STANDARD OPERATING PROCEDURE

Setup Guide for Mobile Automation

1. **Purpose:** The purpose of this SOP is to setup the Mobile Automation Framework.

2. **Scope:** This SOP covers all necessary steps, tools, and processes required for setting up the Mobile Automation Framework. It applies to all team members involved in automation testing for mobile applications, including configuring the environment, installing dependencies, setting up emulators or physical devices.

Document Details	
Reference Number	SOP-MA-001
Version	0.1
Effective Date	01-NOV-2024
Prepared By	Shivom Kumari, Rohit Singh
Reviewed By	Ankur Kumar
Approved By	Anupam Anand

Contents

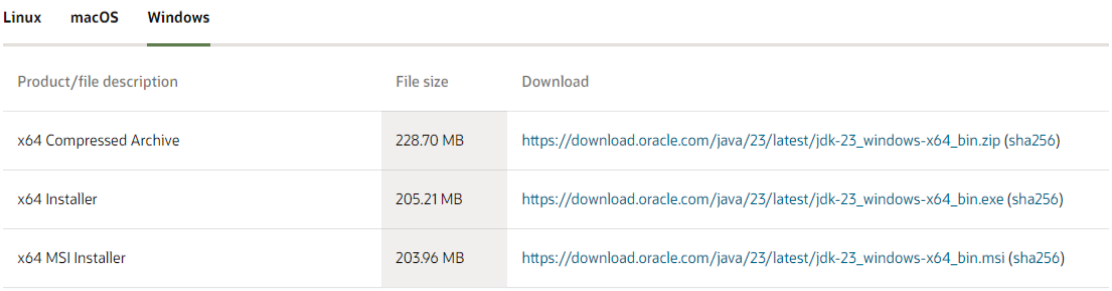
Objective

- The objective of preparing a mobile automation setup guide is to provide clear and comprehensive instructions for establishing an automated testing environment for mobile applications.
- This document aims to standardize the setup process, ensuring consistency across teams while simplifying the experience for both beginners and experienced testers.
- It will recommend suitable tools and frameworks, detail configuration steps for various devices and operating systems, and address common challenges with troubleshooting tips.

Java

- Java's primary purpose is to allow developers to write code once and run it anywhere.
- Java is designed to follow object-oriented programming principles, making it easier to manage and organize code.
- Java provides strong memory management, exception handling, and type checking.

1. **Download Java JDK**:
    - Go to Oracle's JDK download page.
    - Follow the installation instructions.



Linux macOS Windows

Product/file description	File size	Download
x64 Compressed Archive	228.70 MB	https://download.oracle.com/java/23/latest/jdk-23_windows-x64_bin.zip (sha256)
x64 Installer	205.21 MB	https://download.oracle.com/java/23/latest/jdk-23_windows-x64_bin.exe (sha256)
x64 MSI Installer	203.96 MB	https://download.oracle.com/java/23/latest/jdk-23_windows-x64_bin.msi (sha256)

2. **Set Environment Variables**:
    - Add JAVA_HOME to your system environment variables pointing to the JDK installation directory.
    - Add %JAVA_HOME%\bin to the PATH variable.

3. **Download Eclipse**

- Visit **Eclipse Downloads** and install.

Node.js

- Node.js enables building scalable and high-performance web servers and applications using JavaScript.
- Node.js is widely used in testing for various purposes, primarily because of its non-blocking architecture and the ability to leverage JavaScript across the stack.

1. **Download Node.js**:
    - Go to Node.js official website.
    - Download and install the LTS version.
2. **Verify Installation**:

- Open a terminal and run:

```
node -v
npm -v
```

Appium

- Appium is an open-source test automation framework primarily used for mobile applications.
- Appium allows you to write tests for both iOS and Android applications using a single codebase.

1. **Install Appium**:

- Open a terminal and run: npm install -g Appium

2. **Install Ulautomator**

- Open a terminal and run: Appium driver install uiautomator2

3. **Appium Inspector**

- **Click on Releases · appium/appium-inspector (github.com)**
- **Link to download Appium Inspector**

  Appium-Inspector-2024.9.1-win-arm64.exe

Android Studio (for Android testing)

- Android Studio is the official integrated development environment (IDE) for Android app development.
- It includes an Android Emulator for testing applications on various virtual devices, helping developers simulate different device configurations and screen sizes.

1. **Download Android Studio:**
    - Go to **Android Studio download page**.
    - Install following the setup wizard.
2. **Set Up Android SDK:**
    - Open Android Studio and go to SDK Manager.
    - Install the required SDK platforms and tools.
3. **Set Environment Variables:**
    - Add ANDROID_HOME pointing to the Android SDK directory.
    - Add %ANDROID_HOME%\tools and %ANDROID_HOME%\platform-tools to your PATH.

Adding pom.xml Dependency

Find Dependencies from here
Maven Repository: Search/Browse/Explore

Selenium:

https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java

Testng:

https://mvnrepository.com/artifact/org.testng/testng

WebDriver:

https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager

Artifact (Apache POI):

https://mvnrepository.com/artifact/org.apache.poi/poi

XMLbeans:

https://mvnrepository.com/artifact/org.apache.xmlbeans/xmlbeans

Apache common collections:

https://mvnrepository.com/artifact/org.apache.commons/commons-collections4

Apache commons IO:

https://mvnrepository.com/artifact/commons-io/commons-io

Apache Log4j Core:

https://mvnrepository.com/artifact/org.apache.logging.log4j/log4j-core

ExtentReports:

https://mvnrepository.com/artifact/com.aventstack/extentreports

Current Folder Structure

Base

- The Base class acts as the foundation of the framework. It initializes essential elements and configurations required across tests:
- Driver Initialization: Creates and manages a single WebDriver instance.
- Setup and Teardown: Contains methods to set up and tear down test environments before and after each test.
- Configurations: Reads configurations (APK file, Platform Name, Device Name, etc) from config files and applies them to initialize the test environment.
- Logging: Sets up loggers to capture test logs for debugging and tracking test execution flow.

Testcase

- This is the class where individual test cases are written. It inherits from the Base class to gain access to the driver, configurations, and setup/teardown functionalities.
- Test Method Structure: Each test method is defined as a separate function, with steps following the Arrange-Act-Assert pattern for readability.
- Assertions: Includes various assertion checks to validate test outcomes.
- Reporting: Integrates with reporting tools like Extent Reports for visual test reports.

Pages

- The Pages package implements the Page Object Model (POM) pattern, where each page of the application has a corresponding class:
- Page Classes: Each page class represents a specific page, such as LoginPage.
- Locators and Methods: Contains locators and methods specific to interactions on that page.

- Encapsulation: By isolating page interactions in these classes, test scripts are more readable, and maintenance is simpler if the UI changes.

Utilities

- The Utilities package includes helper classes and methods that can be reused across tests:
- Screenshot: Captures screenshots in case of test failure or for verification purposes.
- ReadXLData: Reads data from Excel files if the test cases involve data-driven testing.

TestData

- Includes files like Excel sheets or JSON files containing test data values.
- This enables data-driven testing, where test cases can run with multiple data sets.
- Config Files
- Stores environment configurations such as config.properties, which includes base URLs, credentials, and timeouts.
- These are read by the Base class to initialize the test environment.

Logs

- The framework is configured to generate log files that capture detailed information about test execution, including debug, info, and error messages.
- This helps with tracing and debugging issues in the test cases.