request you want to send to Google server.


**HTTP Request**

Name: HTTP Request

Comments:

Web Server

Server Name or IP:

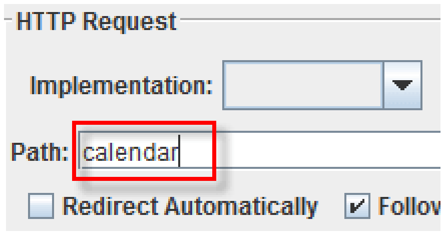
HTTP Request

Implementation: Protocol [http]:

Path:

☐ Redirect Automatically ☑ Follow Redirects ☑ Use KeepAliv

For example, if you enter "calendar" in Path field. JMeter will create the URL request http://www.google.com/calendar to Google server


HTTP Request

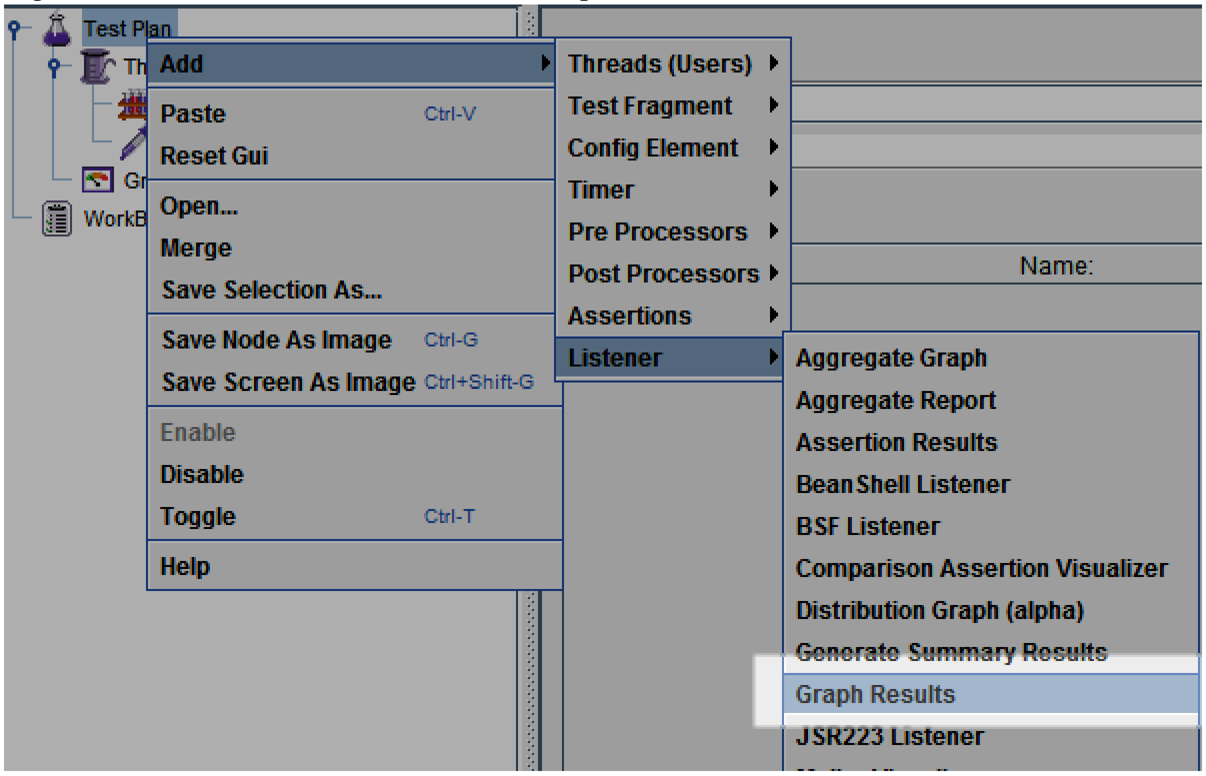
Implementation:

Path: calendar

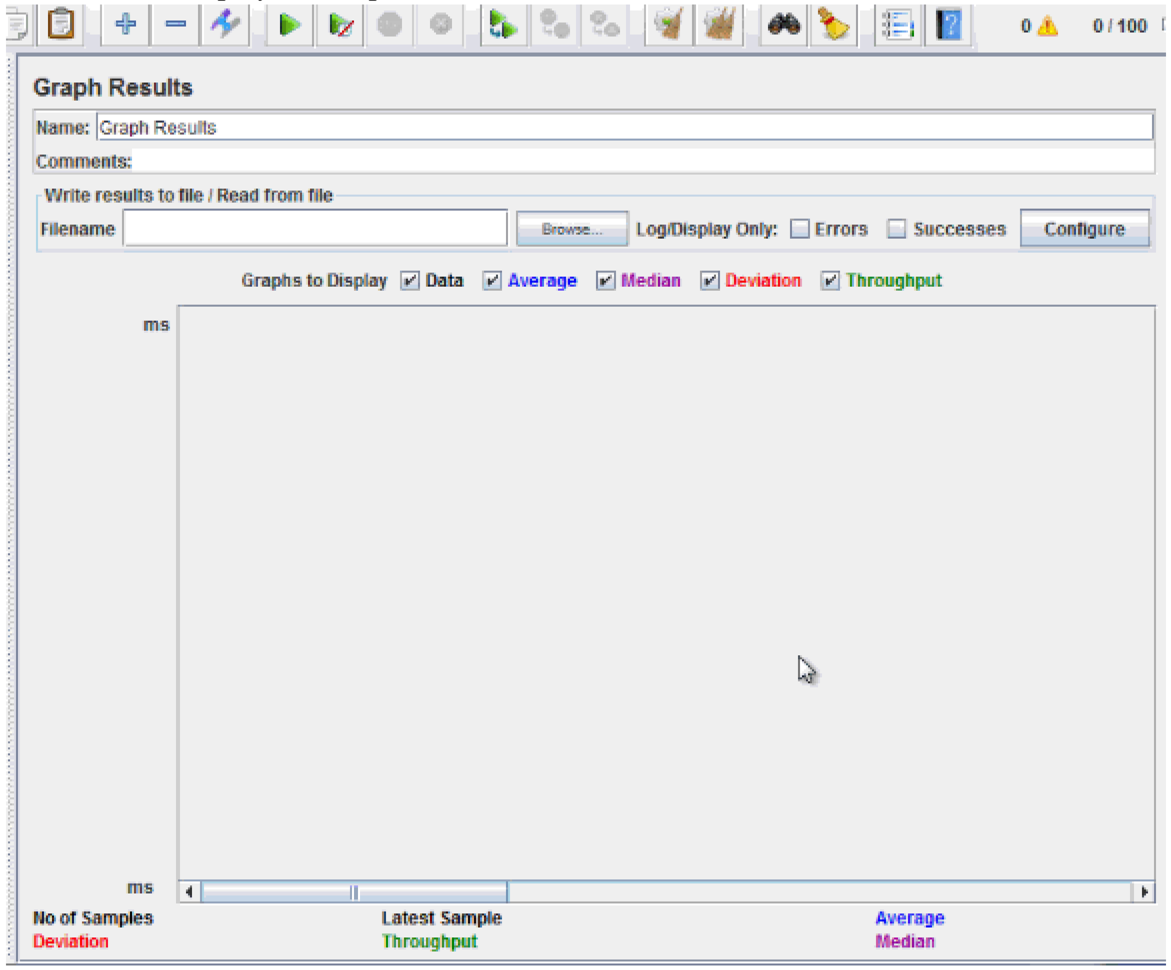
☐ Redirect Automatically ☑ Follov

If you keep the Path field blank JMeter will create the URL request http://www.google.com to Google server.
In this test, you keep the Path field blank to make JMeter create the URL request http://www.google.com to Google server.

**Step 3: Adding Graph result**
JMeter can show the test result in Graph format.
Right click Test Plan, Add -> Listener -> Graph Results



**Step 4: Run Test and get the test result**
Press the Run button (Ctrl + R) on the Toolbar to start the software testing process. You will see the test result display on Graph in the real time.



Graph Results

Name: Graph Results

Comments:

Write results to file / Read from file

Filename

Log/Display Only: ☐ Errors ☐ Successes

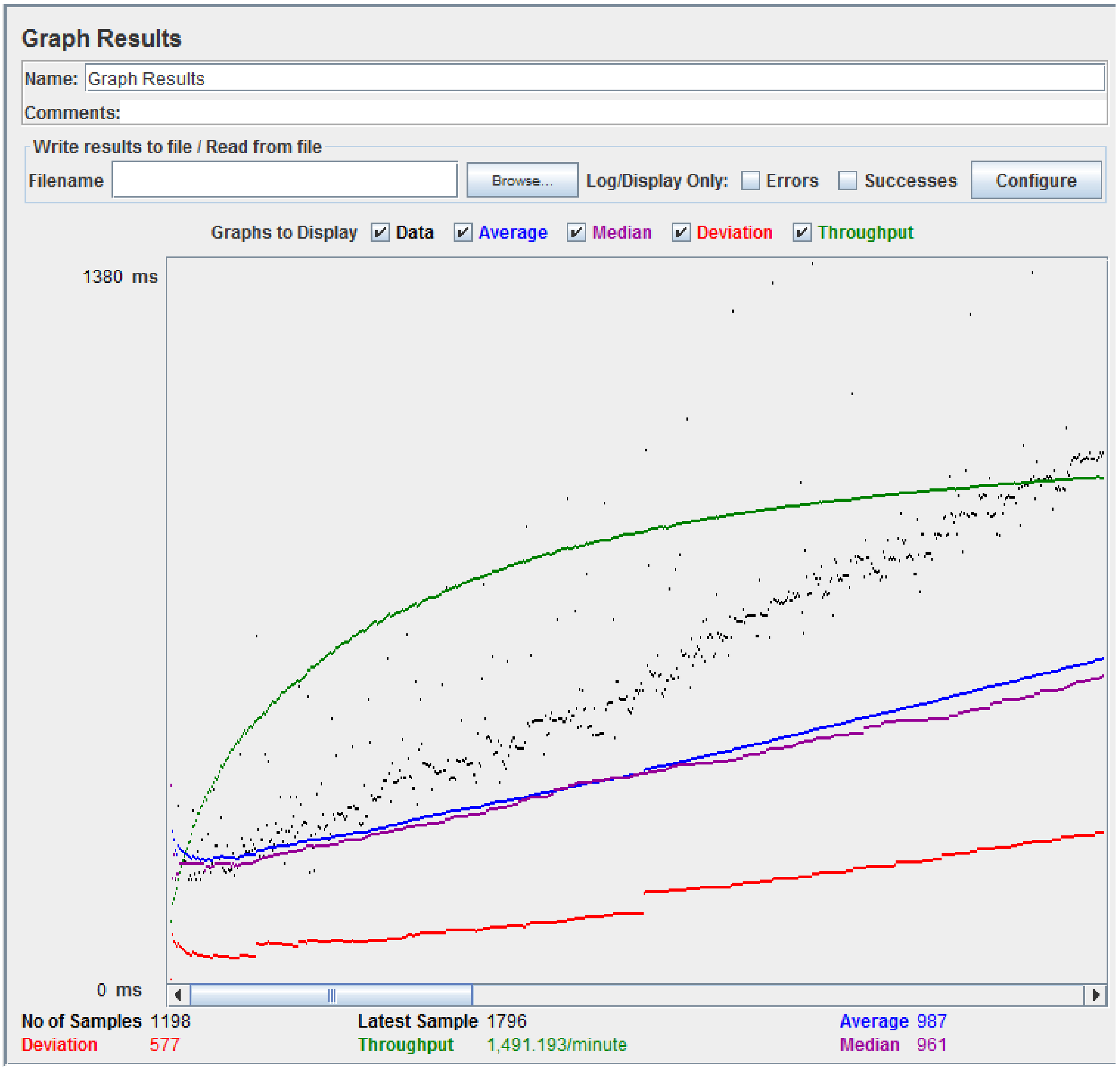
Graphs to Display ✓ Data ✓ Average ✓ Median ✓ Deviation ✓ Throughput

No of Samples
Deviation
Latest Sample
Throughput
Average
Median

At the bottom of the picture, there are the following statistics, represented in colors:

- Black: The total number of current samples sent.
- Blue: The current average of all samples sent.
- Red: The current standard deviation.
- Green: Throughput rate that represents the number of requests per minute the server handled

Let analyze the performance of Google server in below figure.



**Graph Results**

Name: Graph Results

Comments:

Write results to file / Read from file

Filename

Log/Display Only: ☐ Errors ☐ Successes

Graphs to Display ☑ Data ☑ Average ☑ Median ☑ Deviation ☑ Throughput

No of Samples 1198
Deviation 577
Latest Sample 1796
Throughput 1,491.193/minute
Average 987
Median 961

To analyze the performance of the web server under test, you should focus on 2 parameters

- Throughput
- Deviation

The Throughput is the most important parameter. It represents the ability of the server to handle a heavy load. The higher the Throughput is, the better is the server performance.
In this test, the throughput of Google server is 1,491.193/minute. It means Google server can handle 1,491.193 requests per minute. This value is quite high so we can conclude that Google server has good performance
The deviation is shown in red – it indicates the deviation from the average. The smaller the better.