



# Let's make an interactive session







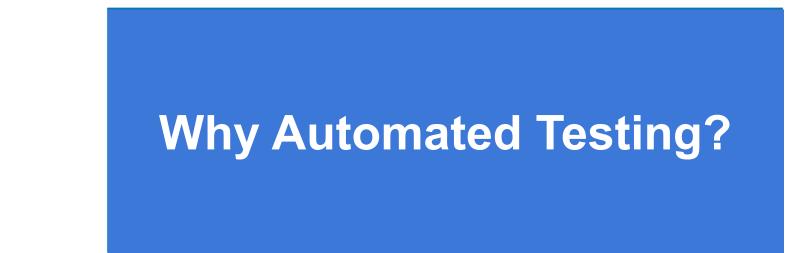


- A testing framework or more specifically a testing automation framework is an execution environment to perform automated tests.
- It is defined as the set of assumptions, concepts, and practices that constitute a work platform or support for automated testing.
- Defining the format in which to express expectations.
- Creating a mechanism to hook into or drive the application under test.
- Executing the tests and reporting the results.



- Faster turnaround
- Repeat execution
- Ensured Quality
- Team's Quality
- Unit Test Automation
- Pre-Automation Testing
- System and Integration Testing
- Sprint Complete













Thanks guys It is my pleasure.





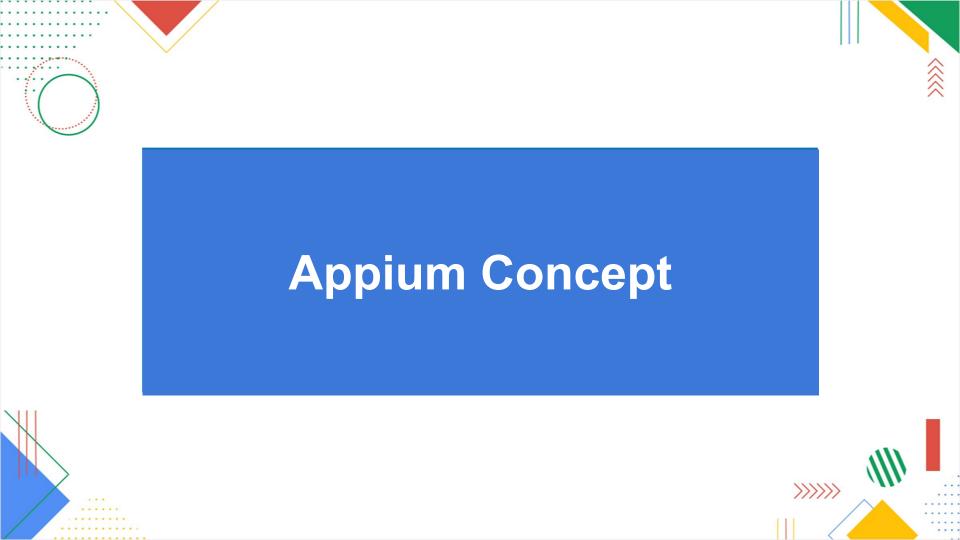


- Appium is an open-source tool for automating native, mobile web, and hybrid applications on iOS mobile, Android mobile, and Windows desktop platforms.
- Importantly, Appium is "cross-platform": it allows you to write tests against multiple platforms (iOS, Android, Windows), using the same API. This enables code reuse between iOS, Android, and Windows testsuites.



# **Appium Philosophy**

- You shouldn't have to recompile your app or modify it in any way in order to automate it.
- You shouldn't be locked into a specific language or framework to write and run your tests.
- A mobile automation framework shouldn't reinvent the wheel when it comes to automation APIs.





### **Client/Server Architecture**

- Appium is at its heart a web server that exposes a REST API.
- It receives connections from a client, listens for commands, executes those commands on a mobile device, and responds with an HTTP response representing the result of the command execution.



### Session

- Automation is always performed in the context of a session.
- Clients initiate a session with a server in ways specific to each library, but they all end up sending a POST /session request to the server, with a JSON object called the 'desired capabilities' object.
- At this point the server will start up the automation session and respond with a session ID which is used for sending further commands.



### **Desired Capabilities**

 Desired capabilities are a set of keys and values (i.e., a map or hash) sent to the Appium server to tell the server what kind of automation session we're interested in starting up.

### For, example:

```
DesiredCapabilities dc = new DesiredCapabilities();
dc.setCapability(MobileCapabilityType.DEVICE_NAME, "emulator-5554");
dc.setCapability(MobileCapabilityType.PLATFORM_VERSION, "7.1.1");
```







# **Android Support**

Android automation is supported with below drivers:

The UiAutomator2 Driver
The (deprecated) UiAutomator Driver
The (deprecated) Selendroid Driver

- Versions 2.3 through 4.2 are supported via Appium's Selendroid Driver
- Mobile web support with android 4.4+
- Devices: Android emulators and real Android devices

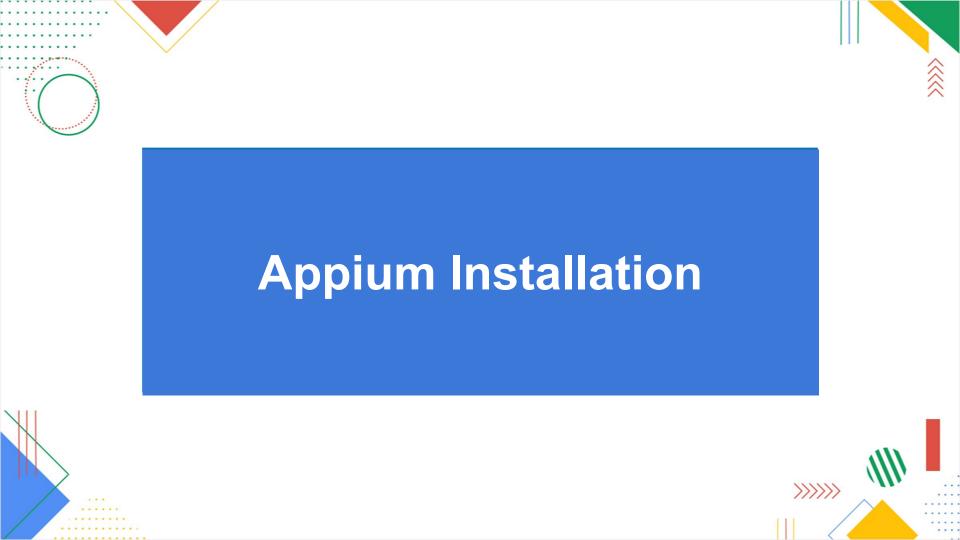


# **iOS Support**

iOS automation is supported with two drivers:

The XCUITest Driver
The (deprecated) UIAutomation Driver

- Versions: 9.0 and up
- Mobile web support: Yes, via automation of mobile Safari.
- Devices: Simulator and real device for iPhone, iPad and tvOS





# **Gradle Configuration**

### Add this two dependency in the app->build.gradle



### **Add Jar Files**

Appium Client Libraries https://appium.io/downloads.html

Selenium Client Libraries https://docs.seleniumhq.org/download/



# **Application Application**

Appium Desktop is an app for Mac, Windows, and Linux which gives you the power of the Appium automation server in a beautiful and flexible UL.

Download from here

Latest release

♥v1.13.0

-O- 9d2f313

#### 1.13.0

- dpgraham released this on May 6 · 40 commits to master since this release
- Update Appium to version 1.13.0 #964
- Add 'copy attributes to clipboard" feature #952
- Russian, Korean and Chinese translations https://crowdin.com/project/appium-desktop
- Add SauceLabs data centers option #966

#### ▼ Assets 11

	146 MB
Appium-linux-1.13.0.AppImage	125 MB
Appium-mac-1.13.0.dmg	140 MB
Appium-mac-1.13.0.dmg.blockmap	151 KB
Appium-windows-1.13.0.exe	117 MB
Appium-windows-1.13.0.exe.blockmap	117 KB
□ latest-linux.yml	379 Bytes
🗇 latest-mac.yml	517 Bytes
□ latest.yml     □	348 Bytes
Source code (zip)	
Source code (tar.gz)	







### **Automation Drivers**

#### RemoteWebDriver

- This driver class comes directly from the upstream Selenium project.
- This is a pretty generic driver where initializing the driver means making network requests to a Selenium hub to start a driver session.
- Since Appium operates on the client-server model, Appium uses this to initialize a driver session.
- However, directly using the RemoteWebDriver is not recommended since there are other drivers available that offer additional features or convenience functions.

#### **AppiumDriver**

 This driver class inherits from the RemoteWebDriver class, but it adds in additional functions that are useful in the context of a mobile automation test through the Appium server.

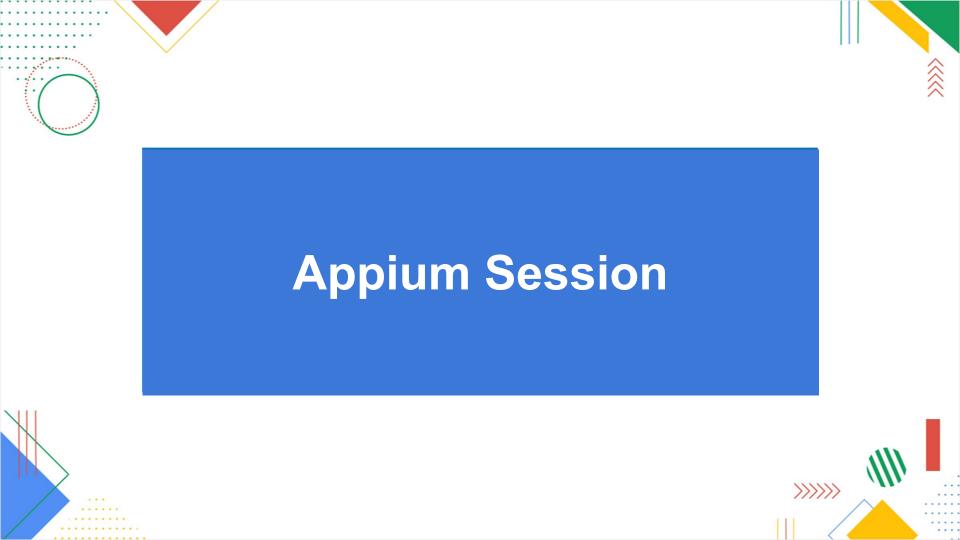


#### **AndroidDriver**

- This driver class inherits from AppiumDriver, but it adds in additional functions that are useful in the context of a mobile automation test on Android devices through Appium.
- Only use this driver class if you want to start a test on an Android device or Android emulator.

#### **IOSDriver**

- This driver class inherits from AppiumDriver, but it adds in additional functions that are useful in the context of a mobile automation test on iOS devices through Appium.
- Only use this driver class if you want to start a test on an iOS device or iOS emulator.





# **General Capabilities**

Capability	Description	Values
automationName	Which automation engine to use	Appium (default) or Selendroid or UiAutomator2 for Android or XCUITest for iOS
platformName	Which mobile OS platform to use	iOS, Android, or FirefoxOS
platformVersion	Mobile OS version	e.g., 7.1, 4.4
deviceName	The kind of mobile device or emulator to use	iPhone Simulator, iPad Simulator, iPhone Retina 4-inch, Android Emulator, Galaxy S4, etc.
orientation	(Sim/Emu-only) start in a certain orientation	LANDSCAPE or PORTRAIT



# **Android Capabilities**

appActivity	Activity name for the Android activity you want to launch from your package.	MainActivity, .Settings
appPackage	Java package of the Android app you want to run. By default this capability is received from the package manifest	com.example.android.myApp
androidInstallPath	The name of the directory on the device in which the apk will be push before install. Defaults to /data/local/tmp	e.g. /sdcard/Downloads/



# iOS Capabilities

bundleId	Bundle ID of the app under test. Useful for starting an app on a real device or for using other caps which require the bundle ID during test startup.	e.g. io.appium.TestApp
udid	Unique device identifier of the connected physical device	e.g. 1ae203187fc012g
appName	The display name of the application under test. Used to automate backgrounding the app in iOS 9+.	e.g., UlCatalog

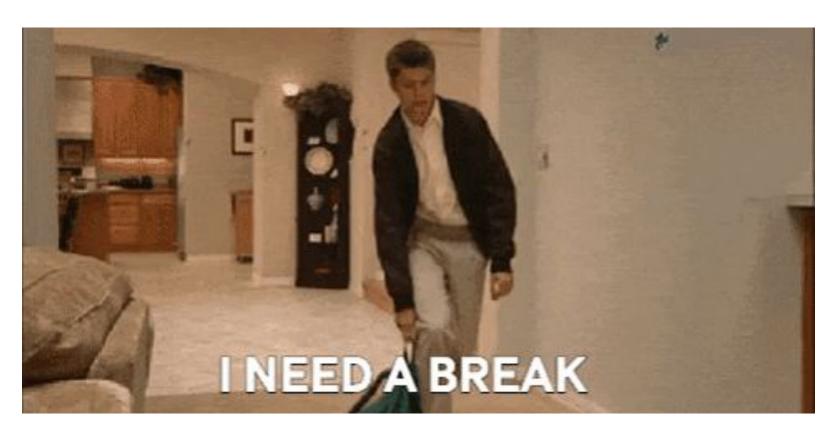


# Other Capabilities

Android and iOS other specific capabilities are listed here,

http://appium.io/docs/en/writing-running-appium/caps/







### **Session Timeout**

### There is two types of wait

- Implicit Wait
- Explicit Wait





# **Implicit Wait**

Implicit Wait means informing appium driver to wait for specific amount of time and if the element is not visible after waiting for that specific time then throw "NoSuchElementException".

```
driver.manage().timeouts().implicitlyWait(30,TimeUnit.SECONDS);
```



# **Explicit Wait**

In Explicit wait along with wait time, we also provide the wait condition. It will wait till the condition or the maximum wait time provided before throwing the Exception"ElementNotVisibleException".

```
WebDriverWait wait = new WebDriverWait(driver, 5);
WebElement element = wait.until(ExpectedConditions
.visibilityOfElementLocated(By.xpath("<xpath>")));
```



### **Session Orientation**

Using the rotate method you can change screen orientation LANDSCAPE to PORTRAIT or vice versa.

```
driver.rotate(ScreenOrientation.LANDSCAPE);
driver.rotate(ScreenOrientation.PORTRAIT);
ScreenOrientation orientation = driver.getOrientation();
```



### **Take Screenshot**

Takes a screenshot of the viewport in a native context (iOS, Android) and takes a screenshot of the window in web context.

```
File scrFile = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
```

Window flag: treat the content of the window as secure, preventing it from appearing in screenshots or from being viewed on non-secure displays.

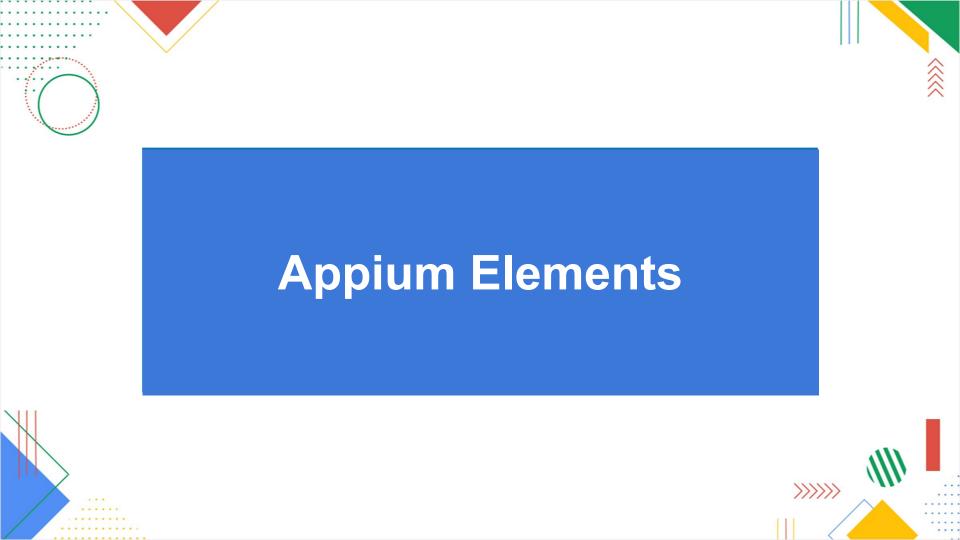
```
getWindow().setFlags(LayoutParams.FLAG_SECURE, LayoutParams.FLAG_SECURE);
```



## Page Source

- In a web context, the source returns the source HTML of the current window. In a native context (iOS, Android, etc...) it will return the application hierarchy XML.
- (NOTE: iOS and Android don't have standard ways of defining their application source, so on calls to 'Get Page Source' Applied traverses the app hierarchy and creates an XML document. Thus, getting the source can often be an expensive and time-consuming operation)

```
String source = driver.getPageSource();
```





#### What is XPath?

- XPath (XML Path Language) is a query language for selecting nodes from an XML document.
- In addition, XPath may be used to compute values (e.g., strings, numbers, or Boolean values) from the content of an XML document.
- XPath was defined by the World Wide Web Consortium (W3C)



## Find Element Using XPath

#### XPath for the text, button etc...

```
//android.widget.Button
[contains(@resource-id,'<_resourceId_>')]
```

#### XPath for the list items etc...

```
//android.support.v7.widget.RecyclerView
[@index='<_childIndex_>']/<_childClsName_>
```

#### XPath for the scrolling

```
new UiScrollable(new UiSelector().resourceId("<_resourceId_>"))
.scrollIntoView(new UiSelector().text("<_childText_>"))
```



### **Click Actions**

#### **Single Click**

```
WebElement element = driver.findElementByXPath(makeXPath(
Const.Button, "button1"));
element.click();
Long Press
WebElement element = driver.findElementByXPath(makeXPath(
Const.RecyclerView, 0, Const.FrameLayout));
new Actions(driver).clickAndHold(element).perform();
```



# **Send Keys Actions**

Send a sequence of key strokes to an element.

```
e.g,
```

```
WebElement element = driver.findElementByXPath(makeXPath(
Const.EditText, "edEmail"));
element.sendKeys("kishandonga.92@gmail.com");
```



#### **Actions With Attributes**

```
WebElement element = driver.findElementByXPath(makeXPath(
Const.EditText, "edEmail"));

String elText = element.getText();
String tagName = element.getTagName();
boolean isSelected = element.isSelected();
boolean isEnabled = element.isEnabled();
boolean isDisplayed = element.isDisplayed();
Point location = element.getLocation();
```



### **Touch Actions**

Appium mobile actions such as swipe, tap, press, multi-touch based on touch.

```
TouchActions action = new TouchActions(driver);
action.singleTap(element);
action.doubleTap(element);
action.down(10, 10);
action.moveTo(50, 50);
action.move(50, 50);
```



# What is difference between WebElement, MobileElement, AndroidElement, and IOSElement?

WebElement is a general interface that all the other classes (RemoteWebElement, MobileElement, AndroidElement, IOSElement) implement. which means that if you have an object of type WebElement it will have all the basic functions a web element can have. If you need more specific functions you can use other objects like: RemoteWebElement, MobileElement, AndroidElement, IOSElement

Also note that WebElement & RemoteWebElement are bundled with Selenium while MobileElement, AndroidElement & IOSElement are specific to Appium.









```
Start an Android activity by providing package name and activity name driver.startActivity(
new Activity("com.example", "ActivityName"));

Get the name of the current Android activity
```

Get the name of the current Android package
String package = driver.getCurrentPackage();

String activity = driver.currentActivity();

## Android IOS



```
driver.installApp("/Users/johndoe/path/to/app.apk");
driver.isAppInstalled("com.example.AppName");
driver.launchApp();
driver.runAppInBackground(Duration.ofSeconds(10));
driver.closeApp();
driver.resetApp();
driver.removeApp("com.example.AppName");
driver.activateApp('com.apple.Preferences');
driver.terminateApp('io.appium.android.apis');
```

0 is not installed. 1 is not running. 2 is running in background or suspended. 3 is running in background. 4 is running in foreground.

```
driver.queryAppState('io.appium.android.apis');
```



#### Set the content of the system clipboard

```
//base64Content is Base64-encoded content
driver.setClipboard("label",
ClipboardContentType.PLAINTEXT, base64Content);
driver.setClipboardText("happy testing");
```

#### Get the content of the system clipboard

```
driver.getClipboard(ClipboardContentType.PLAINTEXT); // get plaintext
driver.getClipboardText();
```

# Android Emulator



For Android emulator. To set the state of the battery charger to connected or not. driver.setPowerAC(PowerACState.OFF);

For Android emulator. To set the battery percentage. driver.setPowerCapacity(100);



#### **Push File**

Place a file onto the device in a particular place

```
driver.pushFile("/path/to/device/foo.bar",
new File("/Users/johndoe/files/foo.bar"));
```

#### **Pull File**

Retrieve a file from the device's file system

```
byte[] fileBase64 = driver.pullFile("/path/to/device/foo.bar");
```

#### **Android IOS**

### Interaction

```
driver.shake();
driver.lockDevice();
driver.unlockDevice();
boolean isLocked = driver.isDeviceLocked();
driver.rotate(new DeviceRotation(10, 10, 10));
```



Because we completed 50 slides.





Thanks guys It is my pleasure.





#### **Press Key Code**

Press a particular key on an Android Device

driver.pressKeyCode(AndroidKeyCode.SPACE,
AndroidKeyMetastate.META\_SHIFT\_ON);

#### **Long Press Key Code**

Press and hold a particular key code on an Android device

driver.longPressKeyCode(AndroidKeyCode.HOME);



#### **Hide Keyboard**

Hide soft keyboard

driver.hideKeyboard();

#### Is Keyboard Shown

Whether or not the soft keyboard is shown

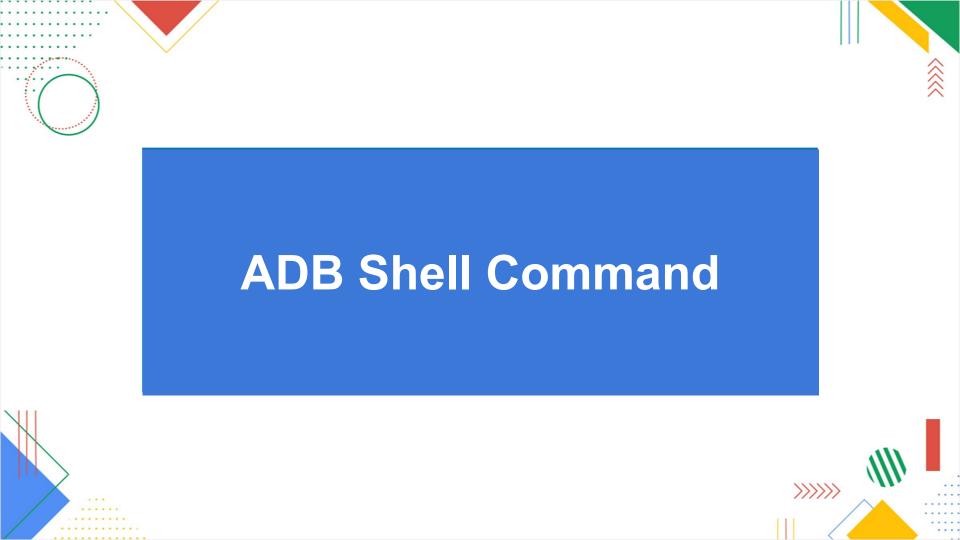
boolean isKeyboardShown = driver.isKeyboardShown();



# Android Network

```
driver.toggleAirplaneMode();
driver.toggleWifi();
driver.toggleLocationServices();
driver.toggleData();
```

Note: This API does not work for Android API level 21+ because it requires system or carrier privileged permission, and Android <= 21 does not support granting permissions.





### adb Shell

Check Device attached or not

Type the command adb devices in the adb shell

Now, type the command adb shell which is located to devices root

Now, type the below mentioned command to get the information of the focused application:

dumpsys window windows | grep -E 'mCurrentFocus|mFocusedApp'

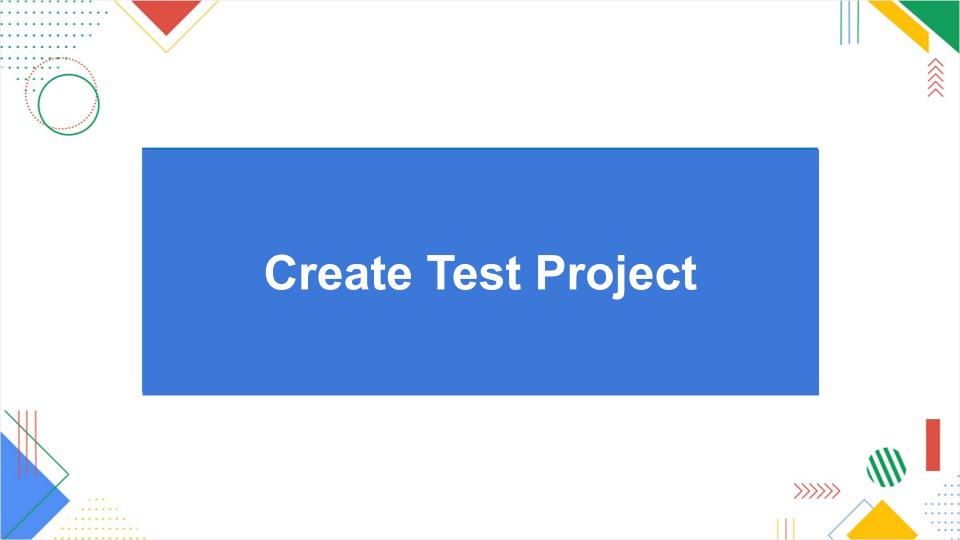






# **Fruit Basket**







# GO Live For Code Lab



# Your Feedback And Question Answers...



# kishandonga.92@gmail.com



@ikishan92



**Kishan Donga** 



ikishan92

