EVERYMOBILE – USER MANUAL – IOS

Contact:

sales@testonneed.com

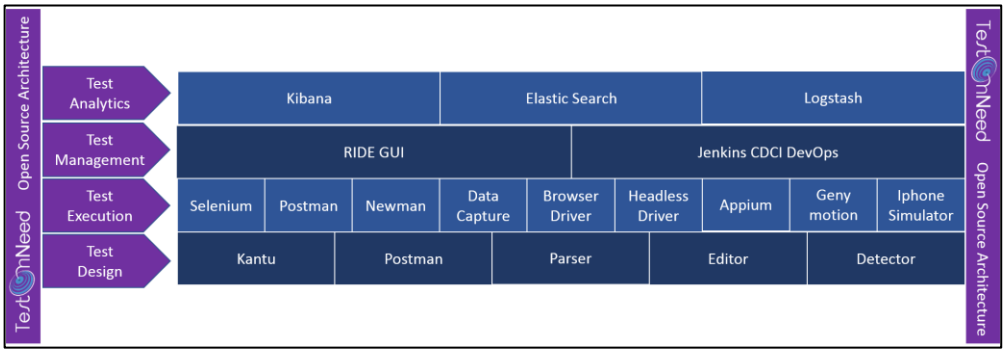
TABLE OF CONTENTS

1. Introduction 2
2. What We Do 3
3. Prerequisites 4
4. Description Of The Website 5
5. EveryMobile Page 7
5.1 Develop Test Case 8
5.2 Prepare Test Case 11
5.3 Execute Test Case 13
5.3.1 Via TestOps – Functional 14
5.3.2 Via TestOps – Automation 17
5.4 Analyze Results 20
6. References 22
6.1 Creation Of New View 22
6.2 Starting Appium 24
6.3 Real Device and Simulator 27
6.3.1 Connecting Simulator 27
6.3.2 Connecting Real Device 29
6.4 Execution 36

1. INTRODUCTION

TestOnNeed, an Open Source Testing Ecosystem, is created for Entrepreneurs, Start-ups and Enterprises to **test** software products and solutions **'on need'**. We have helped global startups, mid to large enterprises to deliver world-class products and solutions with strategic insights to transform and thrive in this rapidly changing world.

- TestOnNeed Open Source Ecosystem is used in the testing of the frontend GUI, backend API calls, mobile application (android and iOS) functionalities.



- This Open Source Ecosystem consists of **Test Design, Test Execution, Test Management and Test Analytics**.
- Each stage of Open Source Ecosystem uses set of open sources that are useful in their own way to achieve product quality.
- **Test Design**
  - In this step, the development or creating of the test cases are done.
  - Kantu, Postman, Parser applications, Editor and Detector are the open sources that are being used.
- **Test Execution**
  - In the step, the execution of the developed or created test cases take place.
  - Selenium, Postman, Newman, Data Capture, Browser Driver, Headless Driver, Appium, Genymotion and iPhone Simulator are the different open sources that are being used.
- **Test Management**
  - In this step, the test cases are given and modified the arguments and parameters that it produces desired result of the test cases.
  - RIDE GUI and Jenkins are the open sources that are being in this step.
- **Test Analytics**
  - Used in the process of analysis of data.
  - Kibana, Elastic Search and Logstash are the open sources that are used in this step.

2. WHAT WE DO

- We help the user to perform an automated testing process that will simplify the process of testing.
- We give the user the benefits of Test Automation, Mobile Testing and DevOps.
- We help to perform the Android and iOS mobile application testing.
- We perform the testing within the allotted time given by the customer.
- We help to attain success by providing expert advice, using open source testing tools augmented by highly skilled software testing resources.

3. PREREQUISITES

The following Open Sources have to be installed in order to run the application. For installation process, please refer Installation Manual.

- **Kantu**

  This is used to record the actions performed by user on the Web Browser, thus automating the manual actions and creating the automation test cases.

- **Postman**

  This is used to record and automate the process of posting requests to server and fetching the corresponding responses back while the user will be executing the actions.

- **RIDE**

  It is used to manage the execution of test cases. The user can select one or multiple test cases, provide various parameters for automation execution.

- **Kibana**

  It is used to analyze and display the data in the form of vertical graphs and pie charts.

- **Jenkins**

  Jenkins is a popular open source tool to perform continuous integration and build automation.

- **Selenium**

  It is used to automate browser to execute web UI test cases.

- **Browser driver**

  It is used to emulate the user actions on Web Browser UI for executing the automation tests in browsers like Chrome, Firefox, IE etc.

- **Appium**

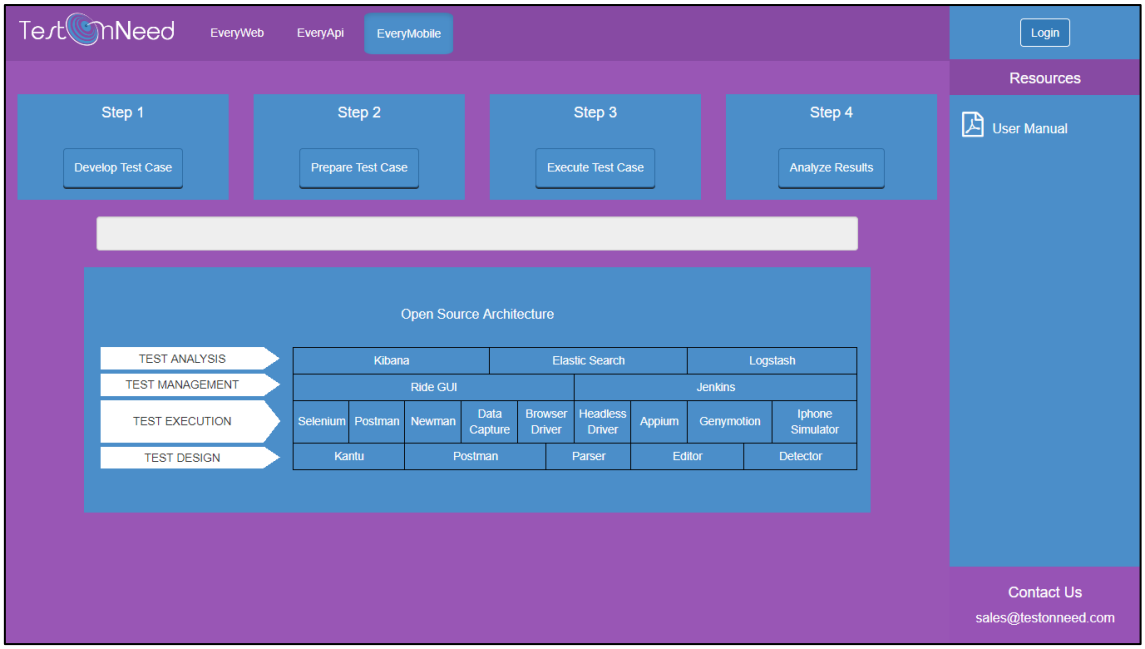
  Appium is an automation tool for running scripts and testing native applications and mobile-web applications on android or iOS using a web driver.

- **Emulator**

  An emulator enables the host system to run software or use peripheral devices designed for the guest system.

4. DESCRIPTION OF THE WEBSITE

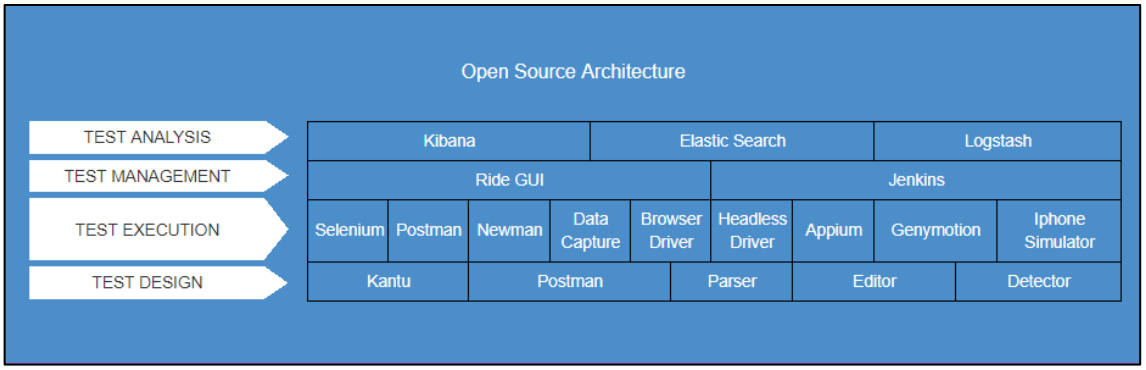
- In order to start the project, there are certain steps to be followed by the user. To know how to start the project, please refer **Quick Start Guide**.
- The outline of the Website looks like below:



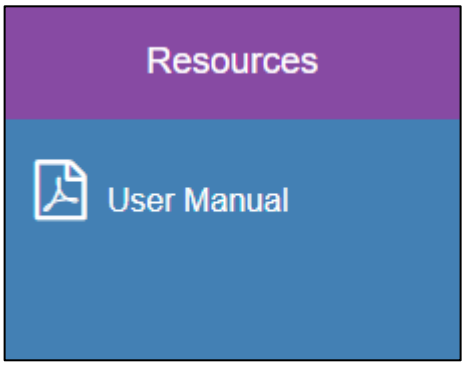
- The application has three tabs namely – EveryWeb, EveryAPI and EveryMobile.

- Things that are common to all the tabs are the stack which contains all the open sources, a status bar and the user manual.
- In the status bar, the user will get know what is being done next by clicking any button.

- The stack of open sources will contain all the applications being used. On mouse hover on any open source, a pop-up window comes up with a brief description of what it is and how it is being used.



- The user manual guides the user through the web application. It will also change from tab to tab since working of each tab is different. It is a hyperlink, on clicking it, will open the respective document in pdf format in a new tab in the browser.



5. EVERYMOBILE PAGE

EveryMobile is a web application testing ecosystem for mobile using open sources. It is used to develop and execute the test cases for the applications using both Android and iOS.

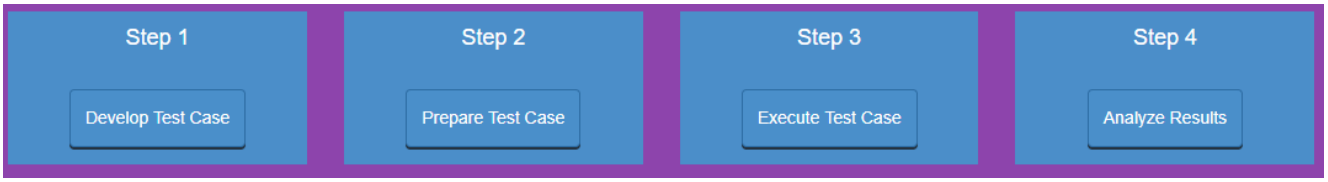
The **Prerequisites of EveryMobile** are the following:

» The user should know Appium scripting.

» The basic of Android (Appium Automation Scripting) and iOS (XC Test Scripting).

» The user should have AUT signed Developer signed account.

- The EveryMobile tab is the third tab of this application.
- This tab has 4 buttons on the top namely –
    1. Develop Test Case
    2. Prepare Test Case
    3. Execute Test Case
    4. Analyze Results.



- **Develop Test Case**

» The user can write a test case in the editor and execute it in the upcoming steps.

- **Prepare Test Case**

» In the background, the downloaded JSON file gets converted to its corresponding Python file.

- **Execute Test Case**

» The user can execute the automated test cases that was recorded in the first step.

- **Analyze Results**

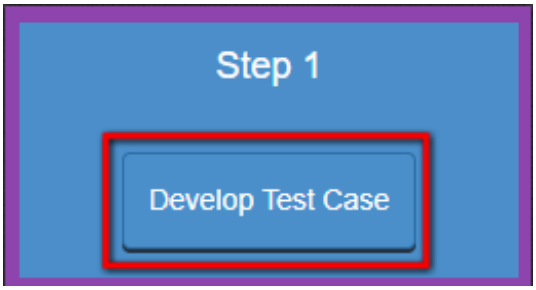
» The user can analyze the results visually with the help of graphs.

- Each step is explained below

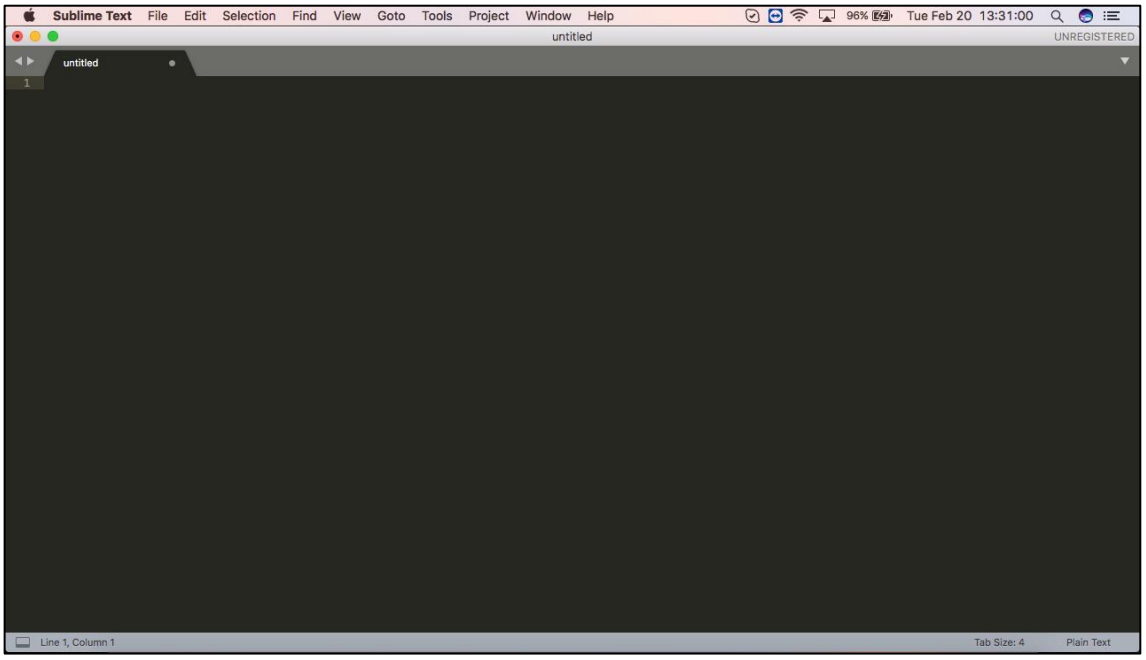
5.1 DEVELOP TEST CASE

The Develop Test Case button when clicked will open Sublime Text for the user. This can be used to write the test scripts for Mobile Application automation by following the guidelines mentioned in the appendix

1. Click the **Develop Test Case**.

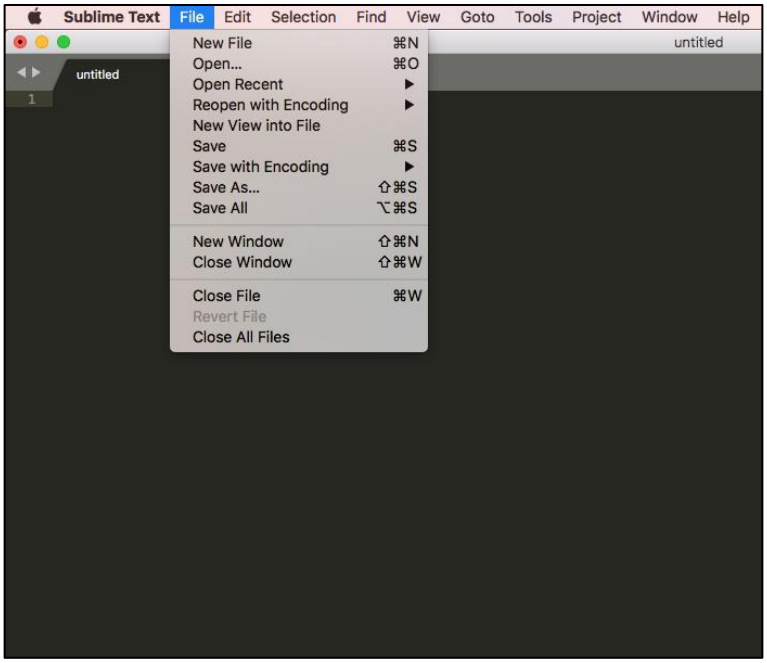


2. On clicking the **Develop Test Case** button, Sublime Text opens up.

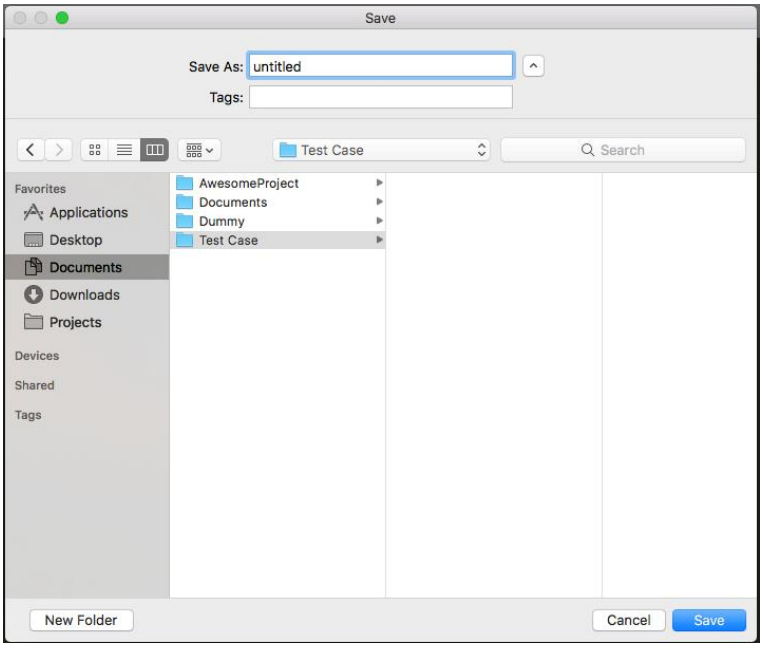


NOTE: The installation process, please refer the installation manual.

3. Write the Appium code for executing the test case. For writing the code, please refer in the Reference section.
4. After writing the code, click Save button.

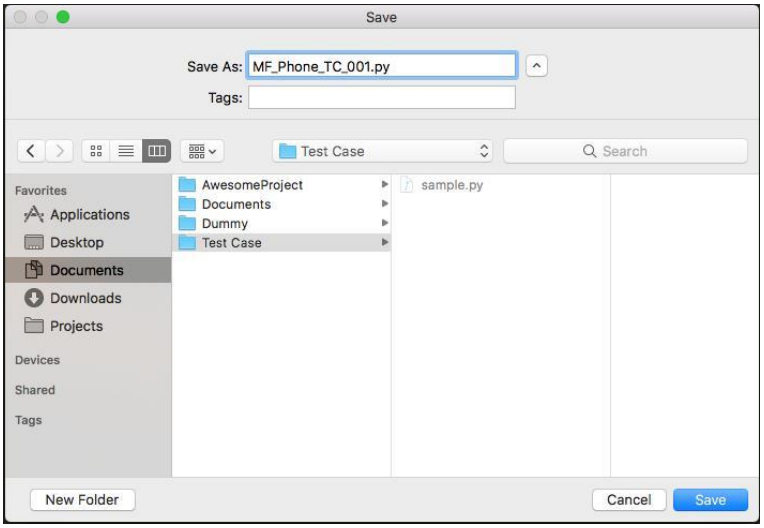


5. Surf to the location where file has to be stored. Example shows folder location as Documents > Test Case.
6. In the **Save as** textbox, give the name of the file according to the naming conventions mentioned below with the file extension.

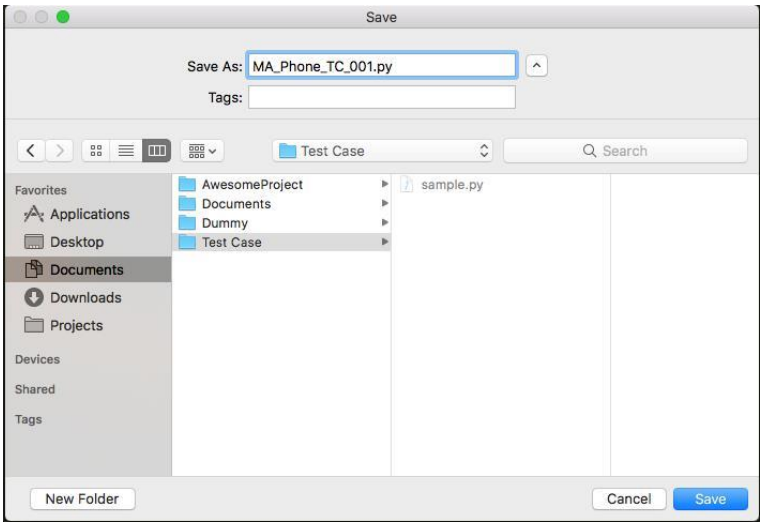


7. Click Save to save the file.
8. Now save the file based on the type of testing functionality. There are 2 types of testing functionality– **Function and Automation.**

9. In the case of **Function**, the test case should be saved as **MF_<TestCase_Name>**. For example, the test case should be saved as **MF_Phone_Calc_TC_001.py.**



10. In the case of **Automation**, the test case should be saved as **MA_<TestCase_Name>**. For example, the test case should be saved as **MA_Phone_Settings_TC_001.py.**

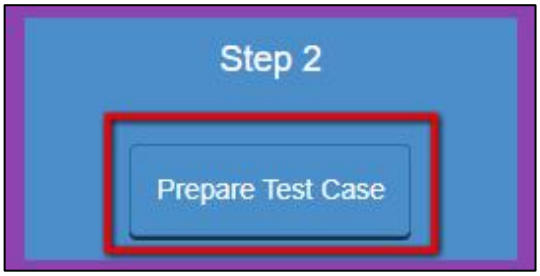


5.2 PREPARE TEST CASE

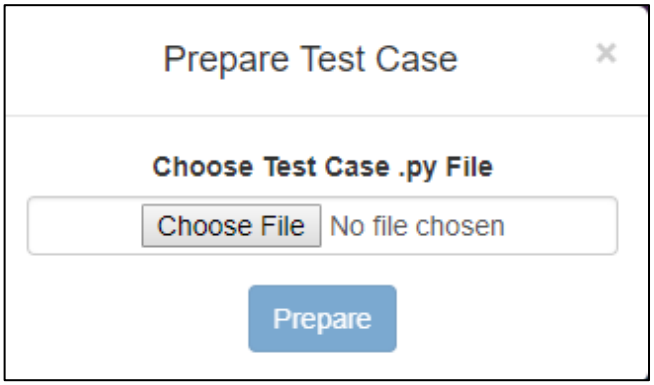
The Prepare Test Case button is used to convert the file uploaded into Python file and save in the auto created folder. In the folder, the user will be able to find the uploaded Python file complied Python file and .xml file.

The user has to keep in mind that the test case that was created for the real device should be uploaded in case of testing using real device and the test case that was created for the simulator should be uploaded in case of testing using simulator. If the user does vice versa, then test case will not be executed.

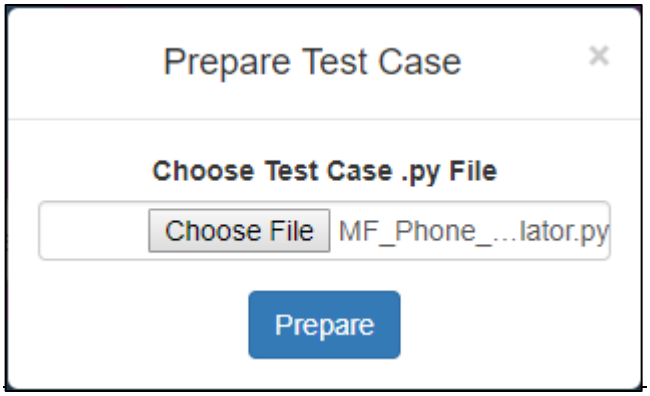
1. Click the **Prepare Test Case** button.



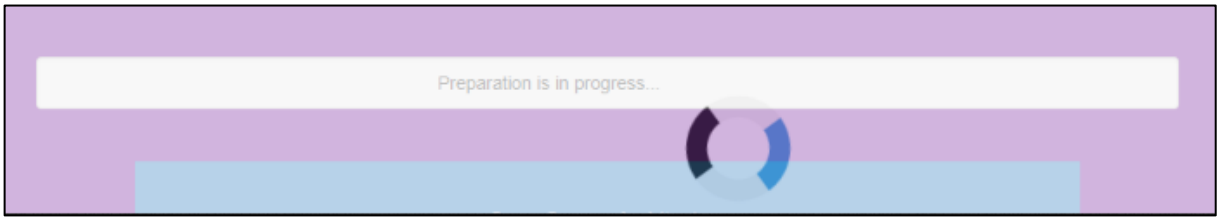
2. On clicking on the button, a pop up appears.



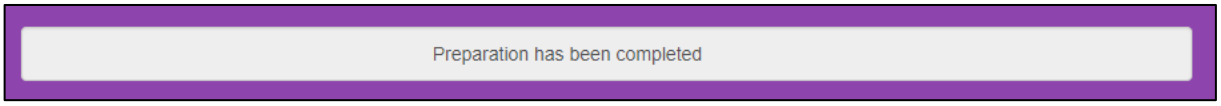
3. Click on the **Choose File** button.
4. Surfing the folder location where saved the Python file that was saved in the previous step.
5. Now upload the Python file.
6. The user can upload **only one file** at a time.
7. On uploading the Python file, the **Prepare** button becomes active.



8. On uploading JSON file and clicking the **Prepare** button, the following will happen at the backend.
    - It creates a folder in the name of the test case that was mentioned in the previous step.
    - The folder gets stored in:
      > TON > MobileTesting > GUI > Demo_TON > <TestCase_Name_Folder>
    - In case if the test case name was saved as **MF_Calc _001** in the previous step then it will create a folder with same name **MF_ Calc _001.**
    - In this folder, the Python file that was uploaded will be found.
9. On clicking of the **Prepare** button, a message will be shown as **"Preparation is in Progress"** in the status bar. And until the preparation is complete, a spinner will be loading stating that the preparation is still on.



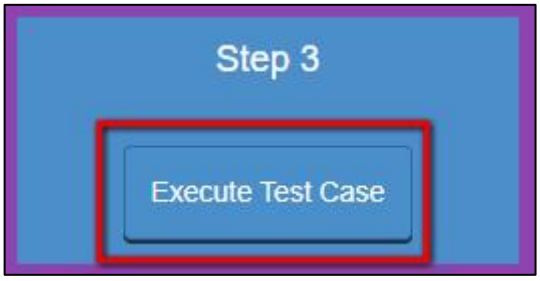
10. Once preparation is finished, a message will be shown as **"Preparation has been completed"** in the status bar.



5.3 EXECUTE TEST CASE

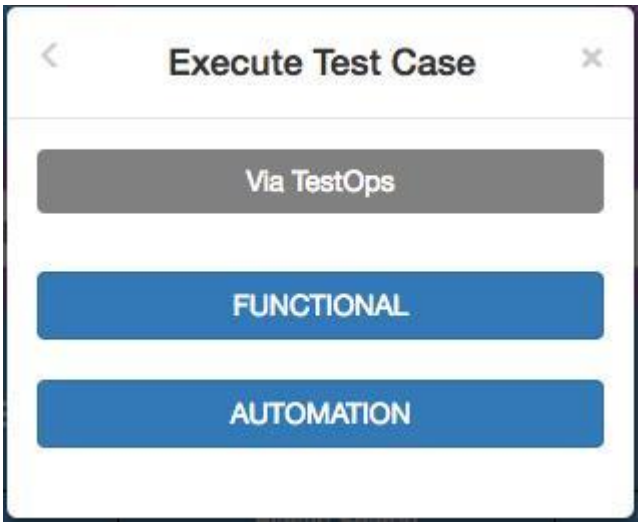
The Execute Test Case button is used for the execution process. The execution process takes place using TestOps. For the execution of the test cases of mobile, the user need to install XCode and Appium in the local system.

1. Click on the **Execute Test Case** button.



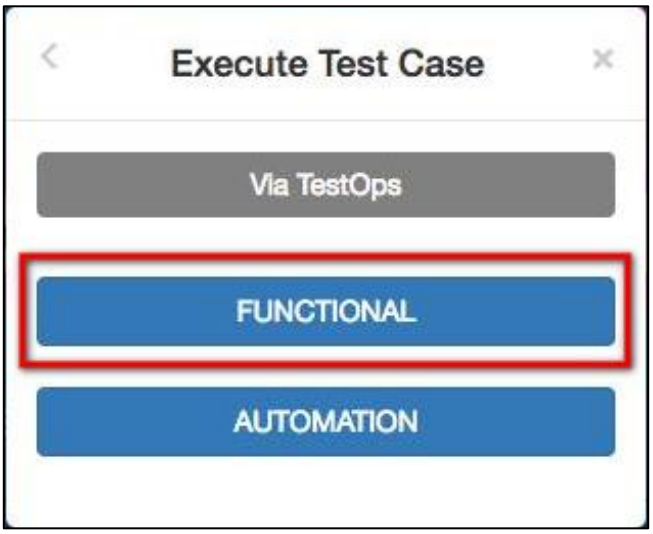
2. On clicking on the button, a pop up appears.
3. From the pop-up, select the button **Via TESTOPS.**

4. On clicking the button, 2 buttons will appear – **Functional and Automation**.

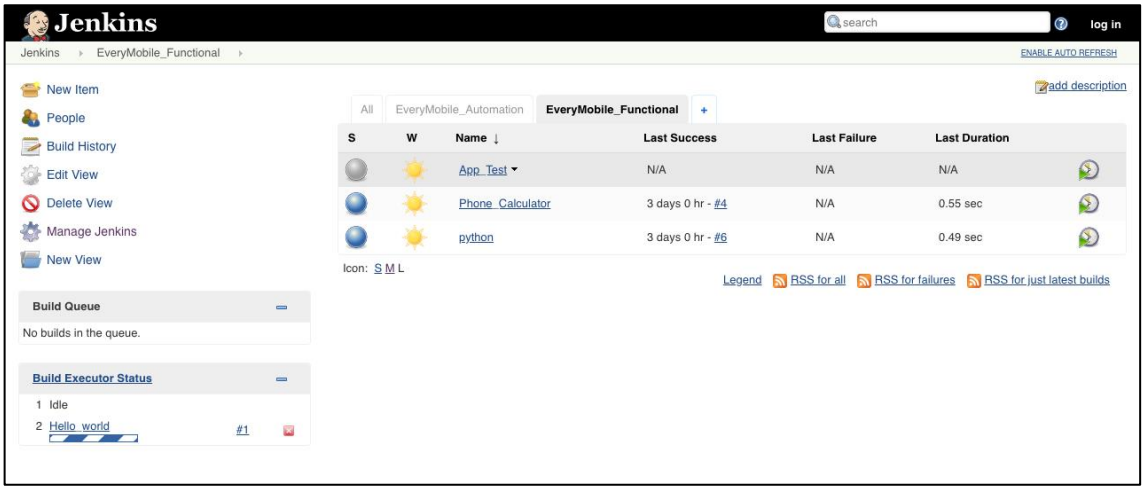


5.3.1 VIA TESTOPS – FUNCTIONAL

1. The user has to start the Appium in order to execute the test case. To know how to start the Appium, please refer to the reference section.
2. Click on the **Functional** button.



3. On click of the button, the **Jenkins** will open.

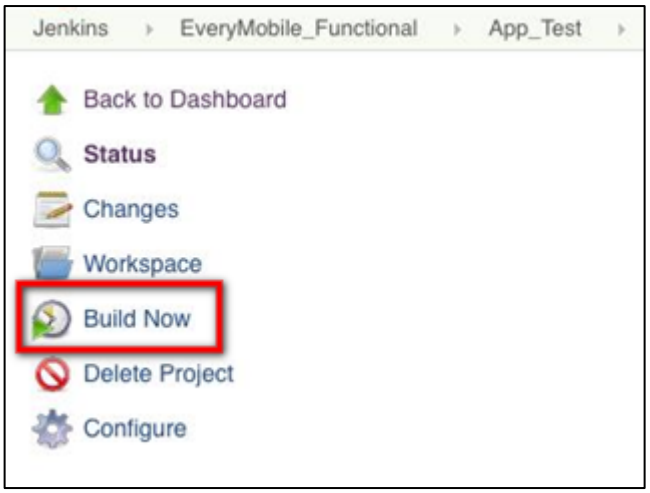


For creation of new View, refer the Reference Section below.

4. As the Jenkins open, the user will be able to see the test case that was created in the previous steps as a new job in the Jenkins.
5. The user has to perform certain steps to execute the test case before the job is being build. Click here to see the steps.

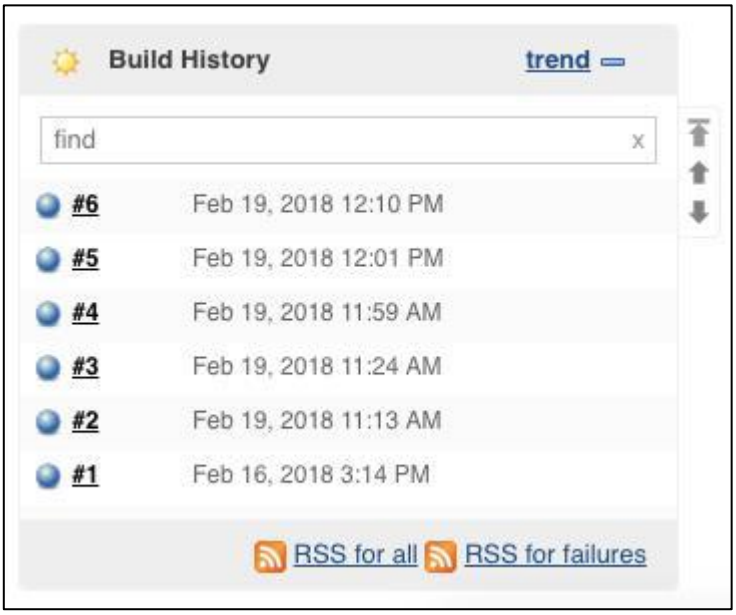
6. Click on the job that has to be executed.

7. On the left side, select Build Now.



8. As the user clicks **Build Now**, the simulator will open if the user has written code for simulator or it will execute in the phone is the code is written for the real device.
9. On the left side, under Build History, the progress of the job is shown.

10. For the first time, while executing the test case, if the WebDriverAgent is not installed already then, the WebDriverAgent will get installed automatically.
11. If the job is successfully built, it will show the built time, number of times of build and status of the built.

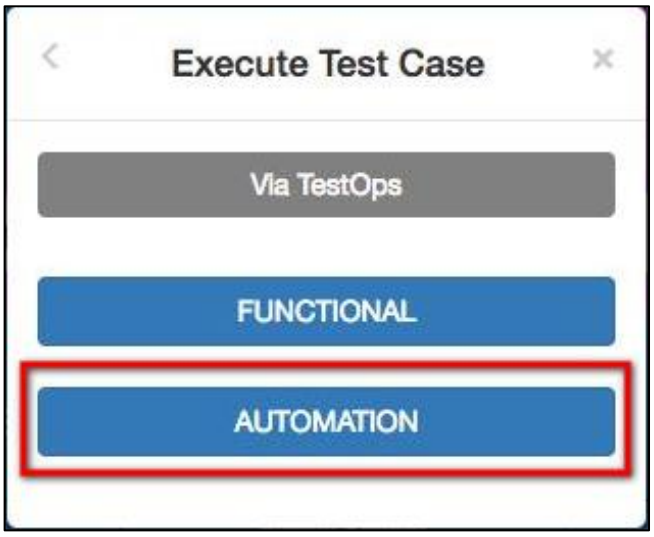


12. All the jobs of the Jenkins will be saved in:

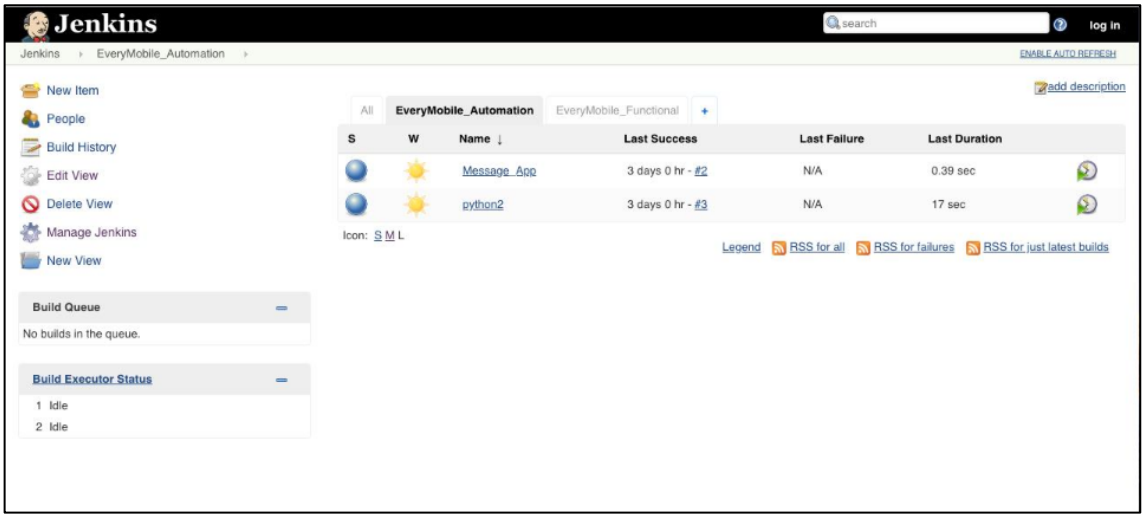
> TON > MobileTesting > TestOps > Demo_TON > <TestCase_Name_Folder> > <Job_Name_Folder>

5.3.2 VIA TESTOPS – AUTOMATION

1. The user has to start the Appium in order to execute the test case. To know how to start the Appium, please refer to the reference section.
2. Click on the **Automation** button.



3. On click of any one button, the **Jenkins** will open.

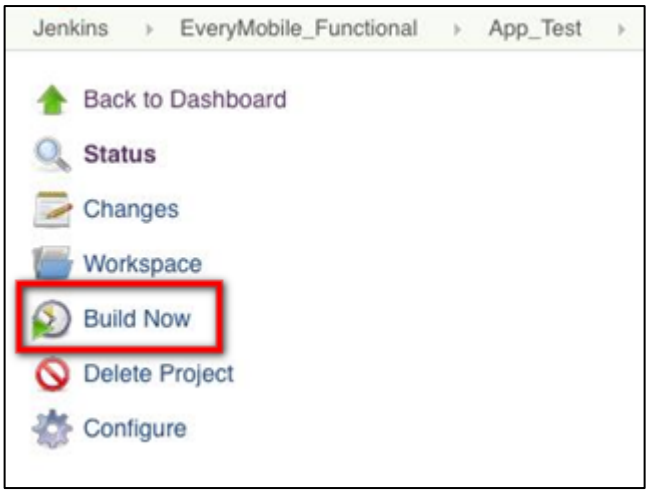


For creation of new View, refer the Reference Section below.

4. As the Jenkins open, the user will be able to see the test case that was created in the previous steps as a new job in the Jenkins.
5. The user has to perform certain steps to execute the test case before the job is being build. Click here to see the steps.

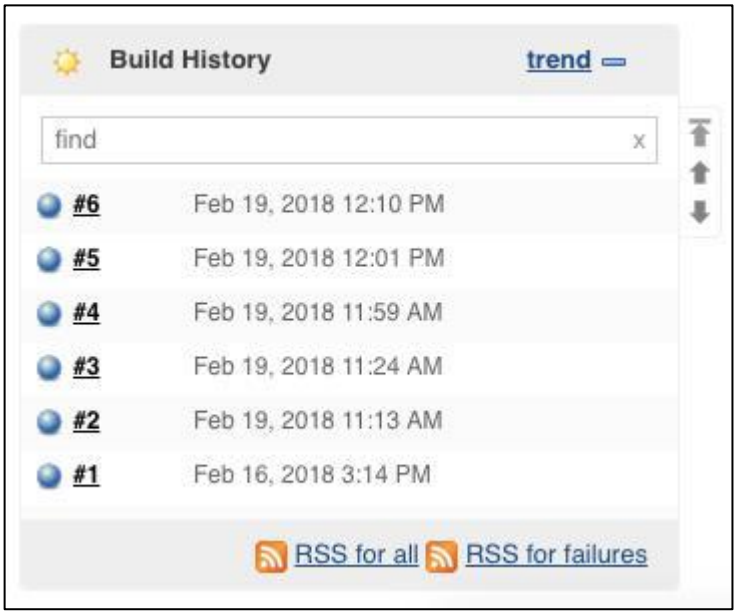
6. Click on the job that has to be executed.

7. On the left side, select Build Now.



8. As the user clicks **Build Now**, the simulator will open if the user has written code for simulator or it will execute in the phone is the code is written for the real device.
9. On the left side, under Build History, the progress of the job is shown.

10. For the first time, while executing the test case, if the WebDriverAgent is not installed already then, the WebDriverAgent will get installed automatically.
11. If the job is successfully built, it will show the built time, number of times of build and status of the built.



12. All the jobs of the Jenkins will be saved in:

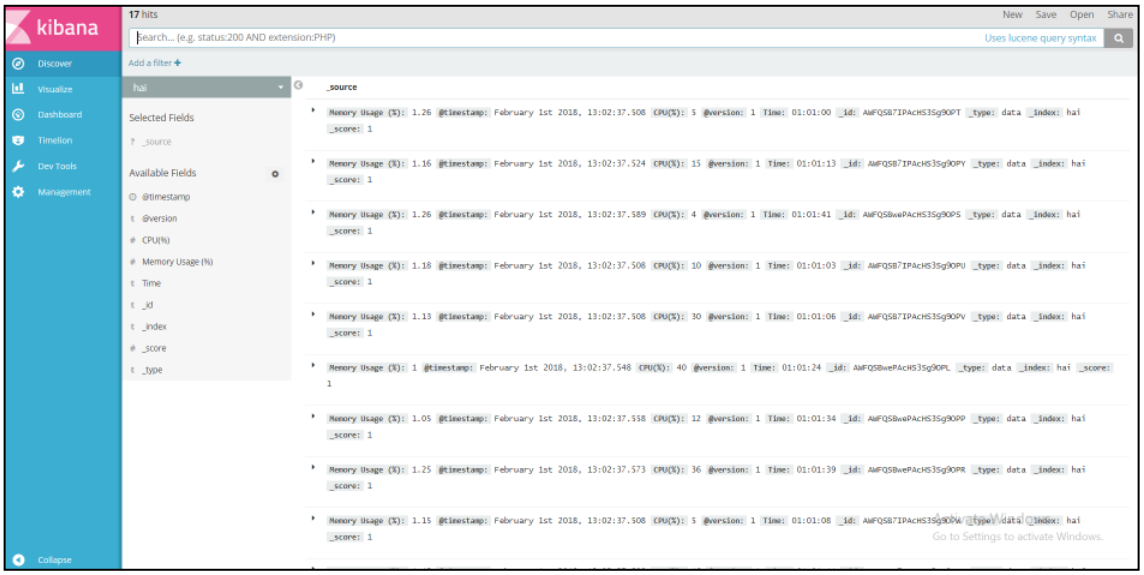
> TON > MobileTesting > TestOps > Demo_TON > <TestCase_Name_Folder> > <Job_Name_Folder>

5.4 ANALYZE RESULTS

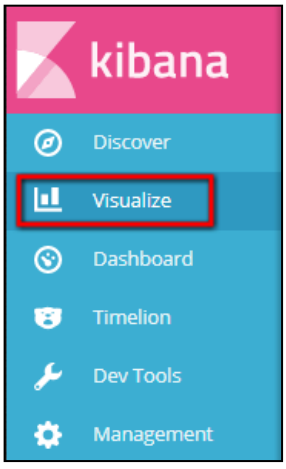
The **Analyze Results** button is used to analyze the final result of the test case that had been recorded and executed in the previous steps. The final result will be displayed in the form of graphs

1. Click on the **Analyze Results**.

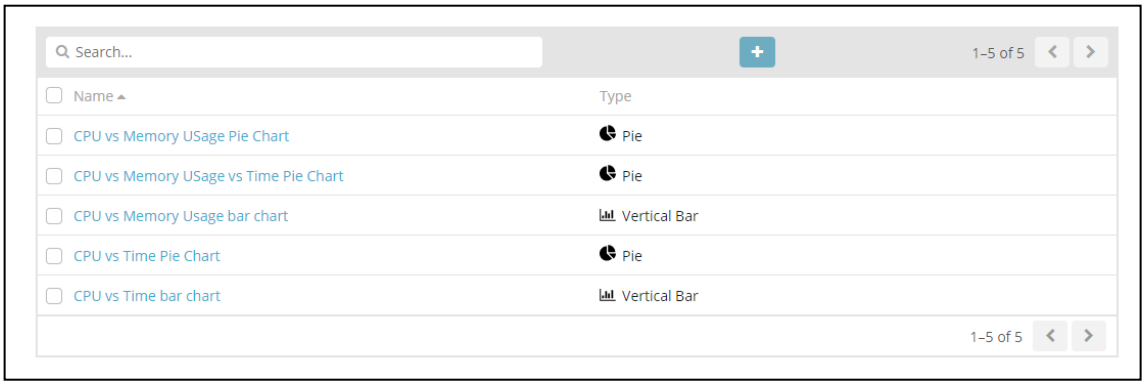
2. On clicking on the **Analyze Results** button, Kibana opens up in a new window.



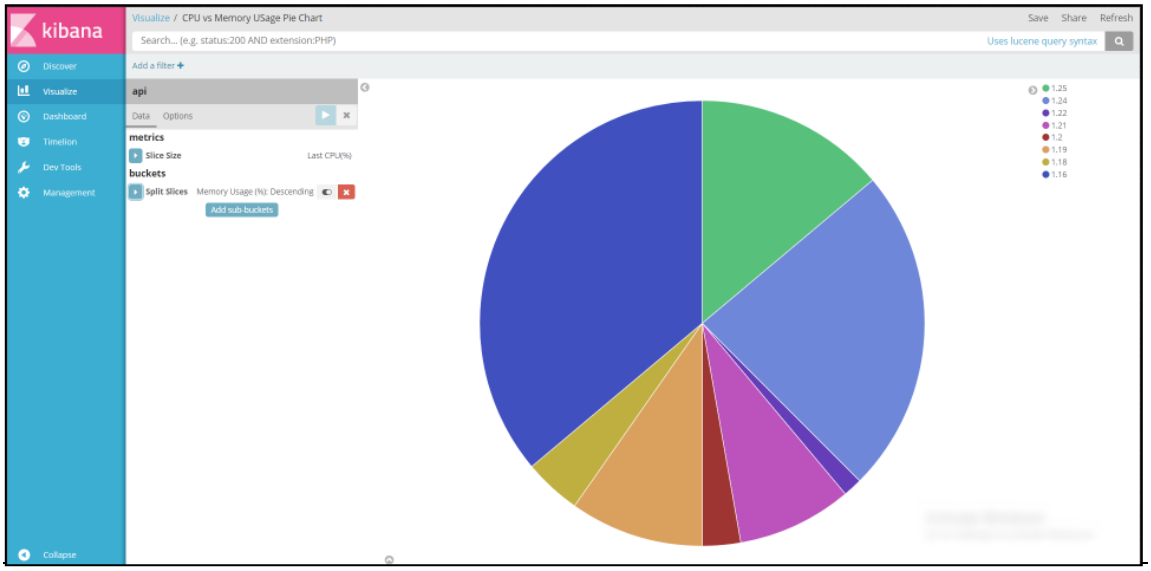
3. On the left panel, click Visualize.



4. The user will find charts of different forms with different combinations like CPU vs Time, CPU vs Memory Usage vs Time etc.



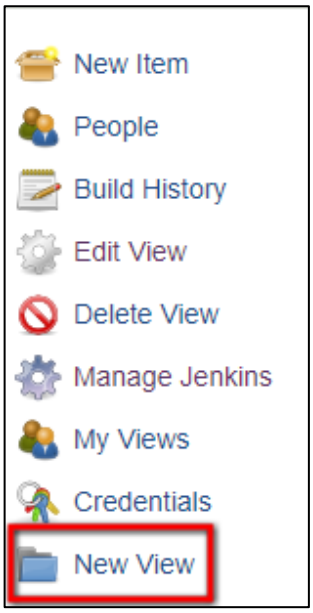
5. On click of the chart name, the user can see the graph based on the type of the chart. (The example shows for Pie Chart)



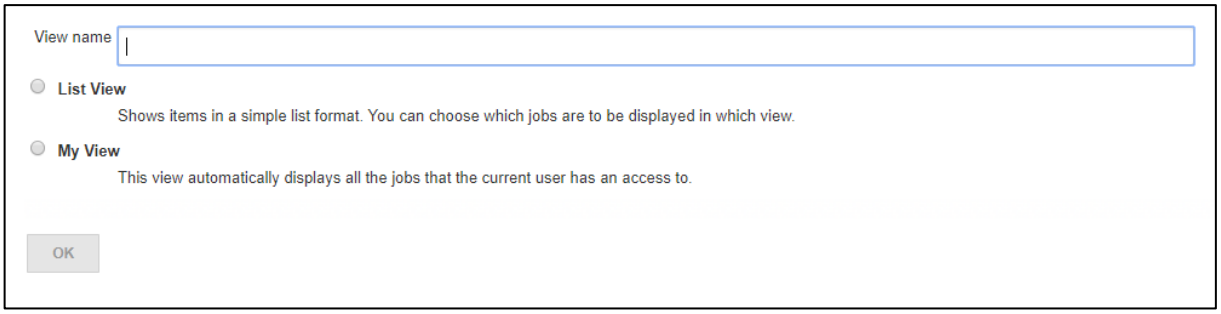
6. REFERENCES

6.1 CREATION OF NEW VIEW

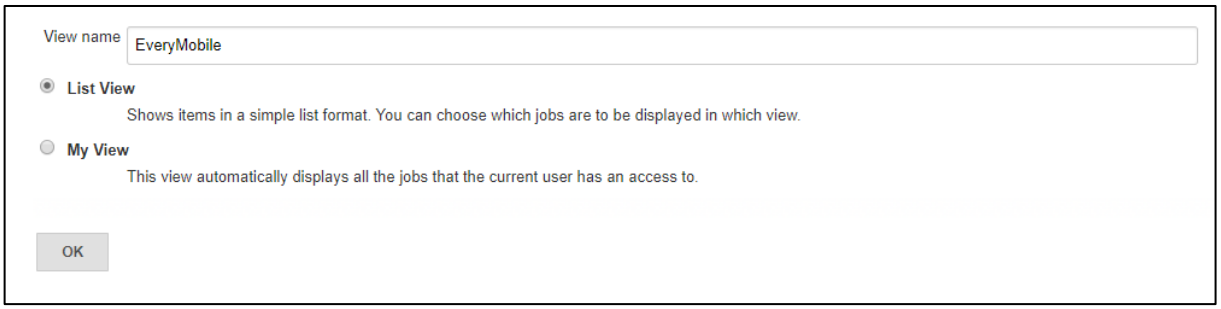
- To create a new view, on the left panel, click **New View**.



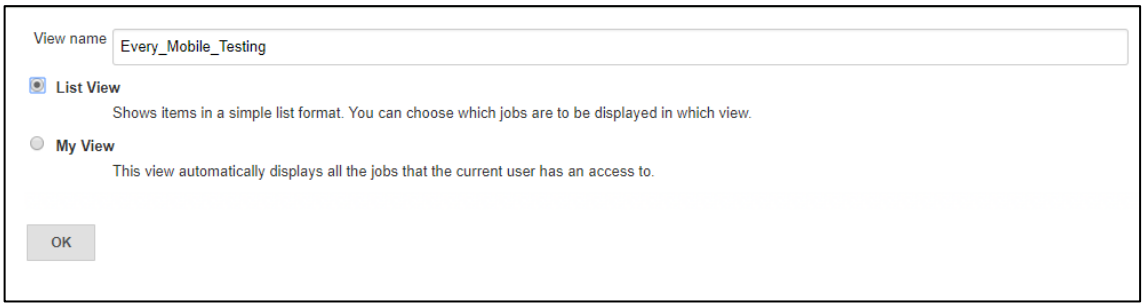
- On clicking, give the name of the tab as **Every_Mobile_Testing** and select the type of view.



- On entering the name and selecting the view type, the OK button becomes active. Click Ok.



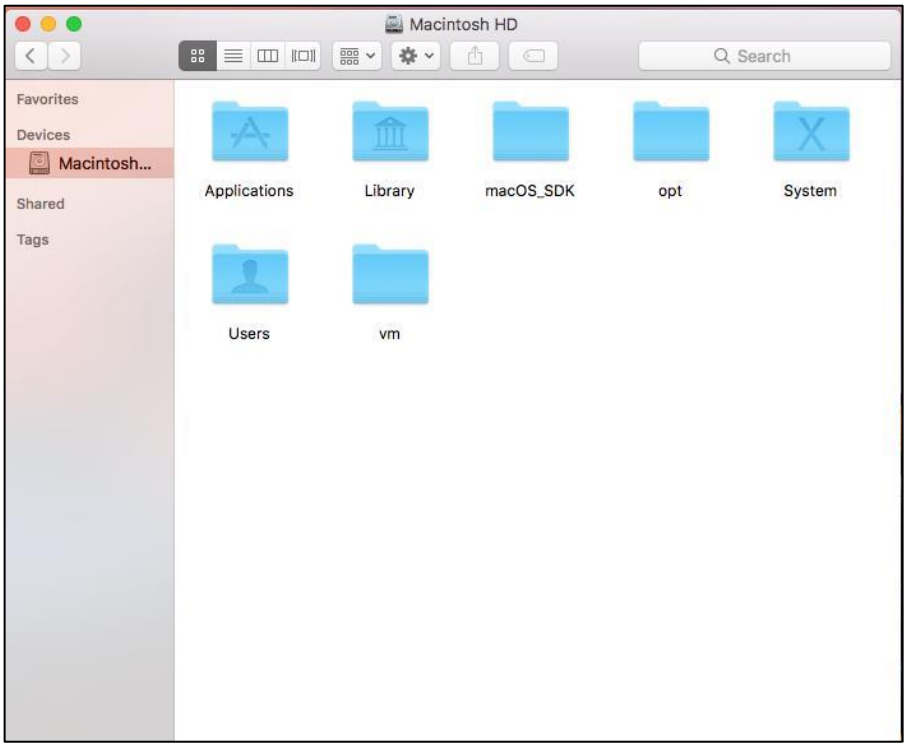
- In the next page, select the jobs that are to be added to this tab.



- Once done, click Apply and Ok to save and Close the creation.

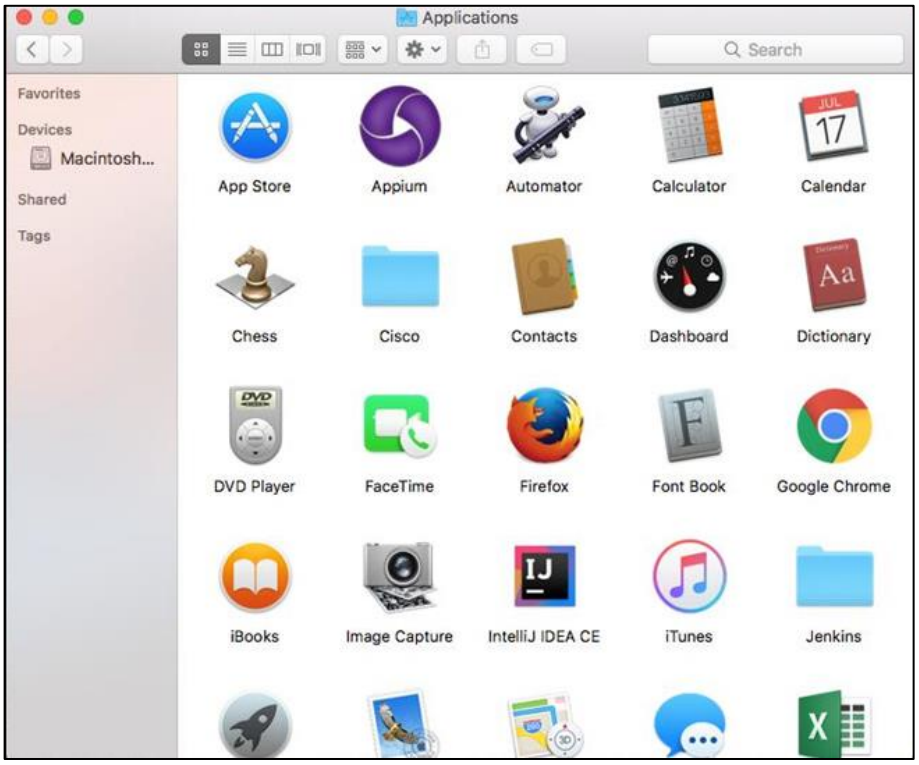
6.2 STARTING APPIUM

- Open Finder and go to the root home folder.



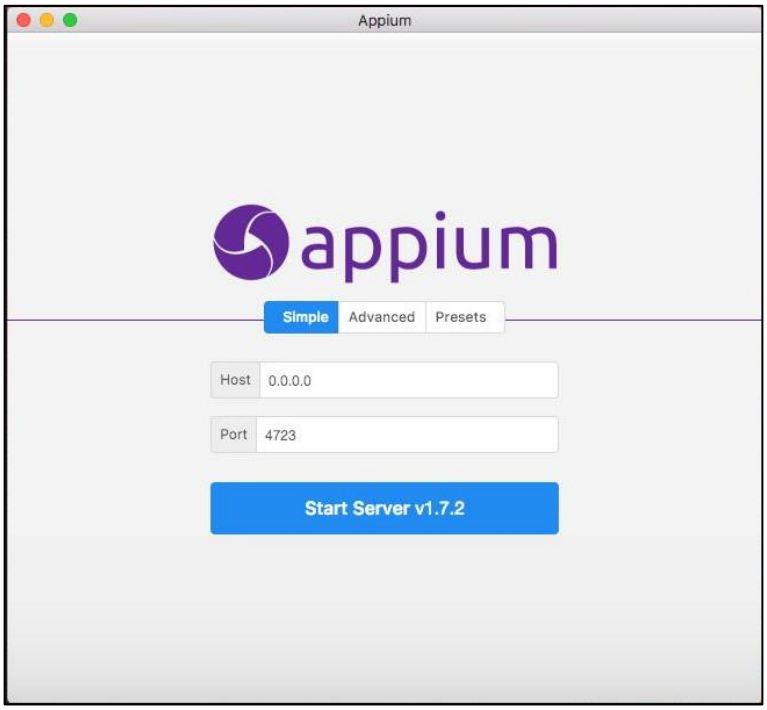
- Click on the Applications Folder.

- In the Applications folder, surf for **Appium**.

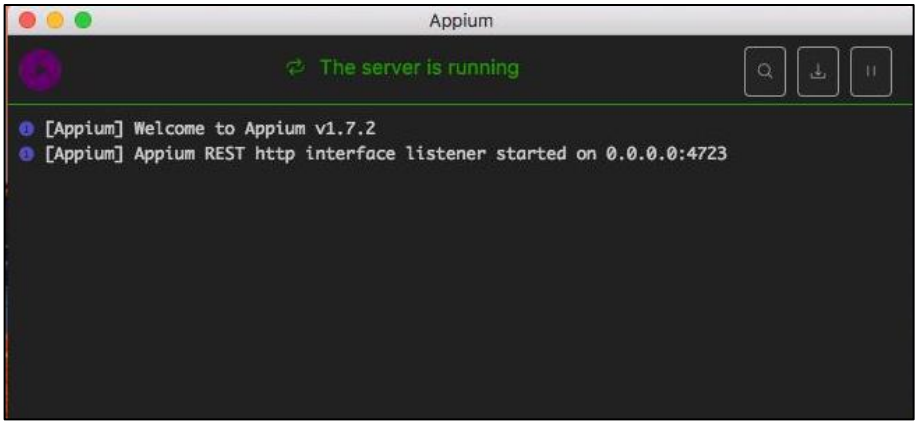


- Double click on the Appium.

- It will open the Appium GUI.



- To start the server, click on the **Start Server**.
- It will start the Appium server.

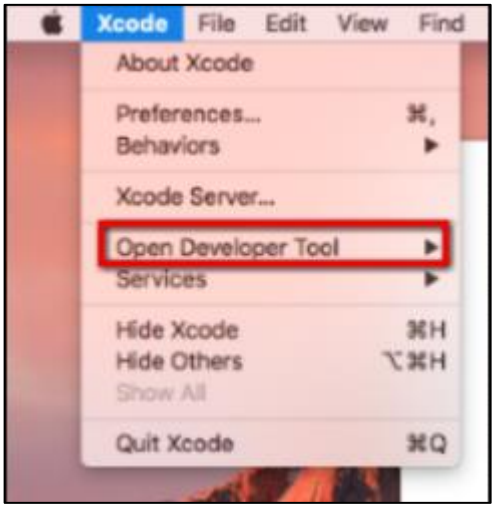


6.3 REAL DEVICE AND SIMULATOR

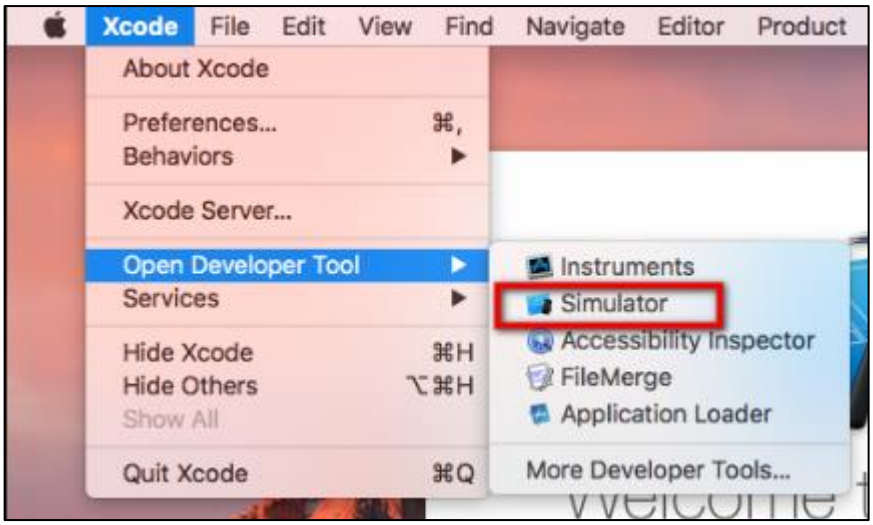
The test cases can be developed using the real users iPhone and the simulator.

6.3.1 CONNECTING SIMULATOR

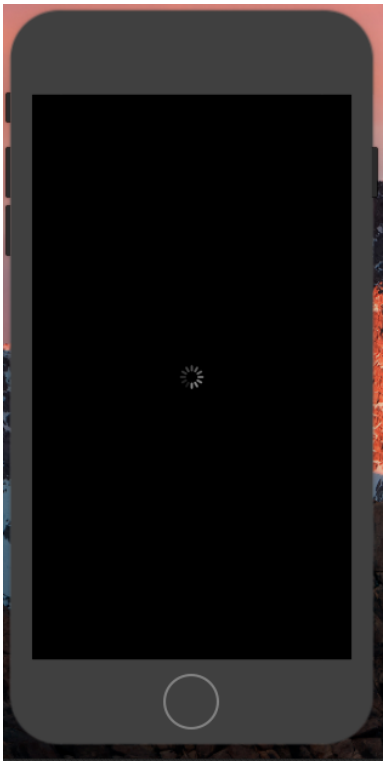
1. Open Xcode.
2. From the top, click on Xcode.
3. Choose Open Developer Tool.



4. From the right-side menu, choose Simulator.



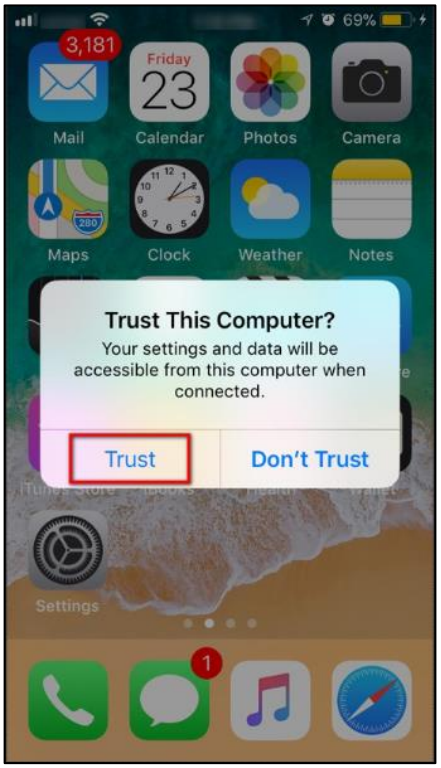
5. It will open the Simulator.



6. The user can choose the type of iPhone and its version (for the first time). From second time, it will open the default simulator.
7. It will take some time to start the simulator.

6.3.2 CONNECTING REAL DEVICE

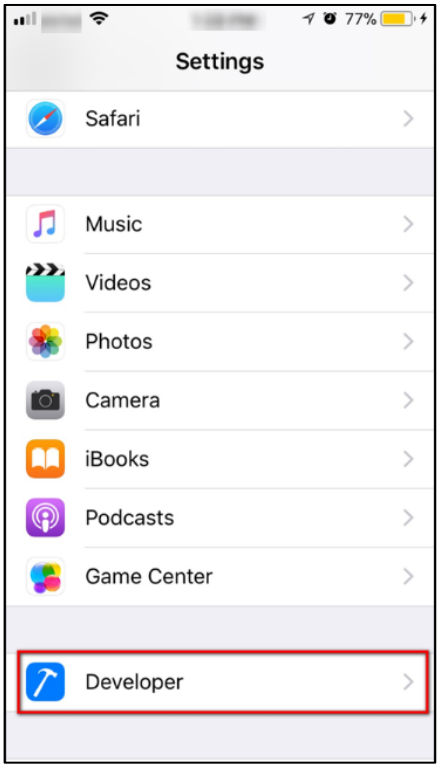
1. Firstly, open the Xcode in the Mac system.
2. Now, the user should connect the phone through the USB cable.
3. As the phone gets connected to the system, it will ask whether to **Trust** this computer or not.
4. Click on the **Trust.**



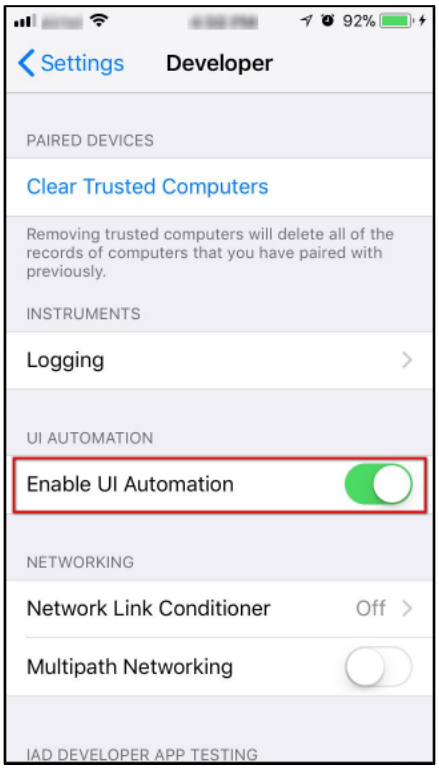
5. It will ask for the password to unlock the phone.

6. Open the settings in the phone.

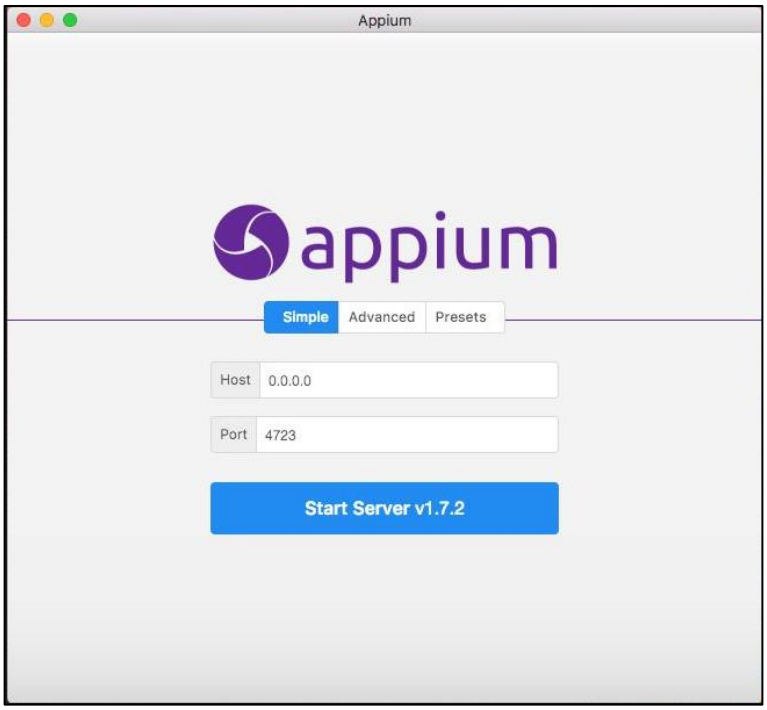
7. Scroll down and the user will find the **Developer** settings.
8. Click on the **Developer** settings.



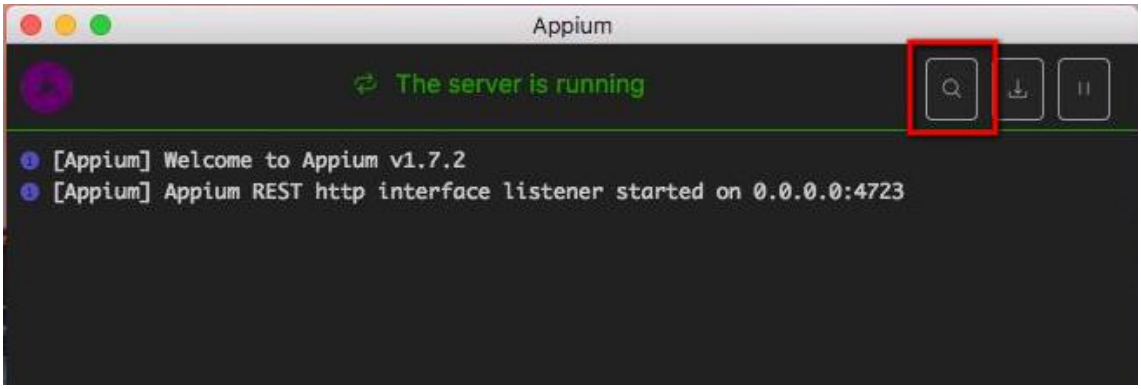
9. Enable the **Enable UI Automation** option.



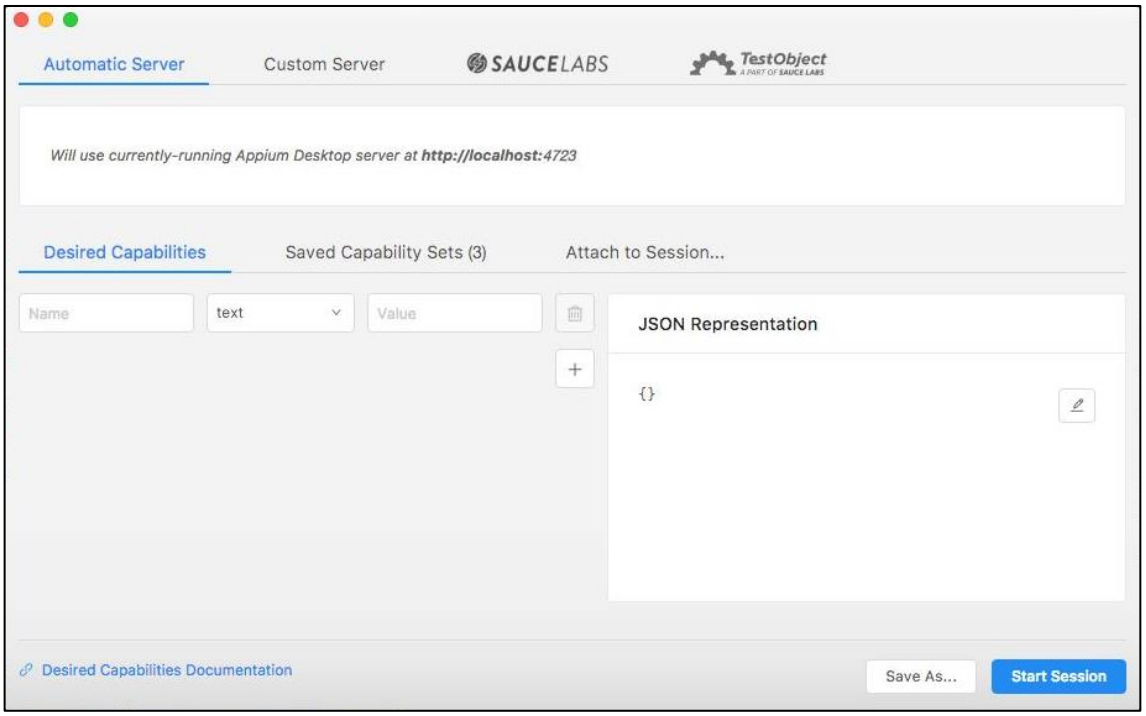
10. Start the Appium server.



11. And click the **Start Server**.
12. On the top right, click the **Inspector Session.**

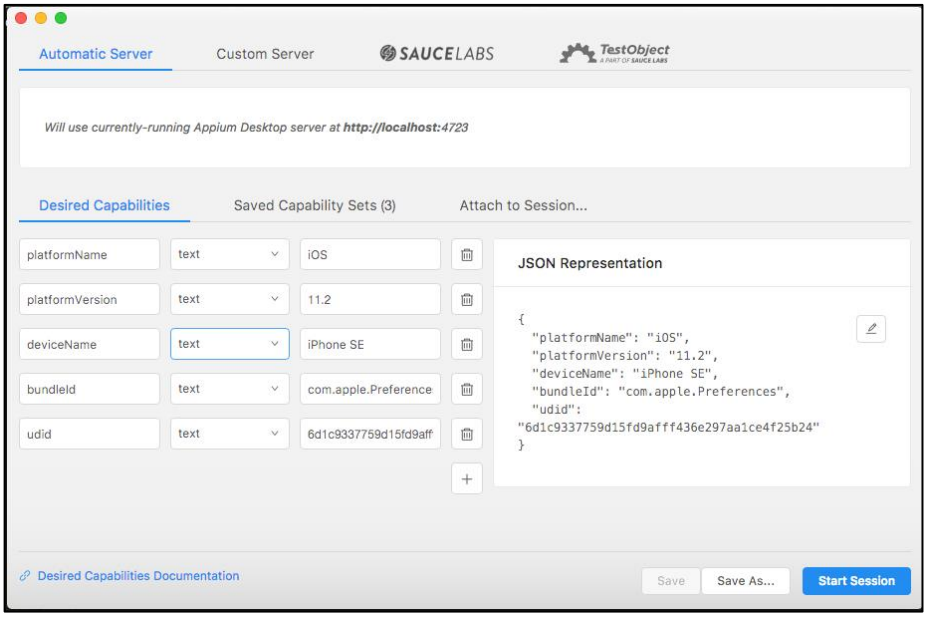


13. It will open a new window.

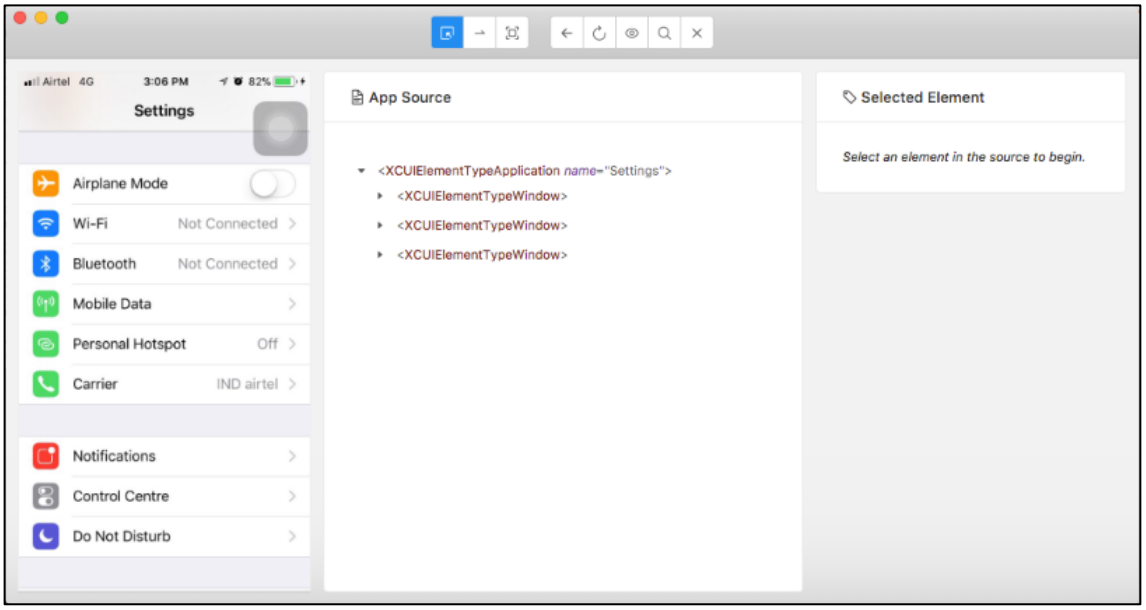


14. Under Desired Capabilities, the user has to enter the following:

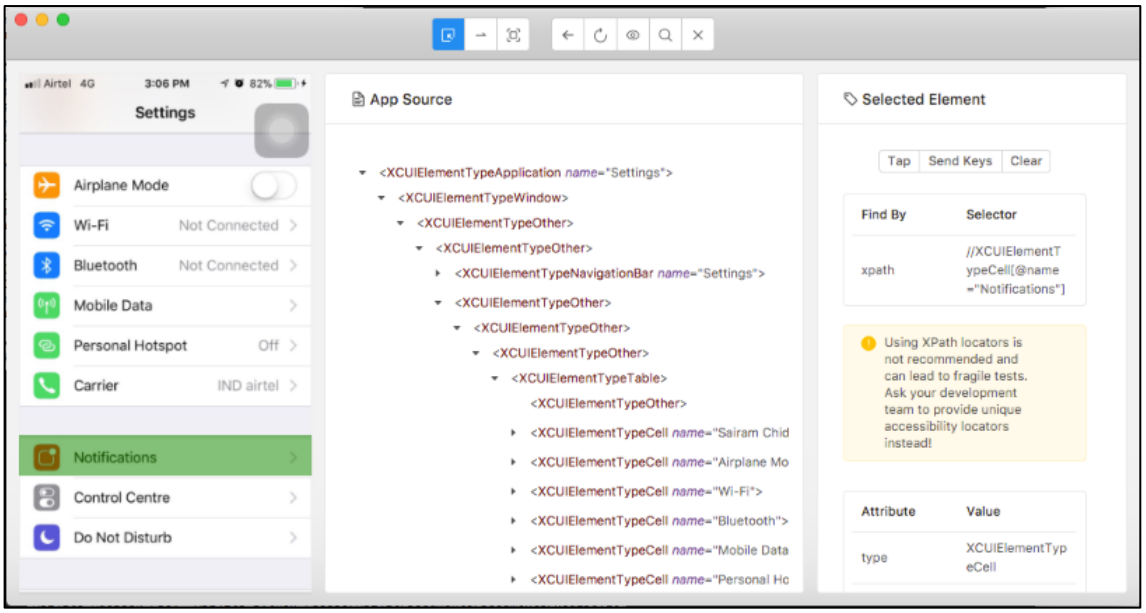
- Device name – must be iPhone that is being used for the testing.
- Platform version – current version in the iPhone is running.
- Platform name – the platform name will remain as **iOS** always.
- Bundle Id – the app that has to be opened. In order to find the bundle id of an application, click here.
- Udid – Unique Device Identifier (applicable for real devices).



15. After entering the desired capabilities, click on the **Save as** button.
16. Click on the **Start Session** button.
17. A new window will open with the app in the iPhone whose bundle id is given.



18. Click on any button on the left and on the right, it will show the details of the selected button.
19. On the right-side panel, the user will be able to see the Xpath.

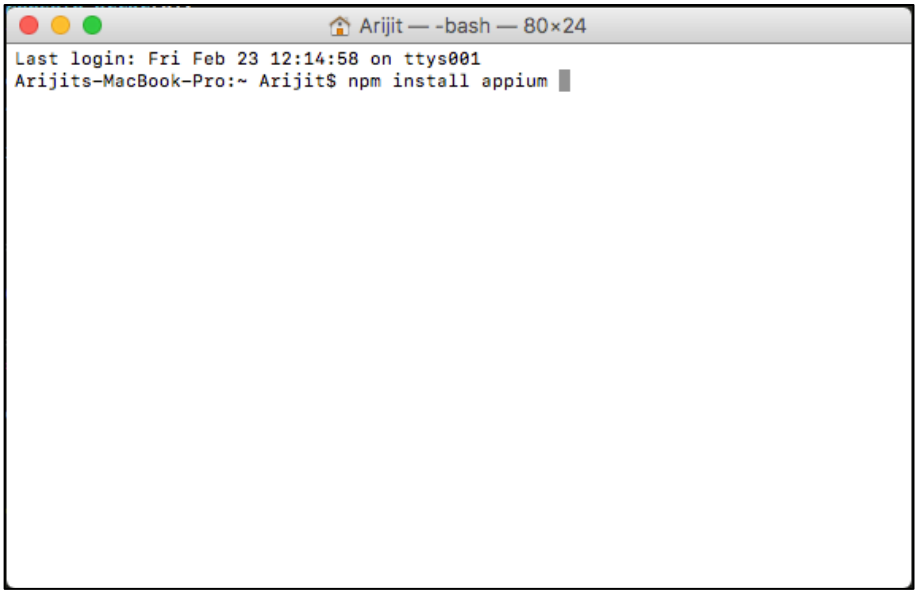


20. The Xpath is used for writing the code.
21. After getting the Xpath, the user must not forget to stop the Appium server.

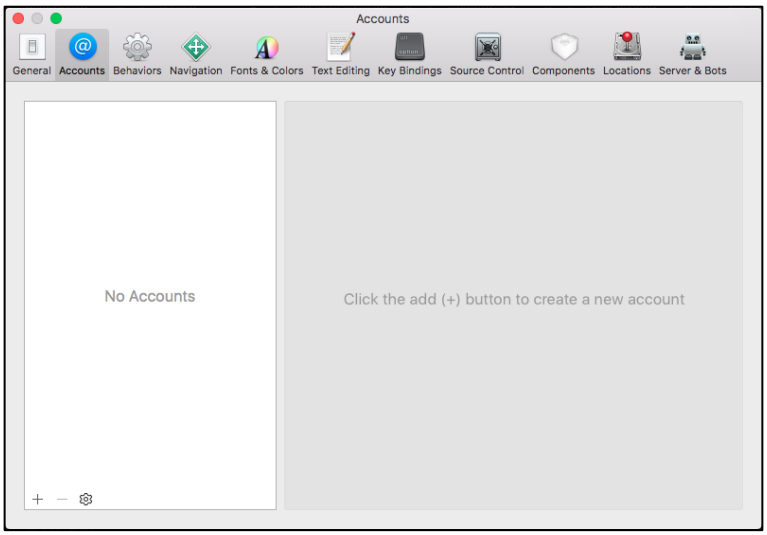
22. In the code, the user must have the following:
    - Device name – must be name of the iPhone that is being used for the testing.
    - Platform version – current version in the iPhone is running.
    - Platform name – the platform name will always remain as **iOS** always.
    - Bundle Id – the app that has to be opened. In order to find the bundle id of an application, click here.
    - Udid – Unique Device Identifier (applicable for real devices).
    - xcodeOrgId – paid team id that the user will get from Apple organization.
    - xcodeSigningId – must be given as "**iPhone developer**".
    - AgentPath – the location where the WebDriverAgent.xcodeproj is stored.
    - BootstrapPath – the location where the WebDriverAgent is stored

6.4 EXECUTION

1. During execution, the same real device that was used in the writing of code must be connected.
2. For the first time for every new iPhone the user connects, the user has to perform the following steps.
3. The user has to install Appium using the command prompt. The command to install is `npm install -g appium`.

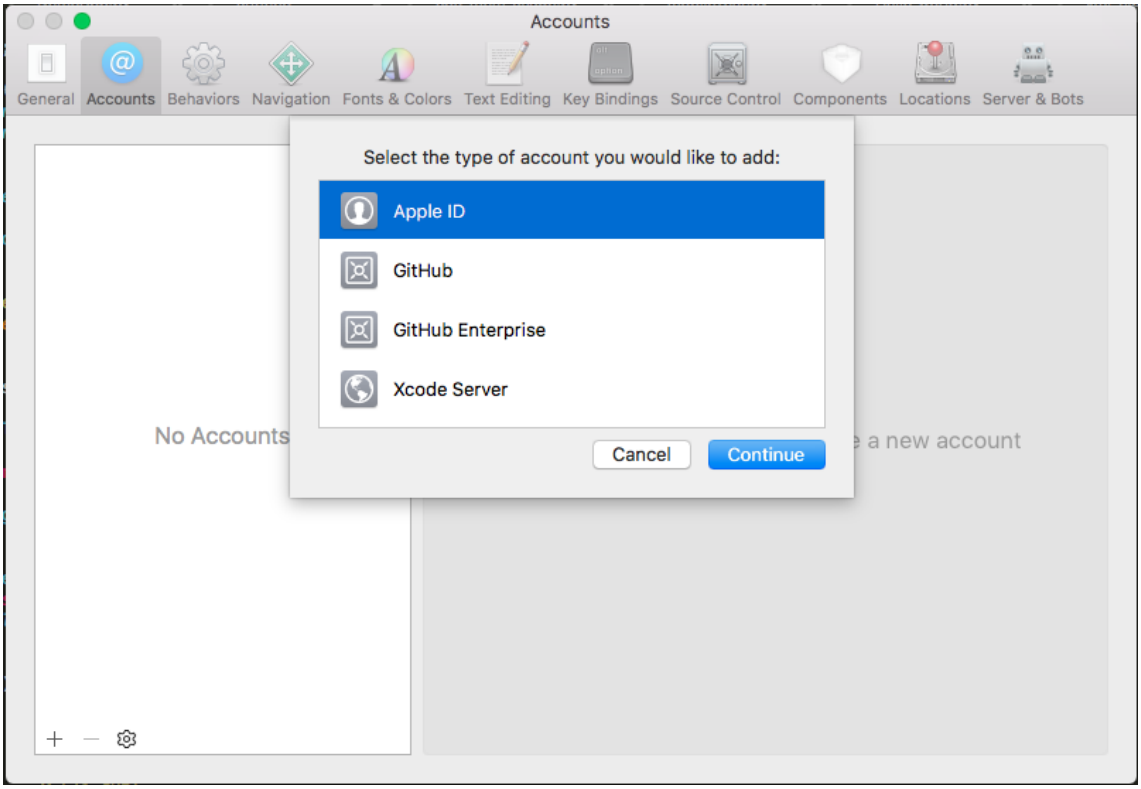


4. To login into the XCode, open Xcode.
5. Click on xcode on the top -> Preferences.
6. It will open a window.

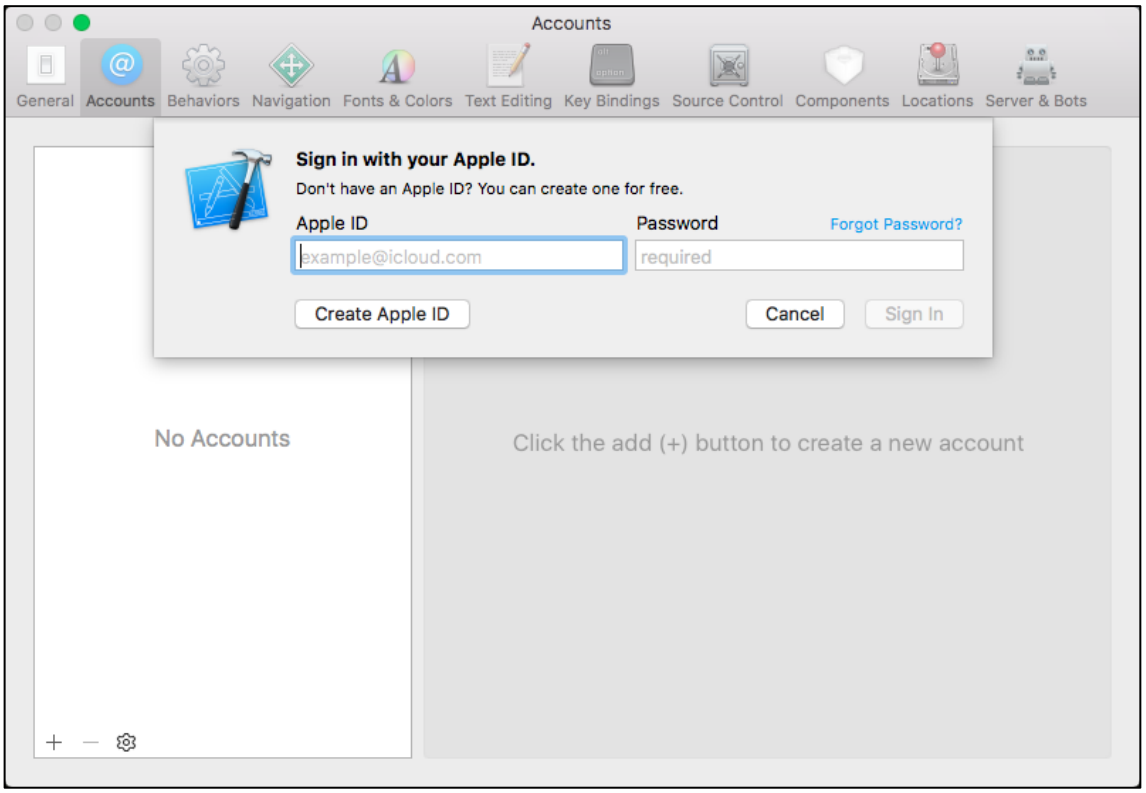


7. From there, select Accounts.
8. Click on the plus icon at the left bottom of the window.

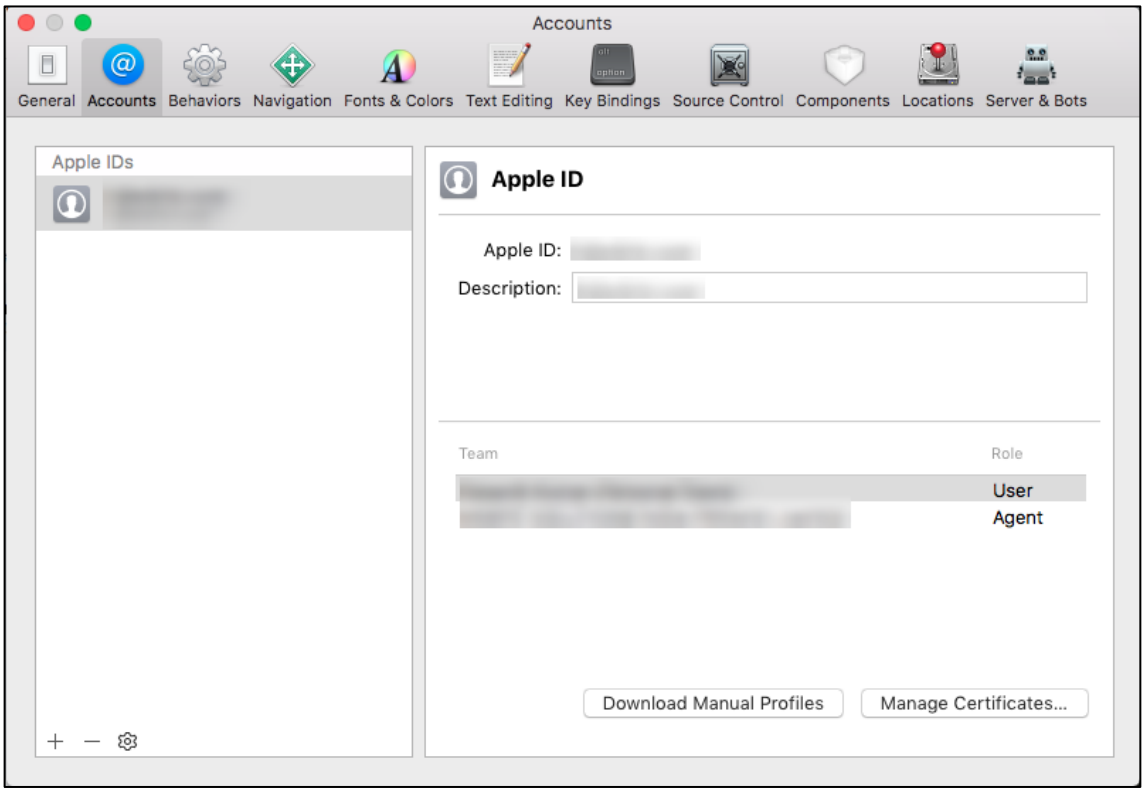
9. Choose apple id and click continue.



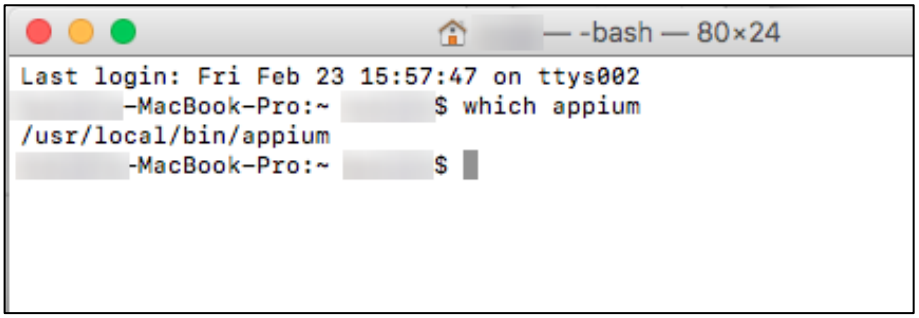
10. Enter the apple id and its password and click sign in.



11. It will show the logged in apple id.



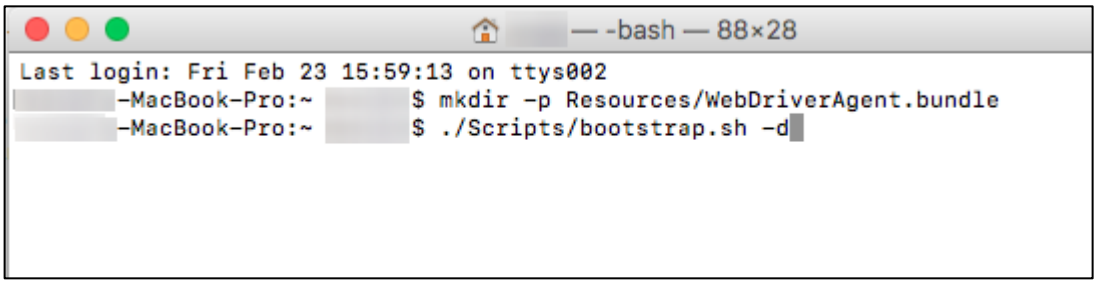
12. To find out where the appium is stored, give the command `which appium` and press enter.



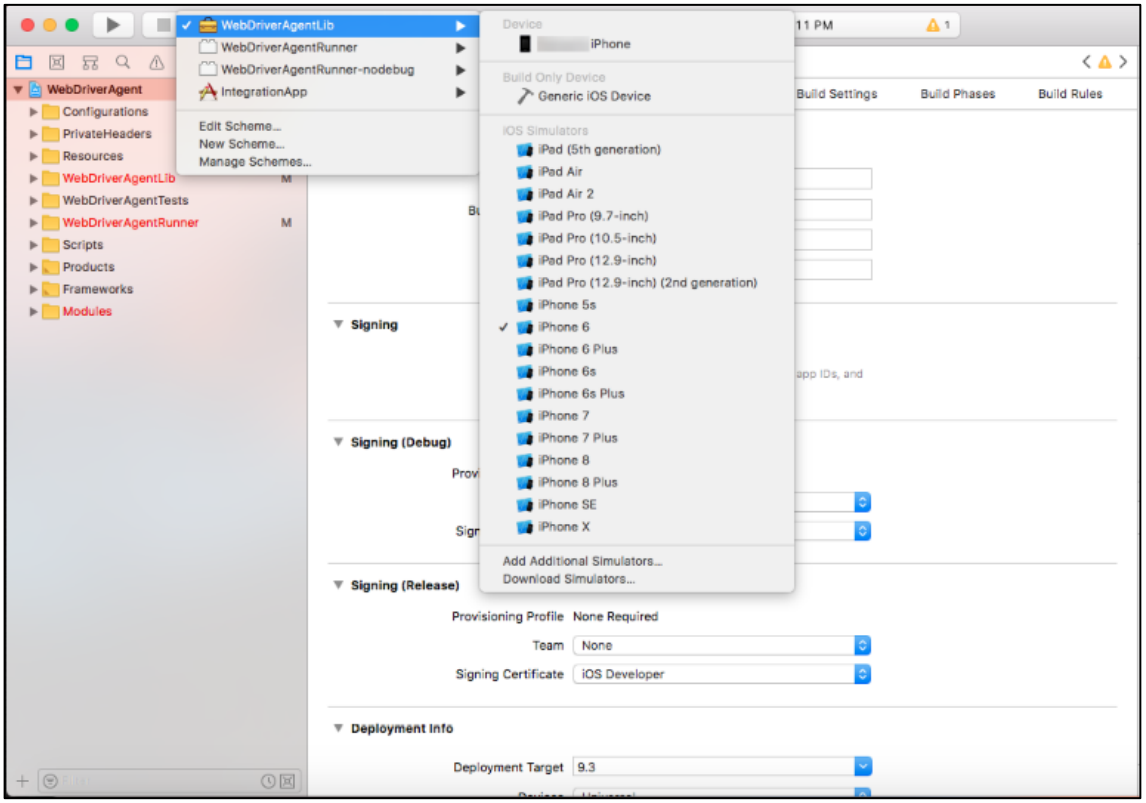
13. It will show the folder path where the appium is stored. For example, the appium is stored in `/usr/local/bin/appium`.

14. The WebAgentDriver will be found in:
`/usr/local/lib/node_modules/appium/node_modules/appium-xcuitest-driver/WebDriverAgent`

15. Now open a terminal and go to the above-mentioned folder location and type the following commands:
    - `mkdir -p Resources/WebDriverAgent.bundle`
    - `./Scripts/bootstrap.sh -d`

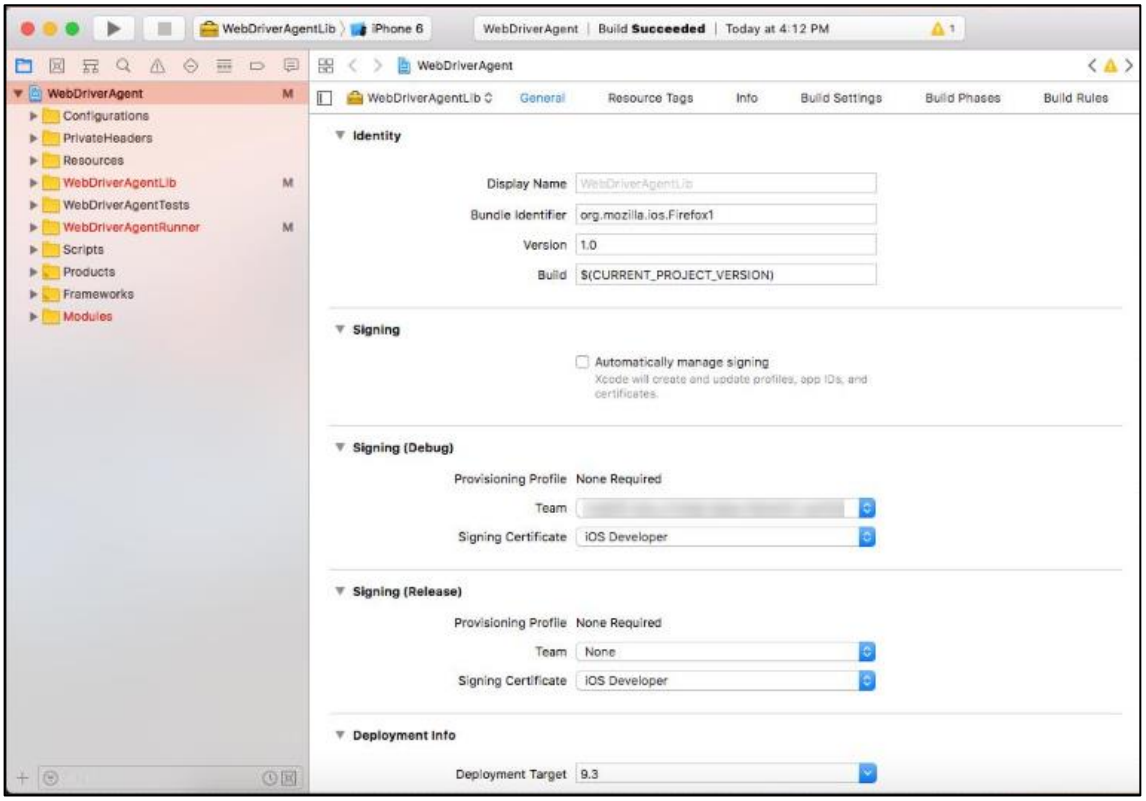


16. Now open the Xcode.
17. Click on **Open another project**.
18. Surf for **WebDriverAgent.xcodeproj** in **All files.**
19. Click and open the **WebDriverAgent.xcodeproj.**
20. It will open WebDriverAgent.
21. Select WebDriverAgentLib.



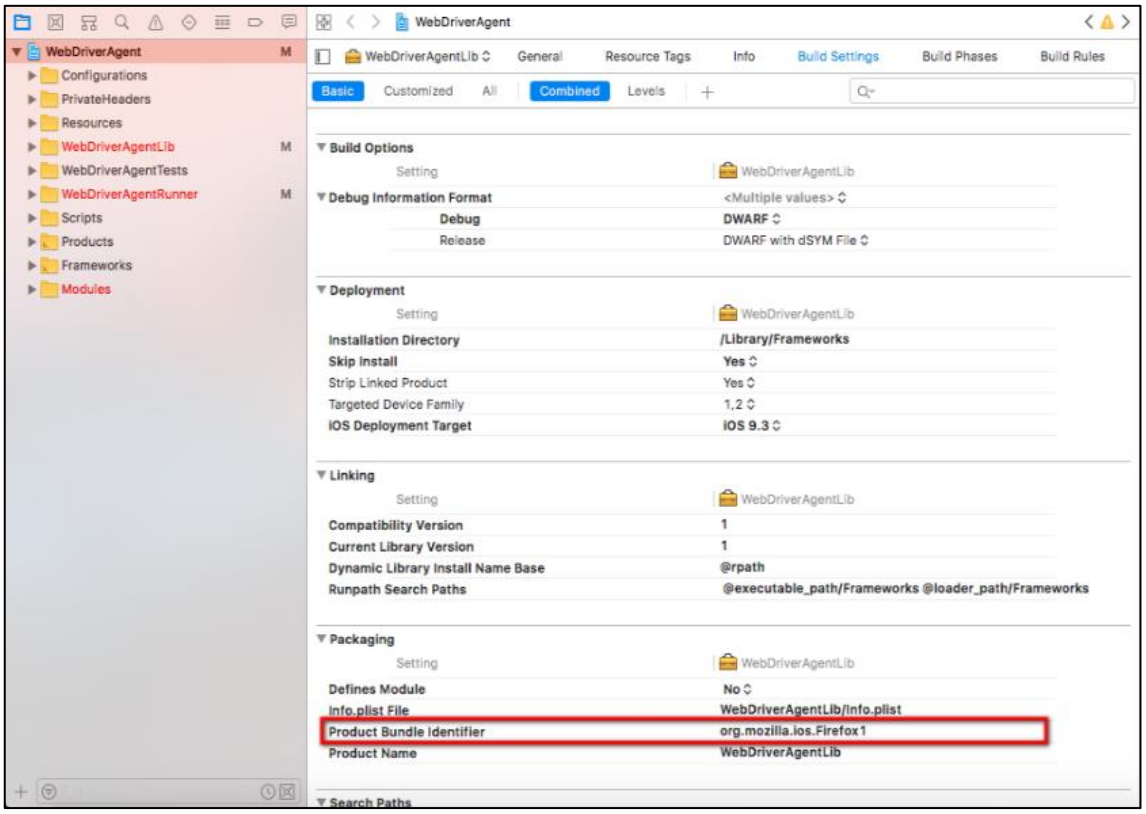
22. Choose the iPhone that is connected to the system.
23. In the **General** tab, under Signing, do the following:

- Select the Team as **Company Name** and NOT the **Personal Team**.
- For signing Certificate, select **iOS Developer** and NOT the **iOS Distribution**.

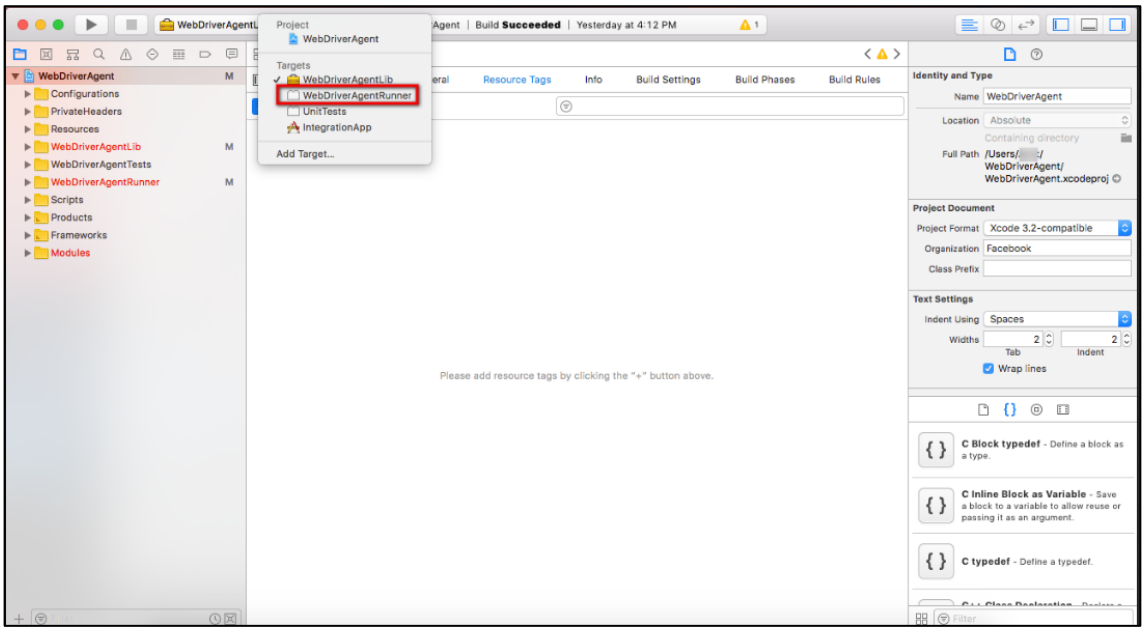


24. Go to **Build Settings**.
25. Look for **Product Bundle Identifier**.

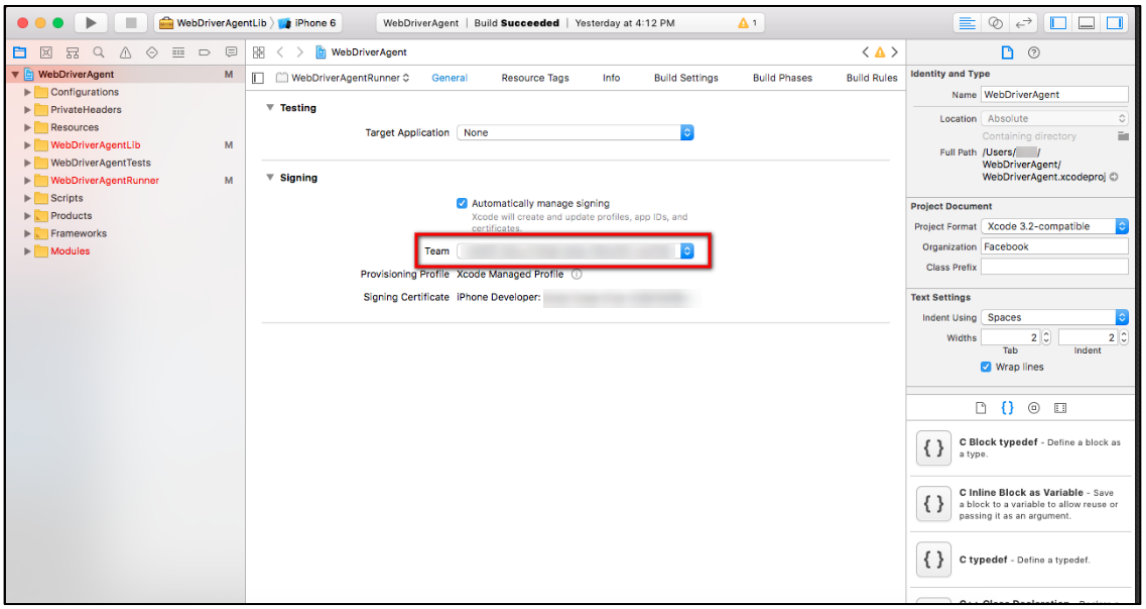
26. Set the Product Bundle Identifier to something that Xcode will accept.



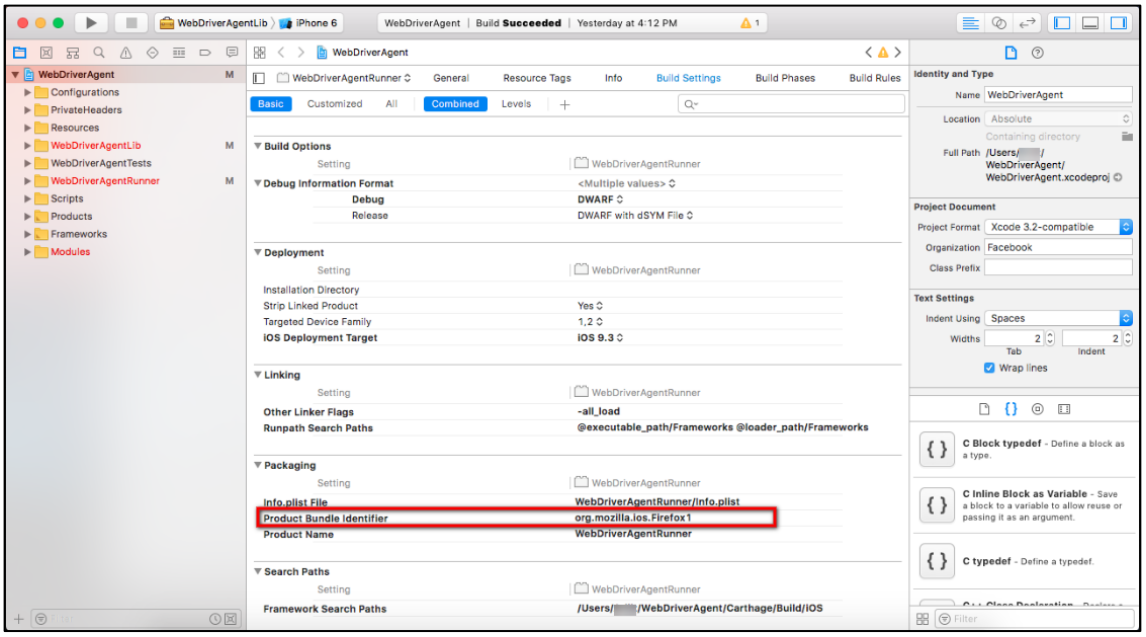
27. Now change the **WebDriverAgentLib** to **WebDriverAgentRunner**.



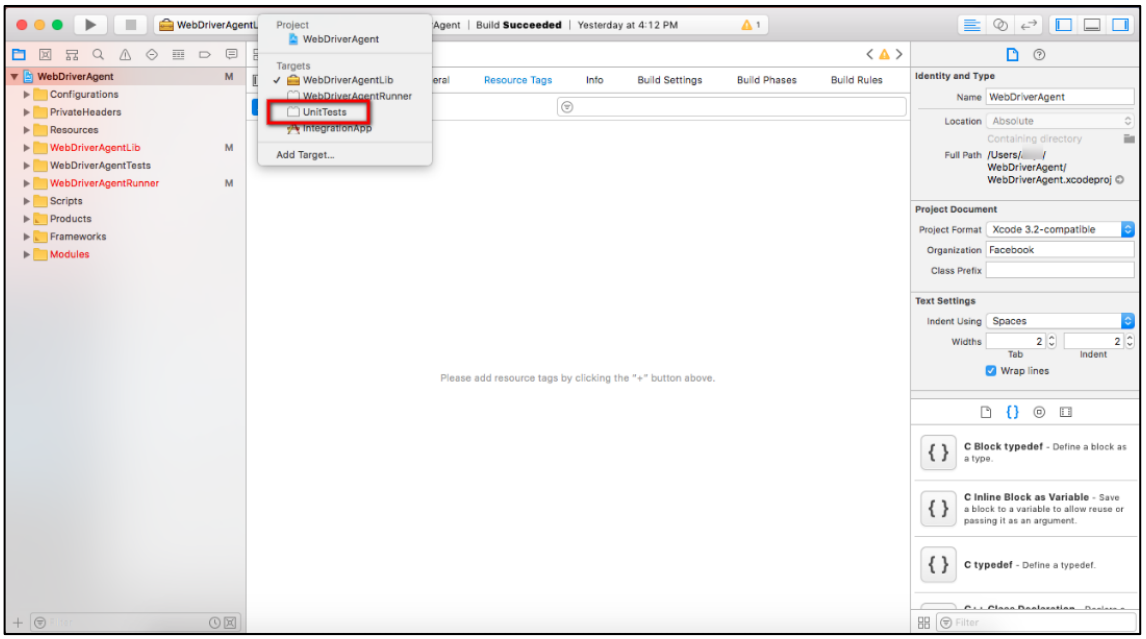
28. Under **General** tab, set the Team to the **Company Name**.



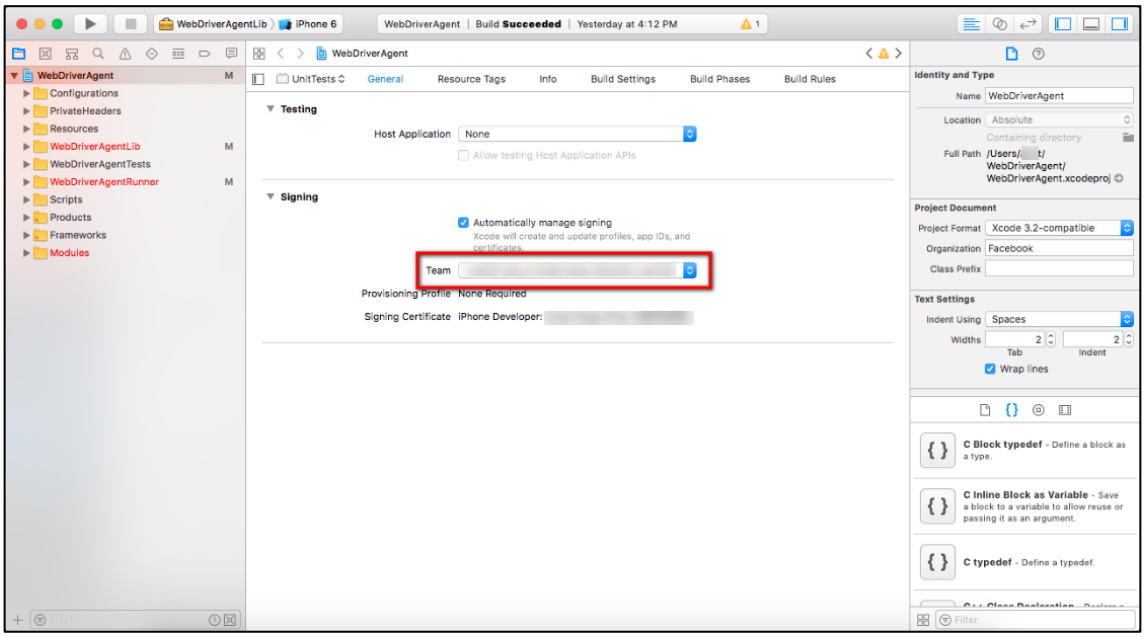
29. Now go to **Build Settings**
30. Look for **Product Bundle Identifier**.
31. Set the Product Bundle Identifier to the same **Product Bundle Identifier** that was given in the **WebDriverAgentLib**.



32. Now change the WebDriverAgentRunner to **UnitTests**.

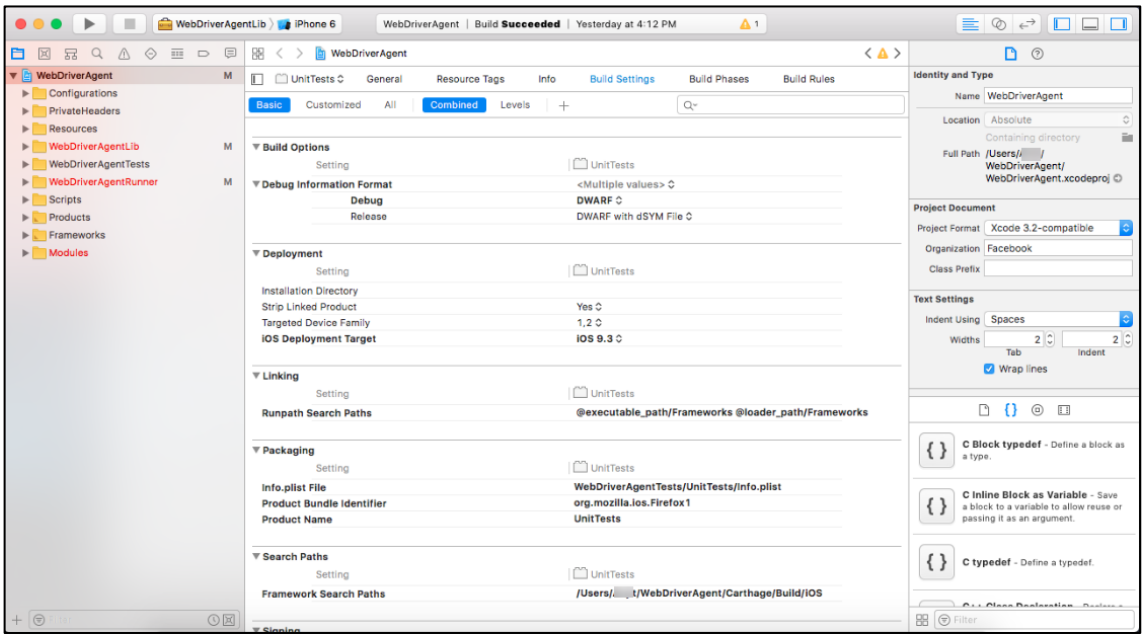


33. Under General tab, set the Team to the **Company Name**.

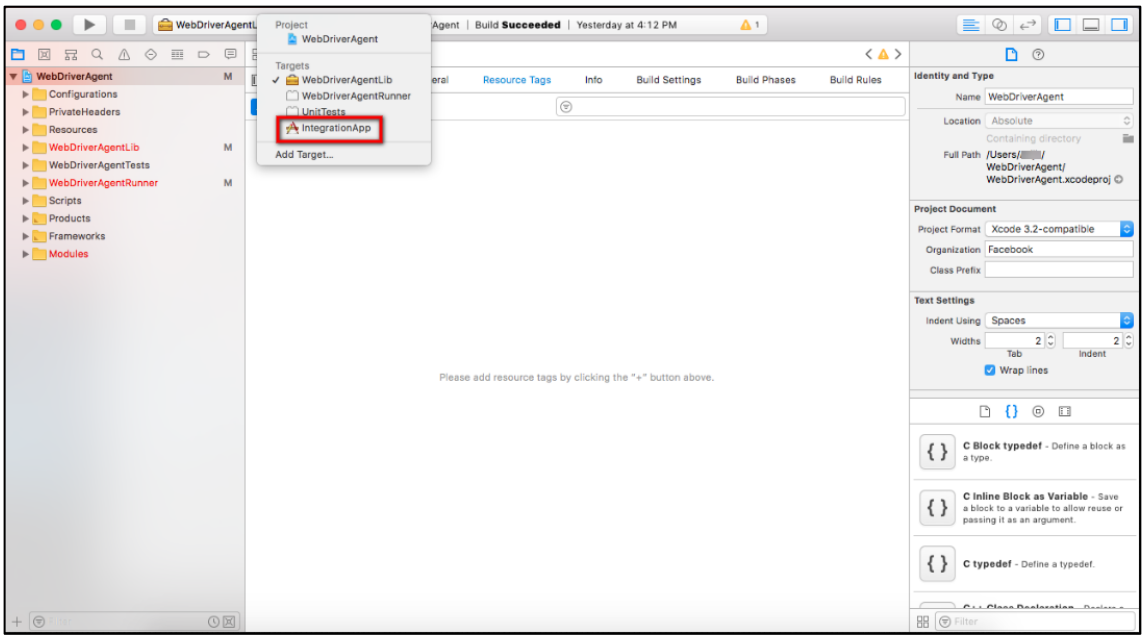


34. Now go to **Build Settings**
35. Look for **Product Bundle Identifier**.

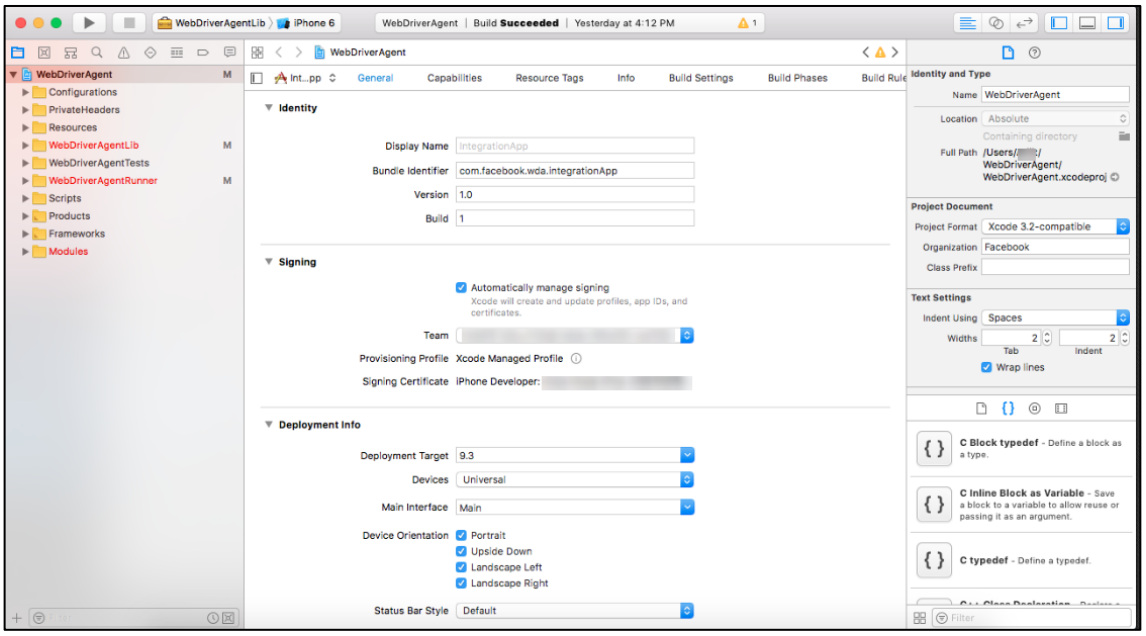
36. Set the Product Bundle Identifier to the same **Product Bundle Identifier** that was given in the **WebDriverAgentLib**.



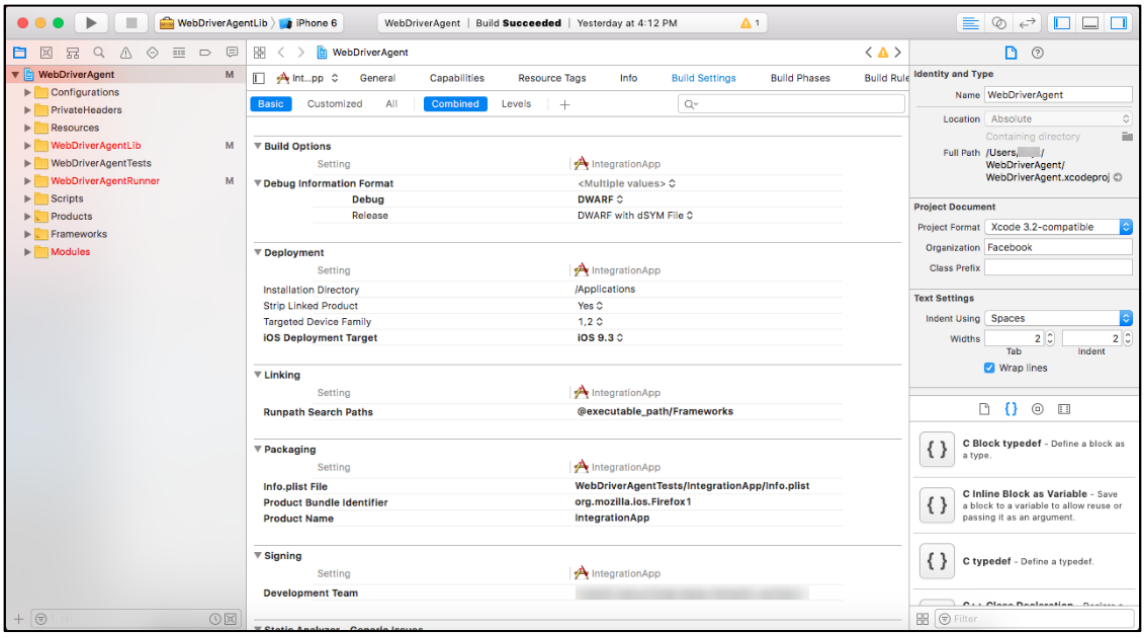
37. Now change the **Integration App** from UnitTests.



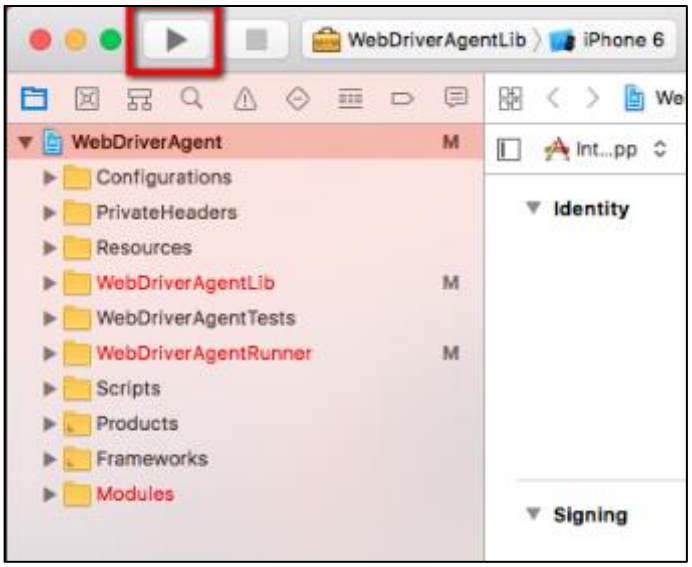
38. Under General tab, set the Team to the **Company Name**.



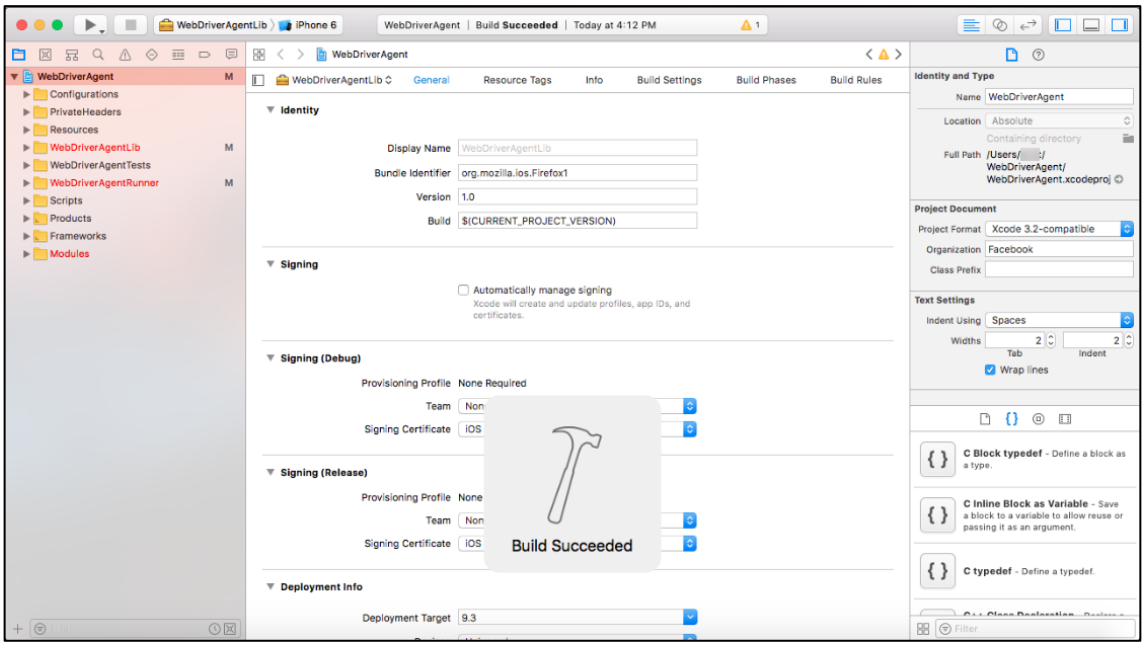
39. Now go to **Build Settings**
40. Look for **Product Bundle Identifier**.
41. Set the Product Bundle Identifier to the same **Product Bundle Identifier** that was given in the **WebDriverAgentLib**.



42. Now click on the **Build** button.



43. It will show that **Build Succeeded** if the building is done successfully.



44. For the first time, while executing the test case, if the WebDriverAgent is not installed already then, the WebDriverAgent will get installed automatically.