Mobile Testing Interview Questions

Mobile Testing is a process where application software developed for handheld mobile devices is tested for its functionality, usability and consistency.

Types of Application:

- **Native Applications:** Native applications are developed by keeping a certain platform in mind. Native apps are developed for use on a specific device and can be installed from the App Store, such as Google Play Store or Apple's App Store. They can work offline and can also use the device notification system.
- **Web Applications:** Web applications are not real applications; they are websites that run on browsers.
- **Hybrid Applications:** Hybrid application is a combination of native and web applications. It can run on any platform.

Types of Files:

- **Android** uses .apk files
- **iOS** uses .ipa files
- **Windows**

How will you get your builds from developer?

- For Android, Developers will provide builds through Teams.
- For iOS, Developers will upload in Test Flight App and we will install it in our device.

Tools for Mobile Automation:

- **Appium** - It's an Open-Source and most commonly used tool, we can connect only Android devices through Windows, For iOS devices we need mac machine.
- **Appium Studio** - It's a Paid Version, we can connect both iOS and Android devices.

Devices Test:

- **Real Device** - Live Devices connected through USB with our system.
- **Emulator** - which is virtual Android device runs slower than Real device and it imitates features of another device and it resembles both software and hardware of a device.
- **Simulator** - which is virtual iOS Device runs faster than Real device. It replicates software by resembling various product configuration that exist in real environment. We need XCode to run simulator.

Appium Configuration:

- First Download Appium & Install
- Then download android studio & Install
- Then open SDK Manager and install Android versions
- Then click SDK Manager
- Then download latest Android Versions
- Then set the ANDROID_HOME
- Then set the path
- Then connect the mobile device (Real Device/Emulator) in machine
- To check Android studio is installed properly, go to command prompt/terminal and enter adb
- To check whether devices are connected with a system, go to command prompt/terminal and enter adb devices

Appium Tool Steps for Object spy:

- First, we need to launch Appium & Start Server
- Then we need to click the start inspect session
- Then set the desired capabilities (Which app want to inspect like device name, platform name, platform version app details)
- Desired capabilities should be given in key, value combination (JSON)
- Then we need to click the start session
- Then we need to click the select element in the screen and move the cursor where you want to spy.
- Then form the xpath /id/text
- Then verify the locator in search for element option.
- Then paste the locator and check the locator is matching.

Note: Appium Navigations:

- Select Elements – To Inspect
- Tap By Coordinates – To click
- Refresh source and screenshot - To refresh a page
- Swipe By Coordinates – To perform Scrolling and swiping

Appium Driver:

Selenium Package -> Remote WebDriver -> Appium Driver -> Android and iOS Driver

Ways to declaring the drivers:

- WebDriver driver = new AppiumDriver();
- AppiumDriver driver = new AppiumDriver();
- AppiumDriver driver = new AndroidDriver();
- AndroidDriver driver = new AndroidDriver();
- AppiumDriver driver = new IOSDriver();
- IOSDriver driver = new IOSDriver();

Application launching in Automation through Appium:

- We need to add java client dependency in pom.xml.
- Then need to set the DesiredCapablities like
    - “deviceName”: “particular device name”
    - “platformName”: “Android / IOS”
    - “platformVersion”: “particular version”
    - “appPackage”: “particular appPackage”
    - “appName”: “particular appName”
- Then declare AppiumDriver And provide Url --> http://localhost:4723/wd/hub

Appium Architecture:

- First the Appium Client which is the interface that we testers interacts with. It's a piece of code (written in programming languages like Java, Python, Ruby, JavaScript, etc.) that sends test scripts to Appium Server.
- Then there is Appium Server acts as a bridge between the Appium client and the mobile device like emulator or real device. It accepts requests from the Appium Client, then processes and communicates with the mobile device to execute the automation commands.
- Then the Appium server communicates with the device using mobile drivers like UIAutomator2 for Android or XCUITest for iOS.
- Appium will be using the JSON Wire Protocol to send commands between the Appium Client and Appium Server.

TouchAction:

TouchAction class is used to handle Mobile actions like tap, swipe, longPress, press, drag and drop, scrollup and Scrolldown. When we are using any methods in TouchAction class, we need to call perform() method

Methods:

- longPress() is used to press and hold at the center of an element until the context menu event has fired.
- press() is used to perform the press action on the screen.
- perform() is used to perform all the chain of actions.
- release() is used to remove the current touching implement from the screen that is withdraw our touch.
- moveTo() is used to move current touch to a new position.
- tap() is used to tap on a position.
- waitAction() is used to waits for specified amount of time to pass before continue to next touchaction.

Features of Appium:

- Appium has multi-platform support.
- Appium allows the parallel execution of test scripts.
- Appium supports various languages like C#, Python, Java, Ruby, PHP, JavaScript with node.js, and many others that have Selenium client library.

Advantages of Appium:

- Appium is an open-source tool.
- It allows the automated testing of hybrid, native, and web applications.
- Appium is a cross-platform tool.

Disadvantages of Appium:

- Since Appium depend on the remote web driver, so it is a bit slow
- In iOS, only one instance can run on one Mac OS device, which means one test can be executed at a time per Mac. If you want to run your tests on multiple iOS devices at the same time, you need to arrange the same number of Mac machines. But it would be expensive to arrange various Mac machines.

Limitations of using Appium:

- Appium does not support testing of Android Version lower than 4.2
- Limited support for hybrid app testing. E.g., not possible to test the switching action of application from the web app to native and vice-versa
- No support to run Appium Inspector on Microsoft Windows

Methods to Scroll down and scroll up:

```
public static void scrollDown(){
        //mobile window size
Dimension size = driver.manage().window().getSize();
        //get start height
Double startHeight = size.getHeight() * 0.7;
        //convert double to int
int start = startHeight.intValue();
        //get end height
Double endHeight = size.getHeight() * 0.2;
        //convert double to int
int end = startHeight.intValue();
        //scrolldown
TouchAction tc = new TouchAction(driver);
tc.press(PointOption.point(0, start)).moveTo(PointOption.point(0, end)).release().perform();
}

public static void scrollUp(){
        //mobile window size
Dimension size = driver.manage().window().getSize();
        //get start height
Double startHeight = size.getHeight() * 0.2;
        //convert double to int
int start = startHeight.intValue();
        //get end height
Double endHeight = size.getHeight() * 0.7;
        //convert double to int
int end = startHeight.intValue();
        //scrolldown
TouchAction tc = new TouchAction(driver);
tc.press(PointOption.point(0, start)).moveTo(PointOption.point(0, end)).release().perform();
}
```

How to switch from the mobile app context to the web view context using Appium?

- First, we need to get the available contexts using the getContextHandles() method of the Appium driver
- Then we need to iterate through the contexts and identify the web view context
- Then we can switch to the web view context using the context() method of the Appium driver

Syntax:

```
        //Assuming you have an instantiated driver object called 'driver'
        //Get the available contexts
Set<String> contextHandles = driver.getContextHandles();
        //Iterate through the contexts to find the web view context
for (String contextHandle: contextHandles) {
   if (contextHandle.contains("WEBVIEW")) {
        //Switch to the web view context
      driver.context(contextHandle);
      break;
   }
}
```

- driver.getContextHandles() retrieves all the available contexts for the current session.
- The code then iterates through the contexts and checks if the context handle contains "WEBVIEW".
- Once the web view context is identified, the code switches to that context using driver.context(contextHandle).
- The loop is then broken to avoid unnecessary iterations.

Launch app in Automation through Appium:

```
DesiredCapabilities cap = new DesiredCapabilities();
cap.setCapability("deviceName", "particular device name");
cap.setCapability("platformName", "particular platform name");
cap.setCapability("platformVersion", "particular platform version");
cap.setCapability("appPackage", "particular app package");
cap.setCapability("appActivity", "particular app activity");


URL url = new URL("http://localhost:4723/wd/hub");
AppiumDriver<MobileElement> driver = new AppiumDriver<MobileElement>(url, cap);
```