Autobot

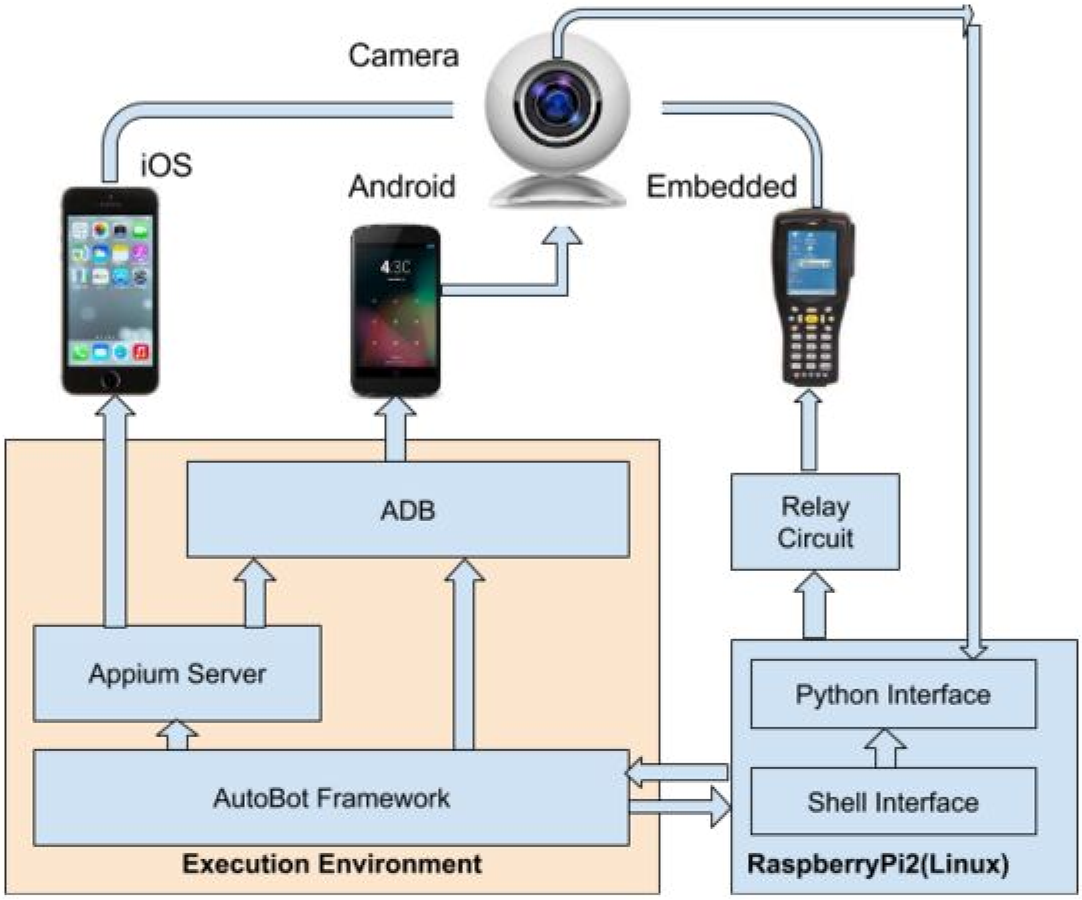
Version 2.0

AFour Technologies

Overview

The framework's goal is inline with the concept of Transformers movie and it promises to work on same idea to provide multiple types of device testing inside one integrated framework and also give power of using/understanding/extending the test framework to anyone from coder to non coder such as business analysts, stakeholders, release managers by use of Cucumber framework layer.

Executing Environment Pre Requisites



With respect to the above diagram the following software will be needed to be installed and setup on the execution environment.

1. Access to GITLAB
2. Jenkins with cucumber plugin integrated to GITLAB Autobot project
3. Java
4. Appium
5. ADB

6. Path Should be set in environment variables
7. Android/iOS Device along with configured to WIFI
8. Embedded devices
9. Webcam
10. Raspberry Pi

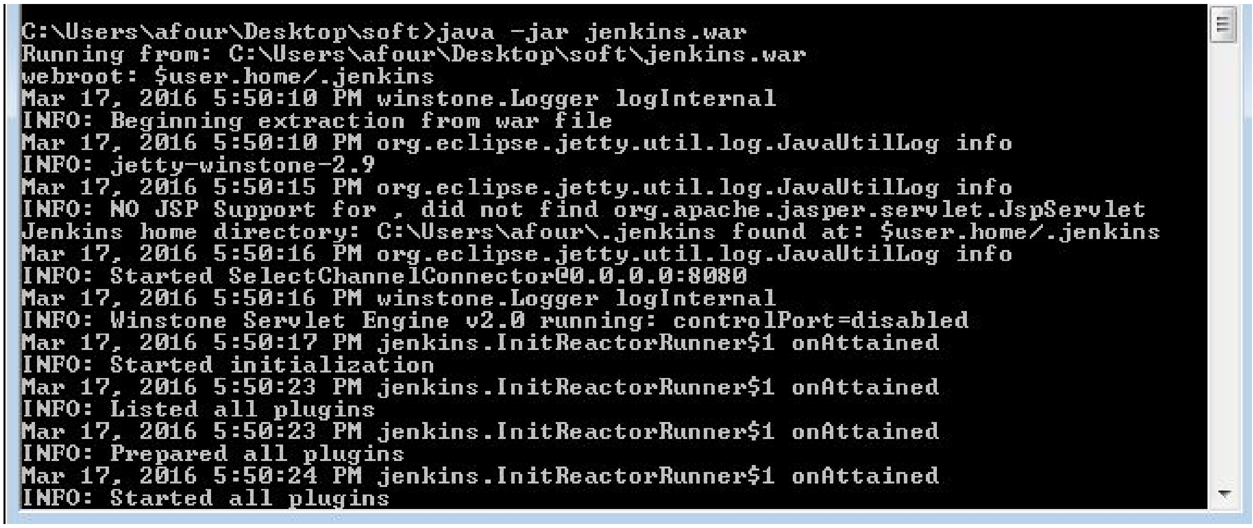
Common Installation Guideline for All Devices.

Access to GITLAB and Sample Project:

User should have access to GITLAB with username and password to url "http://ec2-54-173-216-70.compute-1.amazonaws.com/user/Embedded_OCR_Automation.git"

Linking Jenkins to GITLAB along with Project

- User should have downloaded **jenkins war file** and it should be placed to local drive. Open command prompt and change directory to the location of the jenkins war file.
- Enter the command ***java -jar jenkins.war*** in the command prompt and press enter. Jenkins successful installation message will be displayed.

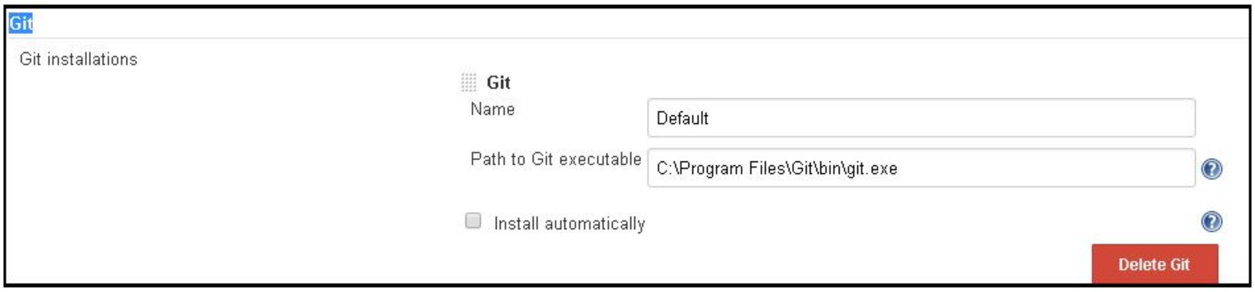


```
C:\Users\afour\Desktop\soft>java -jar jenkins.war
Running from: C:\Users\afour\Desktop\soft\jenkins.war
webroot: $user.home/.jenkins
Mar 17, 2016 5:50:10 PM winstone.Logger logInternal
INFO: Beginning extraction from war file
Mar 17, 2016 5:50:10 PM org.eclipse.jetty.util.log.JavaUtilLog info
INFO: jetty-winstone-2.9
Mar 17, 2016 5:50:15 PM org.eclipse.jetty.util.log.JavaUtilLog info
INFO: NO JSP Support for , did not find org.apache.jasper.servlet.JspServlet
Jenkins home directory: C:\Users\afour\.jenkins found at: $user.home/.jenkins
Mar 17, 2016 5:50:16 PM org.eclipse.jetty.util.log.JavaUtilLog info
INFO: Started SelectChannelConnector@0.0.0.0:8080
Mar 17, 2016 5:50:16 PM winstone.Logger logInternal
INFO: Winstone Servlet Engine v2.0 running: controlPort=disabled
Mar 17, 2016 5:50:17 PM jenkins.InitReactorRunner$1 onAttained
INFO: Started initialization
Mar 17, 2016 5:50:23 PM jenkins.InitReactorRunner$1 onAttained
INFO: Listed all plugins
Mar 17, 2016 5:50:23 PM jenkins.InitReactorRunner$1 onAttained
INFO: Prepared all plugins
Mar 17, 2016 5:50:24 PM jenkins.InitReactorRunner$1 onAttained
INFO: Started all plugins
```

- After that user should navigate to a browser and enter the default port of jenkins **http://localhost:8080**
- Click on **"Manage Jenkins"** on the home page in the browser
- Click on "**Configure System**"
- **GIT:** Fill the fields respectively,

Name:- Default

Path to Git executable:- C:\Program Files\bin\git.exe



Git

Git installations

Git

Name: Default

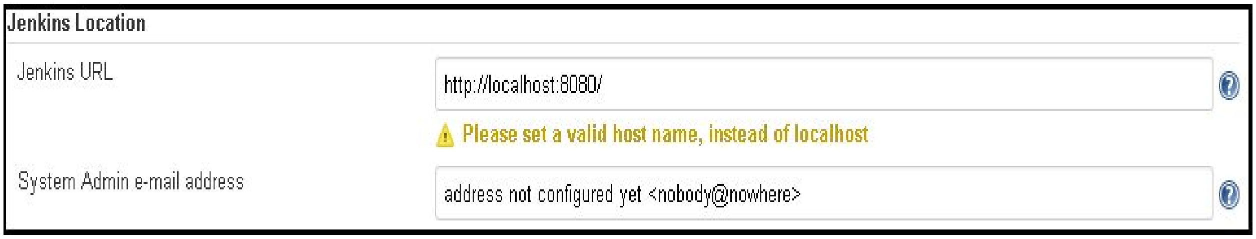
Path to Git executable: C:\Program Files\Git\bin\git.exe

☐ Install automatically

- **Jenkins Location:**

Jenkins URL: http://localhost:8080/

System Admin email address:- address not configured yet.



Jenkins Location

Jenkins URL: http://localhost:8080/

⚠ Please set a valid host name, instead of localhost

System Admin e-mail address: address not configured yet <nobody@nowhere>

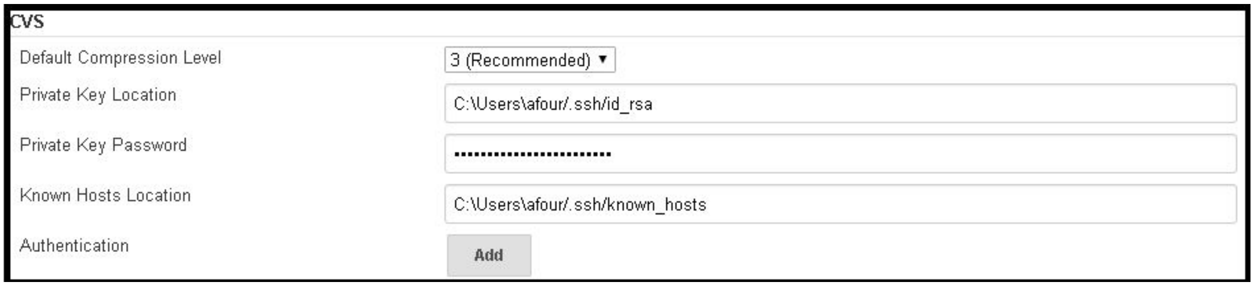
- **CVS:-**

Default Compression Level:-3

Private Key Location: C:\USers\afour/.ssh/id_rsa

Private Key Password:

Known Hosts Location:- C:\Users\afour/.ssh/known_hosts



CVS

Default Compression Level: 3 (Recommended) ▾

Private Key Location: C:\Users\afour/.ssh/id_rsa

Private Key Password: ••••••••••••••••••••••••••

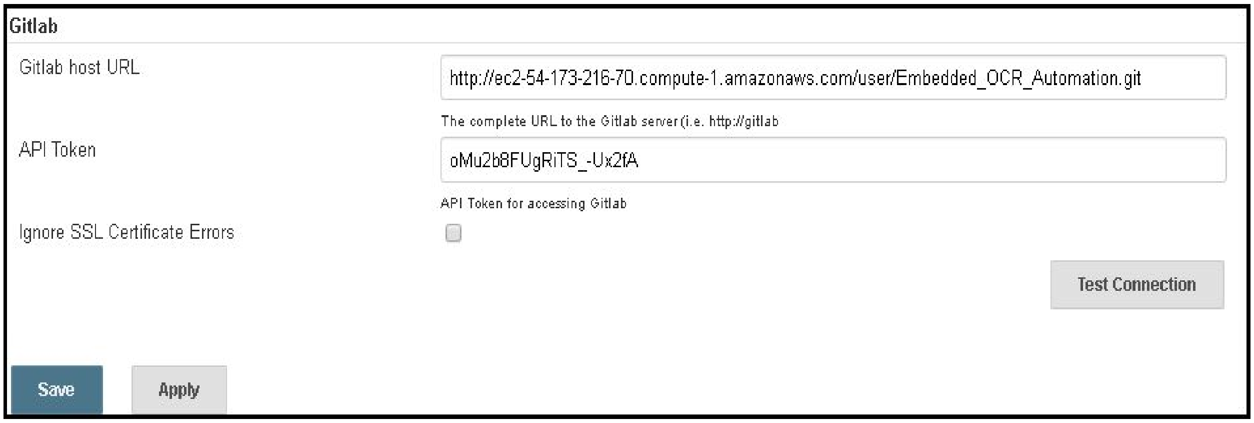
Known Hosts Location: C:\Users\afour/.ssh/known_hosts

Authentication

- **Gitlab:-**

Gitlab host URL :- http://ec2-54-173-216-70.compute-1.amazonaws.com/user/Embedded_OCR_Automation.git

API Token: oMu2b8FUgRiTS_-Ux2fA



Gitlab

Gitlab host URL: http://ec2-54-173-216-70.compute-1.amazonaws.com/user/Embedded_OCR_Automation.git

The complete URL to the Gitlab server (i.e. http://gitlab

API Token: oMu2b8FUgRiTS_-Ux2fA

API Token for accessing Gitlab

Ignore SSL Certificate Errors ☐

- **Click on Apply and then Save.**

Project configuration in jenkins

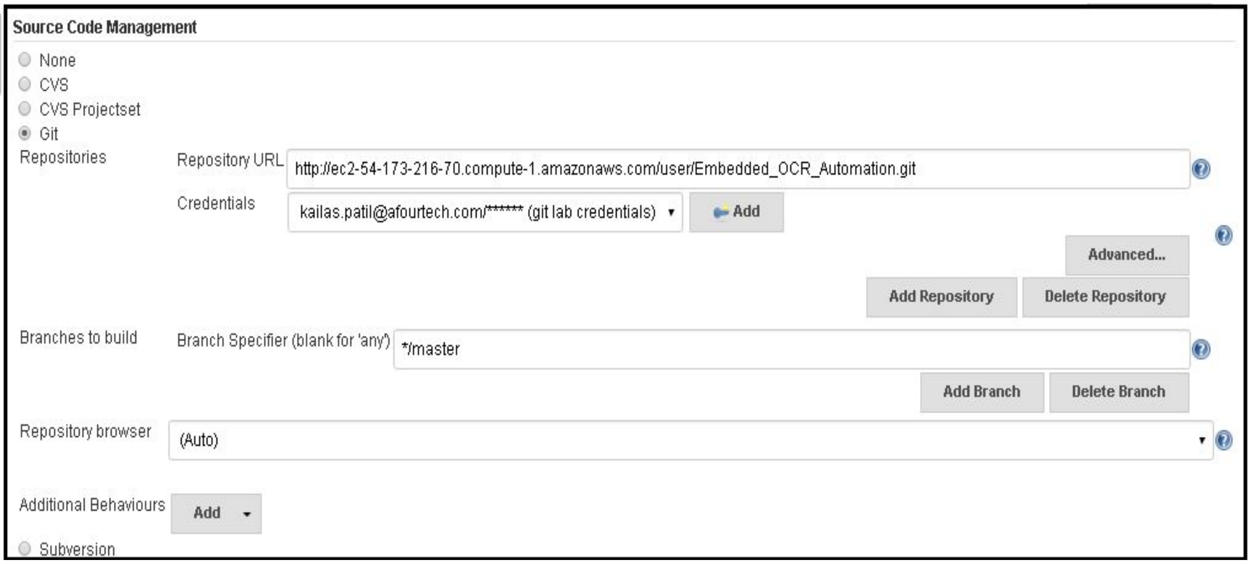
- On Jenkins home page click on “**New Item**” for new project.
- Enter project name in “**Item Name**” field and select “**Free Style”** Project radio button
- Project Configuration page will get displayed.
- **Source Code Management :** Select Git from radio button

Repository URL:

http://ec2-54-173-216-70.compute-1.amazonaws.com/user/Embedded_OCR_Automation.git

Credentials:

Branches to Build : */master



Source Code Management

○ None
○ CVS
○ CVS Projectset
● Git

Repositories

Repository URL: http://ec2-54-173-216-70.compute-1.amazonaws.com/user/Embedded_OCR_Automation.git

Credentials: kailas.patil@afourtech.com/****** (git lab credentials)

Branches to build

Branch Specifier (blank for 'any'): */master

Repository browser: (Auto)

Additional Behaviours

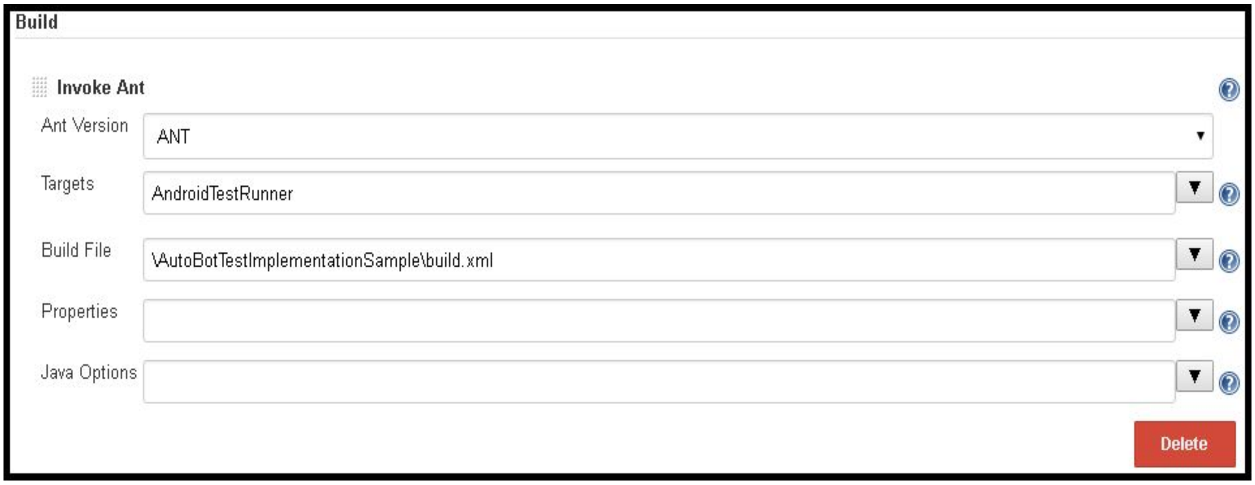
○ Subversion

- **Build:- ** Invoke Ant

Ant Version: Ant

Targets: AndroidTestRunner

Build File: \AutoBotTestImplementationSample\build.xml



Build

Invoke Ant

Ant Version: ANT

Targets: AndroidTestRunner

Build File: \AutoBotTestImplementationSample\build.xml

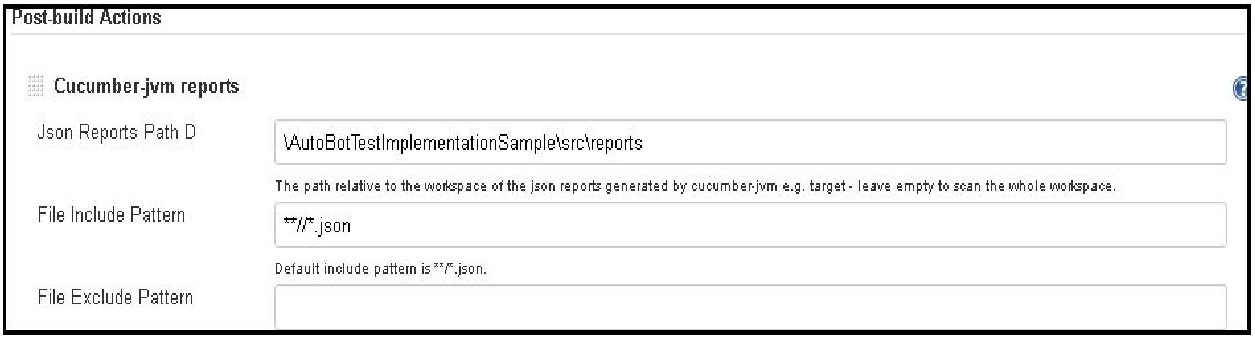
Properties:

Java Options:

- **Post Build Action**

Json Reports Path Drive: \AutobotTestImplementationSample\reports

File Include Pattern: **//* .json



Post-build Actions

Cucumber-jvm reports

Json Reports Path D: \AutoBotTestImplementationSample\src\reports

The path relative to the workspace of the json reports generated by cucumber-jvm e.g. target - leave empty to scan the whole workspace.

File Include Pattern: **//*.json

Default include pattern is **/*.json.

File Exclude Pattern:

- Click on Save and Click on Apply
-

Path Should be set in environment variables

- User should navigate to right click My Computer and select **Properties**.
- On the left side click on **Advanced System Settings**
- Click on **Environment variables**
- Create a new variable and name it ANDROID_HOME and give it path of sdk **“Drive:\Android\sdk”**
- Create a new variable and name it ANT_HOME and give it path of ant **“Drive:\apache-ant-1.9.4\”**
- Create a new variable and name it JAVA_HOME and give it path jdk **“Drive:\Program Files\Java\jdk1.8.0_66”**
- Create a new variable and name it JRE_HOME and give it path jdk **“Drive:\Program Files\Java\jre1.8.0_66;”**
- Edit the Path Environment variable and add path for TortoiseGit“**Drive:\Program Files\TortoiseGit\bin;”**
- Edit the Path Environment variable and add path for jdk ” **Drive:\Program Files\Java\jdk1.8.0_73\bin;**”
- Edit the Path Environment variable and add path for Android sdk tools and Android sdk platform tools “**Drive:\project\Android\sdk\tools\;C:\project\Android\sdk\platform-tools\;**”
- Edit the Path Environment variable and add path for Ant Home “**Drive:\Program Files\Java\jre1.8.0_73;%ANT_HOME%\bin;”**”

Installation Guidelines for Automation on iOS

Java Installation

- Download and Install latest version of java from **https://java.com/en/download/manual.jsp**

Appium Installation

- Json Navigate to site **http://appium.io/**
- Click on Download link and install on the machine

iOS Device configured to WIFI

- Android devices should have been connected to WIFI
- iOS devices should be connected to wifi

Installation Guidelines for Automation on Android

Java Installation

- Download and Install latest version of java from **https://java.com/en/download/manual.jsp**

ADB Installation

- Android SDK folder which is required for ADB will be kept ready to download from a repository rather than downloading a 2.5 GB setup which installs upto 5GB
- User can either download the file from http://developer.android.com/sdk/index.html

Android Device configured to WIFI

- Android devices should have been connected to WIFI
- iOS devices should be connected to wifi

Installation Guidelines for Automation for Embedded

Raspberry Pi

- Android Web Camera connected to raspberry pi will be asked by framework
- Raspberry Pi camera image can also be utilised for iOS/android device as the image capture step in framework will be device platform independent.

- OCR(Optical Character Recognition) is used by all devices for verification and validation.
- **Setting up Raspberry Pi**

1. Install fswebcam module

2. Install & configure motion module(add to init.d for start up trigger)

3. Set up motion module(http://www.instructables.com/id/Raspberry-Pi-as-low-cost-HD-surveillance-camera/step7/Installing-the-motion-detection-software/)

4. Copy Files 'imageCapture.sh', 'keypressTrigger.sh', 'keypressmultiple.py' to home directory

5. Install python module Rpi.GPIO

Deployment Schedule and Resources

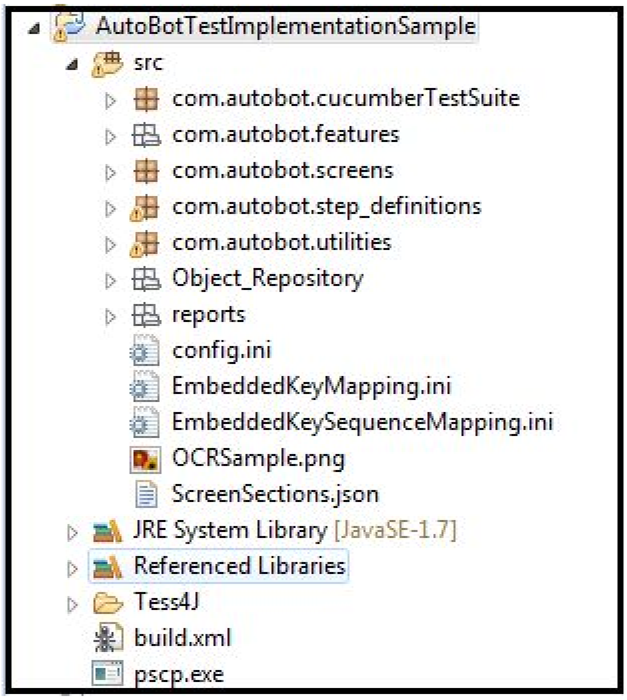
- Latest Releases would be indicated via updates to Autobot 3.0
- Also those will be incorporated in GITLAB to download and use accordingly

Technology Requirements and Support Considerations

- Windows: Would require a 64 bit windows machine with RAM 8GB or higher, 320 GB hard disk.
- iOs Device: Running iOS 9
- Embedded Device
- Android Device: Running KitKat 4.4.2

Execution with Autobot

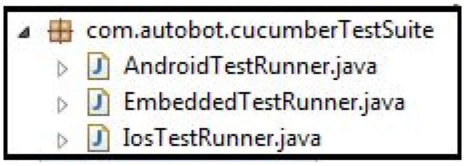
- Gilab access would have a sample project with name ***"AutoBotTestImplementationSample".***
- In this project the user would have the below structure available when opened in an integrated development environment like Eclipse. You can download the latest version from http://www.eclipse.org/epsilon/download/
- Also user would need to install other software in eclipse as ***Cucumber, Testng, Junit***.
- Once the project is opened in eclipse it would look like this in the package explorer



```
AutoBotTestImplementationSample
  src
    com.autobot.cucumberTestSuite
    com.autobot.features
    com.autobot.screens
    com.autobot.step_definitions
    com.autobot.utilities
    Object_Repository
    reports
    config.ini
    EmbeddedKeyMapping.ini
    EmbeddedKeySequenceMapping.ini
    OCRSample.png
    ScreenSections.json
  JRE System Library [JavaSE-1.7]
  Referenced Libraries
  Tess4J
  build.xml
  pscp.exe
```

- Once Under the source you will find numerous packages, which are elaborated below.

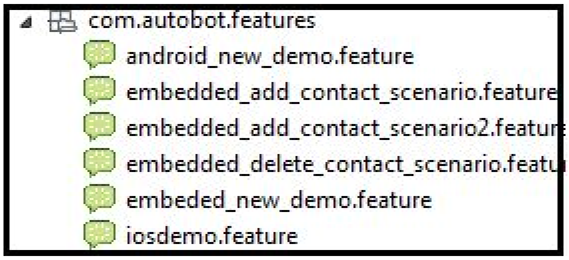
Com.autobot.cucumberTestSuite



```
com.autobot.cucumberTestSuite
  AndroidTestRunner.java
  EmbeddedTestRunner.java
  IosTestRunner.java
```

- The package has java class files. Each of which represent a runnable class. You can choose to run your tests relating to the platform you're working upon.
- User would have to mention ***Cucumber Options such as format, options and glue.***
- ***To execute the project, user would have to right click the respective runner file and click on run as JUnit Test.***

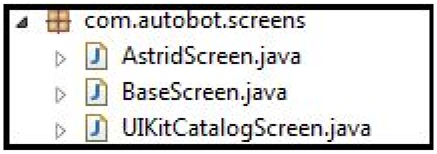
Com.autobot.features



```
com.autobot.features
    android_new_demo.feature
    embedded_add_contact_scenario.feature
    embedded_add_contact_scenario2.featur
    embedded_delete_contact_scenario.featu
    embeded_new_demo.feature
    iosdemo.feature
```

- The package has **.feature** files.
- User can create a Test scripts with **gherkin** keywords for BDD purpose and have them in cucumber syntax in their respective feature files.

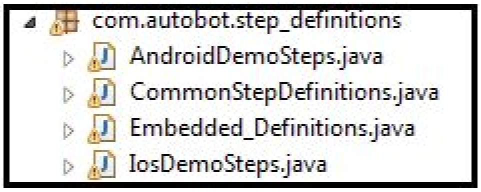
Com.autobot.screens



```
com.autobot.screens
    AstridScreen.java
    BaseScreen.java
    UIKitCatalogScreen.java
```

- The package has java class files, which represent page object model in short.
- Each class represent a particular screen
- The action relating to the all the elements on the screen are present in their respective classes.

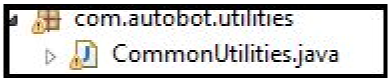
Com.autobot.step_definitions



```
com.autobot.step_definitions
    AndroidDemoSteps.java
    CommonStepDefinitions.java
    Embedded_Definitions.java
    IosDemoSteps.java
```

- The package has java class files, which are detail code steps to the gherkin keywords written in your .feature files for corresponding platforms.
- The implementation of the Autobot jar comes in assistance here.

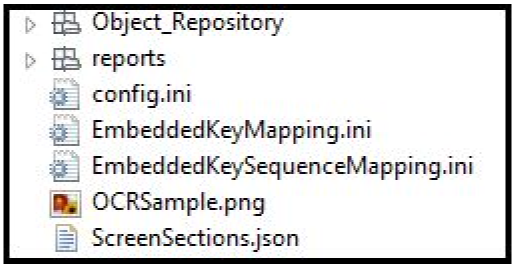
Com.autobot.utilities



```
com.autobot.utilities
    CommonUtilities.java
```

- The package has java class files, which contain properties for rootpath, mapping file and config file, json file required for the project.

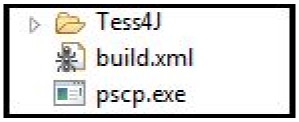
Object_Repository and Reports



Object_Repository
reports
config.ini
EmbeddedKeyMapping.ini
EmbeddedKeySequenceMapping.ini
OCRSample.png
ScreenSections.json

- The Object Repository package has .xml files screen wise.
- Each of which contains element names and their properties.
- Reports package has .json files which are auto generated when the project has run. This property is set in TestRunner class file.
- The config.ini file has property set that for which platform the execution is going to be carried out
- The EmbeddedKeyMapping files have mapping relating to embedded devices.

Tess4j

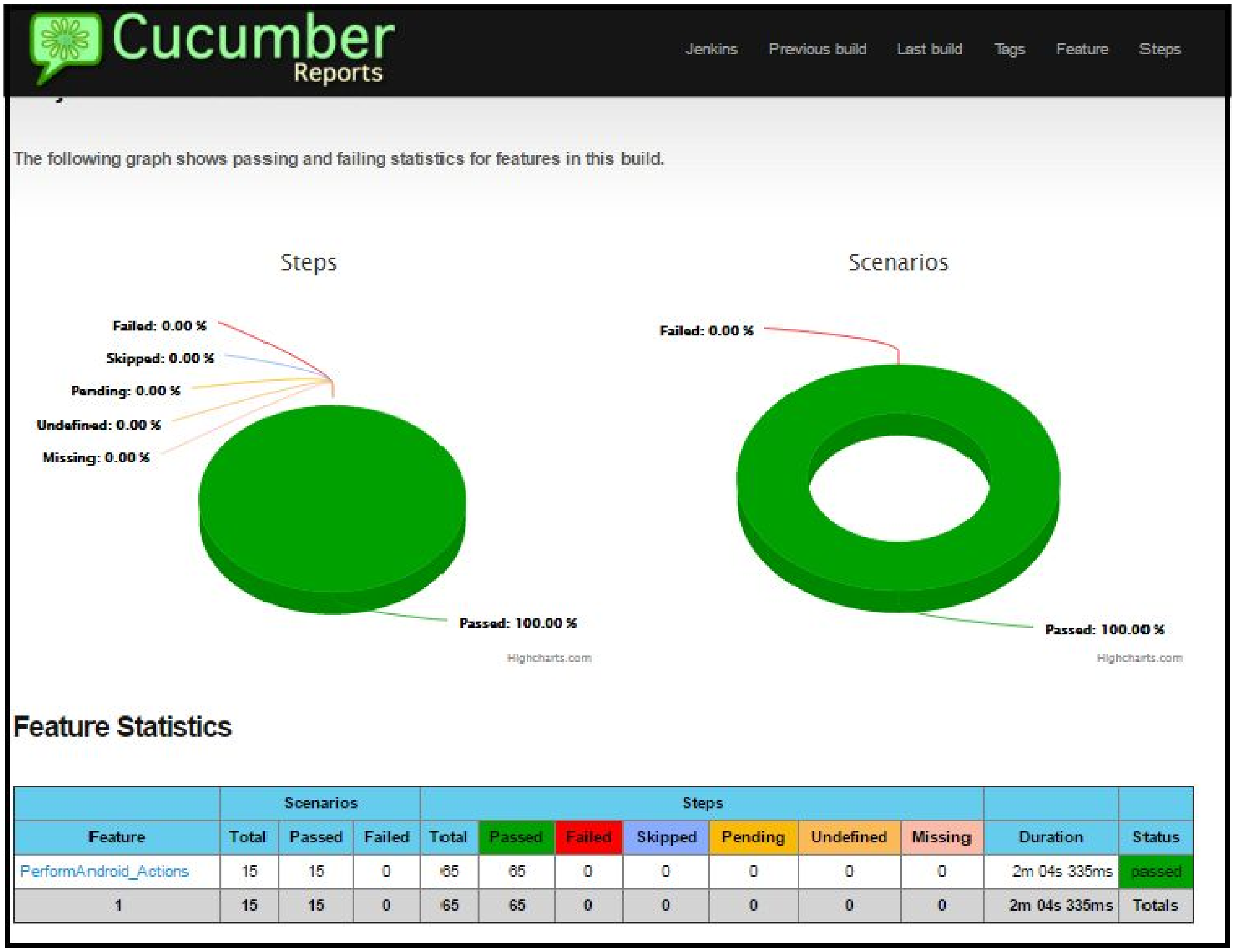


Tess4J
build.xml
pscp.exe

- This is open source OCR engine.
- This will help us verify text off a image to the text that is passed as parameter or otherwise.

Cucumber Reports

- Once the project is configured in jenkins, you can build/execute the project from there.
- Cucumber plugin installed in jenkins gives us a report of the execution carried out.
- User can click on the test results to see which is passed
- If a test fails, user can go to the step in test result to find which step failed.



The following graph shows passing and failing statistics for features in this build.

Feature Statistics

	Scenarios			Steps								
Feature	Total	Passed	Failed	Total	Passed	Failed	Skipped	Pending	Undefined	Missing	Duration	Status
PerformAndroid_Actions	15	15	0	65	65	0	0	0	0	0	2m 04s 335ms	passed
1	15	15	0	65	65	0	0	0	0	0	2m 04s 335ms	Totals

Risks and Known Issues

- Implementation for iOS/Android has not been completely carried out. Methods such as drag and drop, pinch zoom in and zoom out are missing.

Accepted By

- AFour Technologies.