|  |  |
| --- | --- |
| **Practical No: 01** | **Write setup steps regarding installation of Java, Android Studio**  **and Configuration of SDK with virtual devices for Mobile**  **Testing.** |
|  | 1. **Installation of Java**   -> **A: First Check for Java**  1 - Click on Windows + R    **2.** Type cmd and press enter  **3.** Type java — version or java orwhere java    -> **B: Download Java**   * Visit this site - **https://www.oracle.com/in/java/technologies/downloads/**   **-> C: Install Java**  - On the Downloads Page, click x64 installer Download Link under the Windows Category.     * After downloading the installation file, Double-click the downloaded file to start the installation. * Click Next to proceed to the next step. * Choose the destination folder for the Java installation files or stick to the default path. Click Next to proceed. * Wait for the wizard to finish the installation process until the Successfully Installed message appears. Click **Close** to exit the wizard.  **-> D: Set Environmental Variables for Java**  * Press Windows menu. * Type env and press enter. * Click on Environment Variables. * In User variables, Click on New. * Variable name: JAVA\_HOME * Variable Value: C:\Program Files\Java\jdk-21 * Click OK. * In System variables, Click on Path. * Click on Edit. Click on New & paste path C:\Program Files\Java\jdk-21\bin in it.  1. **Installation of Android Studio** 2. **Visit this site and Download Android Studio**  * https://developer.android.com/studio  1. **Set Environmental Variables for Android Studio**  * Press Windows menu and Type env and press enter. * Click on Environment Variables. * In User variables, click on Path and Click on Edit. * Click on New & paste path   C:\Users\AMPICS\AppData\Local\Android\Sdk\platform-tools.   * In System variables, Click on New. * Variable name: ANDROID\_HOME * Variable Value: C:\Users\AMPICS\AppData\Local\Android\Sdk * Click OK. * In System variables, click on Path and click on Edit.   - Click on New & paste following paths in it (one by one).  - **C:\Users\parth\AppData\Local\Android\Sdk\platform-tools**  **C:\Users\parth\AppData\Local\Android\Sdk\tools**  **C:\Users\parth\AppData\Local\Android\Sdk\tools\bin**  **3. Check for Android Studio**  - Open Command Prompt and type **where android**    **4. Configuration of SDK**  **-** Open Android Studio and Click on Tools -> SDK Manager.    - Click on Tools -> Device Manager and Create your Virtual Device as per your needs.  - Launch the Virtual Device & test for working. |

|  |  |
| --- | --- |
| **Practical No: 02** | **Write setup steps regarding Node JS, NPM, Appium or Appium**  **Desktop, Appium Doctor with Appium Inspector for Mobile**  **Testing.** |
|  | **A – Check for Node.js and NPM**  - Open Command Prompt  - For Node.js – type node — version or node –v or where node  - For NPM – type npm — version or npm –v or where npm  **=> For Node.js**    **=> For NPM**    **B – Download Node.js and NPM**  - Visit this website - <https://nodejs.org/en/download/prebuilt-installer>  - Click on Prebuilt Installer and Download Node.js button.  -  **C – Download and Install Appium**  - To install Appium, run the following command.    **D – Check Appium**  - Open Command Prompt and type below or **appium — version or where appium**, you will get version    **E – Download and Install Appium-Doctor**  - To install Appium Doctor, run the following command.    **F – Check Appium-Doctor**  - Open Command Prompt and Type Appium-doctor — version or where Appium-doctor.    **G – Install Appium-Inspector**  - Open Command Prompt and type below code in it. |

|  |  |
| --- | --- |
| **Practical No: 03** | **Create an Appium Maven project with Java client dependencies.** |
|  | To create an Appium Maven project with Java client dependencies.  1. Create a Maven Project:  - Open Eclipse IDE  - Click on New -> Others -> Select Maven Project    - Select Project name and location    - Fill the details    - Open pom.xml    - Write this in the pom.xml file and add the following content:  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>practical3</groupId>  <artifactId>pra3</artifactId>  <version>0.0.1-SNAPSHOT</version>  <description>this is practical 3</description>    <!-- https://mvnrepository.com/artifact/io.appium/java-client -->  <dependency>  <groupId>io.appium</groupId>  <artifactId>java-client</artifactId>  <version>9.3.0</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->  <dependency>  <groupId>org.seleniumhq.selenium</groupId>  <artifactId>selenium-java</artifactId>  <version>4.25.0</version>  </dependency>  </project> |

|  |  |
| --- | --- |
| **Practical No: 04** | **Write a script in Java to create Android Driver and set Desired**  **capabilities through UiAutomator object.** |
|  | package pra4;  import io.appium.java\_client.android.AndroidDriver;  import org.openqa.selenium.remote.DesiredCapabilities;  import java.net.MalformedURLException;  import java.net.URL;  public class Pra4 {  public static void main(String[] args) {    DesiredCapabilities caps = new DesiredCapabilities();  caps.setCapability("platformName", "Android");  caps.setCapability("deviceName", "Android Emulator");  caps.setCapability("automationName", "UiAutomator2");    System.out.println("Hello Appium");  }  } |
|  | <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>practical4</groupId>  <artifactId>pra4</artifactId>  <version>0.0.1-SNAPSHOT</version>  <description>this is practical 4</description>    <dependencies>  <!-- https://mvnrepository.com/artifact/io.appium/java-client -->  <dependency>  <groupId>io.appium</groupId>  <artifactId>java-client</artifactId>  <version>9.2.2</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->  <dependency>  <groupId>org.seleniumhq.selenium</groupId>  <artifactId>selenium-java</artifactId>  <version>4.25.0</version>  </dependency>  </dependencies>    </project> |
|  | from appium import webdriver  def main():  desired\_caps = {  "platformName": "Android",  "deviceName": "Android Emulator",  "automationName": "UiAutomator2"  }  try:  driver = webdriver.Remote("http://127.0.0.1:4723 ", desired\_caps)  print("Hello Appium!")  except Exception as e:  print(f"An error occurred: {e}")  finally:  if 'driver' in locals():  driver.quit()  if \_\_name\_\_ == "\_\_main\_\_":  main() |
| Input | NA |
| Output |  |

|  |  |
| --- | --- |
| **Practical No: 05** | **Write a script in java to execute “www.google.com” in chrome**  **using Appium.** |
|  | <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>Practical5</groupId>  <artifactId>pra5</artifactId>  <version>0.0.1-SNAPSHOT</version>    <dependencies>  <!-- Appium Java Client -->  <dependency>  <groupId>io.appium</groupId>  <artifactId>java-client</artifactId>  <version>9.2.2</version>  </dependency>  <!-- Selenium Java -->  <dependency>  <groupId>org.seleniumhq.selenium</groupId>  <artifactId>selenium-java</artifactId>  <version>4.25.0</version>  </dependency>  </dependencies>  </project> |
|  | package pra5;  import io.appium.java\_client.android.AndroidDriver;  import org.openqa.selenium.By;  import org.openqa.selenium.WebElement;  import org.openqa.selenium.remote.DesiredCapabilities;  import org.openqa.selenium.support.ui.ExpectedConditions;  import org.openqa.selenium.support.ui.WebDriverWait;  import java.net.MalformedURLException;  import java.net.URL;  import java.time.Duration;  public class Pra5 {  private static AndroidDriver driver;  public static void main(String[] args) throws MalformedURLException, InterruptedException {  // Set Desired Capabilities  DesiredCapabilities caps = new DesiredCapabilities();  caps.setCapability("deviceName", "Android Emulator");  caps.setCapability("platformName", "Android");  caps.setCapability("platformVersion", "14.0");  caps.setCapability("appPackage", "com.android.chrome");  caps.setCapability("appActivity", "com.google.android.apps.chrome.Main");  caps.setCapability("automationName", "UiAutomator2");  // Create a new instance of the Appium Driver  driver = new AndroidDriver(new URL("http://127.0.0.1:4723"), caps);  // Create an instance of WebDriverWait  WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(60));  // Wait for the Chrome app to open  Thread.sleep(5000);  // Handle the "Use without an account" button (if it's displayed)  WebElement useWithoutAccountButton = wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//android.widget.Button[@text='Use without an account']")));  useWithoutAccountButton.click();    // Handle the "No thanks" popup (if it's displayed)  WebElement noThanksButton = wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//android.widget.Button[@text='No thanks']")));  noThanksButton.click();  // Now navigate to Google  driver.get("https://www.google.com");  System.out.println("Visited google");    driver.quit();  }  } |
|  | from appium import webdriver  from selenium.webdriver.common.by import By  from selenium.webdriver.support.ui import WebDriverWait  from selenium.webdriver.support import expected\_conditions as EC  import time  def main():  # Desired capabilities  desired\_caps = {  "deviceName": "Android Emulator",  "platformName": "Android",  "platformVersion": "14.0",  "appPackage": "com.android.chrome",  "appActivity": "com.google.android.apps.chrome.Main",  "automationName": "UiAutomator2"  }  try:  driver = webdriver.Remote("http://127.0.0.1:4723", desired\_caps)  # Wait for the Chrome app to open  time.sleep(5)  # Create WebDriverWait instance with 60 seconds timeout  wait = WebDriverWait(driver, 60)    try:  use\_without\_account\_button = wait.until(  EC.visibility\_of\_element\_located((By.XPATH, "//android.widget.Button[@text='Use without an account']"))  )  use\_without\_account\_button.click()  except Exception as e:  print("No 'Use without an account' button found.")  try:  no\_thanks\_button = wait.until(  EC.visibility\_of\_element\_located((By.XPATH, "//android.widget.Button[@text='No thanks']"))  )  no\_thanks\_button.click()  except Exception as e:  print("No 'No thanks' button found.")    driver.get("https://www.google.com")  print("Visited Google.")  except Exception as e:  print(f"An error occurred: {e}")  finally:    if 'driver' in locals():  driver.quit()  if \_\_name\_\_ == "\_\_main\_\_":  main() |
| Input | NA |
| Output |  |

|  |  |
| --- | --- |
| **Practical No: 06** | **Write a script in Java to run HelloWorld.apk using Appium.** |
|  | <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>Practical6</groupId>  <artifactId>pra6</artifactId>  <version>0.0.1-SNAPSHOT</version>    <dependencies>  <!-- Appium Java Client -->  <dependency>  <groupId>io.appium</groupId>  <artifactId>java-client</artifactId>  <version>9.2.2</version>  </dependency>  <!-- Selenium Java -->  <dependency>  <groupId>org.seleniumhq.selenium</groupId>  <artifactId>selenium-java</artifactId>  <version>4.25.0</version>  </dependency>  <dependency>  <groupId>practical4</groupId>  <artifactId>pra4</artifactId>  <version>0.0.1-SNAPSHOT</version>  </dependency>  </dependencies>  </project> |
|  | package pra6;  import java.net.MalformedURLException;  import java.net.URL;  import io.appium.java\_client.AppiumDriver;  import io.appium.java\_client.android.AndroidDriver;  import io.appium.java\_client.android.options.UiAutomator2Options;  import io.appium.java\_client.service.local.AppiumServiceBuilder;  import io.appium.java\_client.service.local.AppiumDriverLocalService;  import io.appium.java\_client.service.local.flags.GeneralServerFlag;  public class Pra6 {  public static void main(String[] args) throws MalformedURLException {    AppiumDriverLocalService appiumService = new AppiumServiceBuilder()  .usingAnyFreePort()  .withArgument(GeneralServerFlag.LOG\_LEVEL, "info")  .build();  appiumService.start();    UiAutomator2Options options = new UiAutomator2Options();  options.setPlatformName("Android");  options.setDeviceName("Android Emulator");  options.setAutomationName("UiAutomator2");  options.setApp("C:\\MASTER OF COMPUTER APPLICATIONS (MCA)\\MCA SEM 3\\MOBILE TESTING\\dummy\\app\\build\\outputs\\apk\\debug\\app-debug.apk");  // Initialize AndroidDriver with options  AppiumDriver driver = new AndroidDriver(new URL(appiumService.getUrl().toString()), options);  // Print success message  System.out.println("HelloWorld.apk is launched successfully!");    // Quit the driver and stop the Appium server  driver.quit();  appiumService.stop();  }  } |
|  | from appium import webdriver  from appium.webdriver.appium\_service import AppiumService  import time  def main():    appium\_service = AppiumService()  appium\_service.start(  args=['--base-path', '/wd/hub', '--log-level', 'info']  )  print(f"Appium Server started at: {appium\_service.service\_url}")  desired\_caps = {  "platformName": "Android",  "deviceName": "Android Emulator",  "automationName": "UiAutomator2",  "app": "C:\\MASTER OF COMPUTER APPLICATIONS (MCA)\\MCA SEM 3\\MOBILE TESTING\\dummy\\app\\build\\outputs\\apk\\debug\\app-debug.apk"  }    try:  driver = webdriver.Remote(appium\_service.service\_url, desired\_caps)  print("HelloWorld.apk is launched successfully!")  except Exception as e:  print(f"An error occurred: {e}")  finally:    if 'driver' in locals():  driver.quit()  appium\_service.stop()  print("Appium Server stopped.")  if \_\_name\_\_ == "\_\_main\_\_":  main() |
| Input | NA |
| Output |  |

|  |  |
| --- | --- |
| **Practical No: 07** | **Write a script in Java to demonstrate the assert (Verify if the**  **element contains the expected text ) in apk using Appium.** |
|  | <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>Practical7</groupId>  <artifactId>pra7</artifactId>  <version>0.0.1-SNAPSHOT</version>    <dependencies>  <!-- Appium Java Client -->  <dependency>  <groupId>io.appium</groupId>  <artifactId>java-client</artifactId>  <version>9.2.2</version>  </dependency>  <!-- Selenium Java -->  <dependency>  <groupId>org.seleniumhq.selenium</groupId>  <artifactId>selenium-java</artifactId>  <version>4.25.0</version>  </dependency>  </dependencies>  </project> |
|  | package pra7;  import io.appium.java\_client.android.AndroidDriver;  import org.openqa.selenium.By;  import org.openqa.selenium.WebElement;  import org.openqa.selenium.remote.DesiredCapabilities;  import org.openqa.selenium.support.ui.ExpectedConditions;  import org.openqa.selenium.support.ui.WebDriverWait;  import java.net.MalformedURLException;  import java.net.URL;  import java.time.Duration;  public class Pra7 {  private static AndroidDriver driver;  public static void main(String[] args) throws MalformedURLException, InterruptedException {  // Set Desired Capabilities  DesiredCapabilities caps = new DesiredCapabilities();  caps.setCapability("deviceName", "Android Emulator");  caps.setCapability("platformName", "Android");  caps.setCapability("platformVersion", "14.0"); // Ensure this matches your emulator  caps.setCapability("appPackage", "com.example.myapplication"); // Replace with your APK's package name  caps.setCapability("appActivity", "com.example.myapplication.MainActivity"); // Replace with your APK's main activity  caps.setCapability("automationName", "UiAutomator2");  caps.setCapability("app", "C:\\MASTER OF COMPUTER APPLICATIONS (MCA)\\MCA SEM 3\\MOBILE TESTING\\new app practical 10\\app\\build\\outputs\\apk\\debug\\text.apk");  // Create a new instance of the Appium Driver  driver = new AndroidDriver(new URL("http://127.0.0.1:4723"), caps);  // Create an instance of WebDriverWait  WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(30));  // Wait for the specific element to be visible  WebElement element = wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("com.example.myapplication:id/text")));  // Get the text of the element  String actualText = element.getText();  String expectedText = "Hello World!"; // Replace with the text you expect  // Assert (Verify) if the element contains the expected text  if (actualText.equals(expectedText)) {  System.out.println("Text verification passed: " + actualText);  } else {  System.out.println("Text verification failed. Expected: " + expectedText + ", but got: " + actualText);  }  driver.quit();  }  } |
|  | from appium import webdriver  from selenium.webdriver.common.by import By  from selenium.webdriver.support.ui import WebDriverWait  from selenium.webdriver.support import expected\_conditions as EC  def main():  # Set Desired Capabilities  desired\_caps = {  "deviceName": "Android Emulator",  "platformName": "Android",  "platformVersion": "14.0",  "appPackage": "com.example.myapplication",  "appActivity": "com.example.myapplication.MainActivity",  "automationName": "UiAutomator2",  "app": "C:\\MASTER OF COMPUTER APPLICATIONS (MCA)\\MCA SEM 3\\MOBILE TESTING\\new app practical 10\\app\\build\\outputs\\apk\\debug\\text.apk"  }  driver = webdriver.Remote("http://127.0.0.1:4723 ", desired\_caps)  try:  wait = WebDriverWait(driver, 30)  element = wait.until(  EC.visibility\_of\_element\_located((By.ID, "com.example.myapplication:id/text"))  )    actual\_text = element.text  expected\_text = "Hello World!" # Replace with the expected text    if actual\_text == expected\_text:  print(f"Text verification passed: {actual\_text}")  else:  print(f"Text verification failed. Expected: {expected\_text}, but got: {actual\_text}")  except Exception as e:  print(f"An error occurred: {e}")  finally:  driver.quit()  if \_\_name\_\_ == "\_\_main\_\_":  main() |
| Input | NA |
| Output |  |

|  |  |
| --- | --- |
| **Practical No: 08** | **Write a script in Java to login the application with element of**  **username and password using Appium.** |
|  | package com.example.myapp;  import androidx.appcompat.app.AppCompatActivity;  import android.content.Intent; import android.os.Bundle; import android.view.View; import android.widget.Button; import android.widget.EditText; import android.widget.Toast;  public class MainActivity extends AppCompatActivity {  private EditText usernameField;  private EditText passwordField;  private Button loginButton;   @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_main*);   usernameField = findViewById(R.id.*username*);  passwordField = findViewById(R.id.*password*);  loginButton = findViewById(R.id.*loginButton*);   loginButton.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  String username = usernameField.getText().toString();  String password = passwordField.getText().toString();   if (validateLogin(username, password)) {  // Move to WelcomeActivity on successful login  Intent intent = new Intent(MainActivity.this, WelcomeActivity.class);  startActivity(intent);  } else {  Toast.*makeText*(MainActivity.this, "Login failed!", Toast.*LENGTH\_SHORT*).show();  }  }  });  }   private boolean validateLogin(String username, String password) {  return username.equals("admin") && password.equals("12345");  } } |
|  | <?xml version="1.0" encoding="utf-8"?> <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  xmlns:tools="http://schemas.android.com/tools"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  tools:context=".MainActivity">   <EditText  android:id="@+id/username"  android:layout\_width="0dp"  android:layout\_height="wrap\_content"  android:hint="Username"  app:layout\_constraintTop\_toTopOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintEnd\_toEndOf="parent"  android:layout\_margin="16dp"/>   <EditText  android:id="@+id/password"  android:layout\_width="0dp"  android:layout\_height="wrap\_content"  android:hint="Password"  android:inputType="textPassword"  app:layout\_constraintTop\_toBottomOf="@id/username"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintEnd\_toEndOf="parent"  android:layout\_margin="16dp"/>   <Button  android:id="@+id/loginButton"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="Login"  app:layout\_constraintTop\_toBottomOf="@id/password"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintEnd\_toEndOf="parent"  android:layout\_marginTop="32dp"/> </androidx.constraintlayout.widget.ConstraintLayout> |
|  | package com.example.myapp;  import androidx.appcompat.app.AppCompatActivity;  import android.os.Bundle;  public class WelcomeActivity extends AppCompatActivity {   @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_welcome*);  } } |
|  | <?xml version="1.0" encoding="utf-8"?> <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  xmlns:tools="http://schemas.android.com/tools"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  tools:context=".WelcomeActivity">   <TextView  android:id="@+id/welcomeMessage"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_centerInParent="true"  android:text="Welcome to the app!"  android:textSize="24sp"  app:layout\_constraintBottom\_toBottomOf="parent"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintHorizontal\_bias="0.5"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toTopOf="parent"  app:layout\_constraintVertical\_bias="0.5" />  </androidx.constraintlayout.widget.ConstraintLayout> |
|  | <project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">    <modelVersion>4.0.0</modelVersion>  <groupId>Practical8</groupId>  <artifactId>pra8</artifactId>  <version>0.0.1-SNAPSHOT</version>    <dependencies>  <!-- Appium Java Client -->  <dependency>  <groupId>io.appium</groupId>  <artifactId>java-client</artifactId>  <version>9.2.2</version>  </dependency>  <!-- Selenium Java -->  <dependency>  <groupId>org.seleniumhq.selenium</groupId>  <artifactId>selenium-java</artifactId>  <version>4.25.0</version>  </dependency>    <dependency>  <groupId>org.testng</groupId>  <artifactId>testng</artifactId>  <version>7.7.0</version>  </dependency>  </dependencies>  <build>  <plugins>  <!-- Compiler plugin for setting the Java version -->  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-compiler-plugin</artifactId>  <version>3.8.1</version>  <configuration>  <source>17</source>  <target>17</target>  </configuration>  </plugin>  </plugins>  </build>  </project> |
|  | package pra8;  import io.appium.java\_client.android.AndroidDriver;  import org.openqa.selenium.By;  import org.openqa.selenium.WebElement;  import org.openqa.selenium.remote.DesiredCapabilities;  import org.openqa.selenium.support.ui.ExpectedConditions;  import org.openqa.selenium.support.ui.WebDriverWait;  import java.net.MalformedURLException;  import java.net.URL;  import java.time.Duration;  public class Pra8 {  public static void main(String[] args) throws MalformedURLException, InterruptedException {  // Set Desired Capabilities  DesiredCapabilities caps = new DesiredCapabilities();  caps.setCapability("deviceName", "Android Emulator");  caps.setCapability("platformName", "Android");  caps.setCapability("platformVersion", "14.0");  caps.setCapability("appPackage", "com.example.myapp");  caps.setCapability("appActivity", "com.example.myapp.MainActivity");  caps.setCapability("automationName", "UiAutomator2");  caps.setCapability("app", "C:\\MASTER OF COMPUTER APPLICATIONS (MCA)\\MCA SEM 3\\MOBILE TESTING\\dummy\\app\\build\\outputs\\apk\\debug\\app-debug.apk");  // Create a new instance of the Appium Driver  AndroidDriver driver = new AndroidDriver(new URL("http://127.0.0.1:4723"), caps);  // Wait for the app to load  WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(20));  // Find elements and interact with them  WebElement usernameField = wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("com.example.myapp:id/username")));  WebElement passwordField = wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("com.example.myapp:id/password")));  WebElement loginButton = wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("com.example.myapp:id/loginButton")));  // Input credentials  usernameField.sendKeys("admin");  passwordField.sendKeys("12345");  // Click the login button  loginButton.click();  // Wait for the WelcomeActivity to load and check for the welcome message  WebElement welcomeMessageElement = wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("com.example.myapp:id/welcomeMessage")));  if (welcomeMessageElement.isDisplayed()) {  System.out.println("Welcome message displayed");  } else {  System.out.println("Welcome message not displayed!");  }  // Quit the driver  driver.quit();  }  } |
|  | from appium import webdriver  from selenium.webdriver.common.by import By  from selenium.webdriver.support.ui import WebDriverWait  from selenium.webdriver.support import expected\_conditions as EC  def main():  # Set Desired Capabilities  desired\_caps = {  "deviceName": "Android Emulator",  "platformName": "Android",  "platformVersion": "14.0",  "appPackage": "com.example.myapp",  "appActivity": "com.example.myapp.MainActivity",  "automationName": "UiAutomator2",  "app": "C:\\MASTER OF COMPUTER APPLICATIONS (MCA)\\MCA SEM 3\\MOBILE TESTING\\dummy\\app\\build\\outputs\\apk\\debug\\app-debug.apk"  }  driver = webdriver.Remote("http://127.0.0.1:4723 ", desired\_caps)  try:  wait = WebDriverWait(driver, 20)  username\_field = wait.until(  EC.visibility\_of\_element\_located((By.ID, "com.example.myapp:id/username"))  )  password\_field = wait.until(  EC.visibility\_of\_element\_located((By.ID, "com.example.myapp:id/password"))  )  login\_button = wait.until(  EC.visibility\_of\_element\_located((By.ID, "com.example.myapp:id/loginButton"))  )  username\_field.send\_keys("admin")  password\_field.send\_keys("12345")  login\_button.click()    welcome\_message = wait.until(  EC.visibility\_of\_element\_located((By.ID, "com.example.myapp:id/welcomeMessage"))  )  if welcome\_message.is\_displayed():  print("Welcome message displayed")  else:  print("Welcome message not displayed!")  except Exception as e:  print(f"An error occurred: {e}")  finally:  driver.quit()  if \_\_name\_\_ == "\_\_main\_\_":  main() |
| Input |  |
| Output |  |

|  |  |
| --- | --- |
| **Practical No: 09** | **Write a script in Java to taking screenshot of each stages of**  **Application using Appium.** |
|  | <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>Practical9</groupId>  <artifactId>pra9</artifactId>  <version>0.0.1-SNAPSHOT</version>    <dependencies>  <!-- Appium Java Client -->  <dependency>  <groupId>io.appium</groupId>  <artifactId>java-client</artifactId>  <version>9.2.2</version>  </dependency>  <!-- Selenium Java -->  <dependency>  <groupId>org.seleniumhq.selenium</groupId>  <artifactId>selenium-java</artifactId>  <version>4.25.0</version>  </dependency>      </dependencies>  </project> |
|  | package pra9;  import io.appium.java\_client.android.AndroidDriver;  import org.openqa.selenium.By;  import org.openqa.selenium.OutputType;  import org.openqa.selenium.WebElement;  import org.openqa.selenium.remote.DesiredCapabilities;  import org.openqa.selenium.support.ui.ExpectedConditions;  import org.openqa.selenium.support.ui.WebDriverWait;  import java.io.File;  import java.io.IOException;  import java.net.MalformedURLException;  import java.net.URL;  import java.nio.file.Files;  import java.time.Duration;  public class Pra9 {  private static AndroidDriver driver;  private static final String SCREENSHOT\_DIR = "C:\\MASTER OF COMPUTER APPLICATIONS (MCA)\\MCA SEM 3\\MOBILE TESTING\\Screenshots\\";    public static void main(String[] args) throws MalformedURLException, InterruptedException, IOException {  // Set Desired Capabilities  DesiredCapabilities caps = new DesiredCapabilities();  caps.setCapability("deviceName", "Android Emulator");  caps.setCapability("platformName", "Android");  caps.setCapability("platformVersion", "14.0");  caps.setCapability("appPackage", "com.example.myapp");  caps.setCapability("appActivity", "com.example.myapp.MainActivity");  caps.setCapability("automationName", "UiAutomator2");  caps.setCapability("app", "C:\\MASTER OF COMPUTER APPLICATIONS (MCA)\\MCA SEM 3\\MOBILE TESTING\\dummy\\app\\build\\outputs\\apk\\debug\\app-debug.apk");  // Create a new instance of the Appium Driver  driver = new AndroidDriver(new URL("http://127.0.0.1:4723"), caps);  // Create an instance of WebDriverWait  WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(20));  // Screenshot after launching the app  takeScreenshot("app\_launched");  // Find elements and interact with them  WebElement usernameField = wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("com.example.myapp:id/username")));  takeScreenshot("username\_field\_visible");  WebElement passwordField = wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("com.example.myapp:id/password")));  takeScreenshot("password\_field\_visible");  WebElement loginButton = wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("com.example.myapp:id/loginButton")));  takeScreenshot("login\_button\_visible");  // Input credentials  usernameField.sendKeys("admin");  passwordField.sendKeys("12345");  takeScreenshot("credentials\_entered");  // Click the login button  loginButton.click();  takeScreenshot("login\_button\_clicked");  // Wait for the WelcomeActivity to load and check for the welcome message  WebElement welcomeMessageElement = wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("com.example.myapp:id/welcomeMessage")));  takeScreenshot("welcome\_message\_visible");  // Quit the driver  driver.quit();  }  // Method to take a screenshot  private static void takeScreenshot(String step) throws IOException {  File screenshot = driver.getScreenshotAs(OutputType.FILE);  File destination = new File(SCREENSHOT\_DIR + step + ".png");  Files.copy(screenshot.toPath(), destination.toPath());  System.out.println("Screenshot taken: " + destination.getAbsolutePath());  }  } |
|  | from appium import webdriver  from selenium.webdriver.common.by import By  from selenium.webdriver.support.ui import WebDriverWait  from selenium.webdriver.support import expected\_conditions as EC  import time  import os  import shutil  SCREENSHOT\_DIR = "C:\\MASTER OF COMPUTER APPLICATIONS (MCA)\\MCA SEM 3\\MOBILE TESTING\\Screenshots\\"  def take\_screenshot(driver, step):  """Take a screenshot and save it with the given step name."""  screenshot\_path = os.path.join(SCREENSHOT\_DIR, f"{step}.png")  screenshot = driver.get\_screenshot\_as\_file(screenshot\_path)  if screenshot:  print(f"Screenshot taken: {screenshot\_path}")  else:  print(f"Failed to take screenshot: {step}")  def main():  # Set Desired Capabilities  desired\_caps = {  "deviceName": "Android Emulator",  "platformName": "Android",  "platformVersion": "14.0",  "appPackage": "com.example.myapp",  "appActivity": "com.example.myapp.MainActivity",  "automationName": "UiAutomator2",  "app": "C:\\MASTER OF COMPUTER APPLICATIONS (MCA)\\MCA SEM 3\\MOBILE TESTING\\dummy\\app\\build\\outputs\\apk\\debug\\app-debug.apk"  }  # Initialize the Appium driver  driver = webdriver.Remote("http://127.0.0.1:4723 ", desired\_caps)  try:  wait = WebDriverWait(driver, 20)  take\_screenshot(driver, "app\_launched")  username\_field = wait.until(  EC.visibility\_of\_element\_located((By.ID, "com.example.myapp:id/username"))  )  take\_screenshot(driver, "username\_field\_visible")  password\_field = wait.until(  EC.visibility\_of\_element\_located((By.ID, "com.example.myapp:id/password"))  )  take\_screenshot(driver, "password\_field\_visible")  login\_button = wait.until(  EC.visibility\_of\_element\_located((By.ID, "com.example.myapp:id/loginButton"))  )  take\_screenshot(driver, "login\_button\_visible")  username\_field.send\_keys("admin")  password\_field.send\_keys("12345")  take\_screenshot(driver, "credentials\_entered")  login\_button.click()  take\_screenshot(driver, "login\_button\_clicked")  welcome\_message = wait.until(  EC.visibility\_of\_element\_located((By.ID, "com.example.myapp:id/welcomeMessage"))  )  take\_screenshot(driver, "welcome\_message\_visible")  except Exception as e:  print(f"An error occurred: {e}")  finally:  driver.quit()  if \_\_name\_\_ == "\_\_main\_\_":    os.makedirs(SCREENSHOT\_DIR, exist\_ok=True)  main() |
| Input | NA |
| Output |  |

|  |  |  |  |
| --- | --- | --- | --- |
| app launched | username visible | password visible |  |
| loginbutton visible | credentials entered | login button clicked | welcome message |

**Screenshots**

|  |  |
| --- | --- |
| **Practical No: 10** | **Automate the composed email process for the Gmail app on an**  **android device using Appium and Java.** |
|  | <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>Practical10</groupId>  <artifactId>pra10</artifactId>  <version>0.0.1-SNAPSHOT</version>    <dependencies>  <!-- Appium Java Client -->  <dependency>  <groupId>io.appium</groupId>  <artifactId>java-client</artifactId>  <version>9.2.2</version>  </dependency>  <!-- Selenium Java -->  <dependency>  <groupId>org.seleniumhq.selenium</groupId>  <artifactId>selenium-java</artifactId>  <version>4.25.0</version>  </dependency>  </dependencies>  </project> |
|  | package pra10;  import io.appium.java\_client.android.AndroidDriver;  import org.openqa.selenium.By;  import org.openqa.selenium.OutputType;  import org.openqa.selenium.WebElement;  import org.openqa.selenium.remote.DesiredCapabilities;  import org.openqa.selenium.support.ui.ExpectedConditions;  import org.openqa.selenium.support.ui.WebDriverWait;  import java.io.File;  import java.io.IOException;  import java.net.MalformedURLException;  import java.net.URL;  import java.nio.file.Files;  import java.time.Duration;  public class Pra10 {  private static AndroidDriver driver;  private static final String SCREENSHOT\_DIR = "C:\\MASTER OF COMPUTER APPLICATIONS (MCA)\\MCA SEM 3\\MOBILE TESTING\\Pra 10 screenshots\\";  public static void main(String[] args) throws MalformedURLException, InterruptedException, IOException {  // Set Desired Capabilities  DesiredCapabilities caps = new DesiredCapabilities();  caps.setCapability("deviceName", "Android Emulator");  caps.setCapability("platformName", "Android");  caps.setCapability("platformVersion", "14.0");  caps.setCapability("appPackage", "com.google.android.gm");  caps.setCapability("appActivity", "com.google.android.gm.ConversationListActivityGmail");  caps.setCapability("automationName", "UiAutomator2");  // Create a new instance of the Appium Driver  driver = new AndroidDriver(new URL("http://127.0.0.1:4723"), caps);  // Create an instance of WebDriverWait  WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(120), Duration.ofMillis(1000)); // Increase wait duration  // Handle the "Got it" button (if it's displayed)  WebElement gotItButton = wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("com.google.android.gm:id/welcome\_tour\_got\_it")));  gotItButton.click();  takeScreenshot("got\_it\_button\_clicked");  // Handle the "Take me to Gmail" button (if it's displayed)  WebElement takeMeToGmailButton = wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("com.google.android.gm:id/action\_done")));  takeMeToGmailButton.click();  takeScreenshot("take\_me\_to\_gmail\_button\_clicked");      // Check for permission dialog and click "Allow"  WebElement allowButton = wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("com.android.permissioncontroller:id/permission\_allow\_button")));  allowButton.click();      // Now wait for the "Got it" button to appear  WebElement gotItButton1 = wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//android.widget.Button[@text='Got it']")));  gotItButton1.click();    // Wait for the Gmail main screen to load  wait.until(ExpectedConditions.presenceOfElementLocated(By.id("com.google.android.gm:id/compose\_button")));  // Click the "Compose" button  WebElement composeButton = wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("com.google.android.gm:id/compose\_button")));  composeButton.click();  takeScreenshot("compose\_button\_clicked");  // Wait for the "To" field  WebElement toField = wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//android.widget.EditText[@text='To']")));  toField.click();  toField.sendKeys("23034211064@gnu.ac.in");  // Enter the subject  WebElement subjectField = wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//android.widget.EditText[contains(@resource-id, 'subject')]")));  subjectField.sendKeys("This is Practical 10");    // Enter email body  WebElement bodyField = wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//android.widget.EditText[contains(@resource-id, 'body')]")));  bodyField.sendKeys("I am Appium");    // Click the "Send" button  WebElement sendButton = wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("com.google.android.gm:id/send")));  sendButton.click();    driver.quit();  }  // Method to take a screenshot  private static void takeScreenshot(String step) throws IOException {  File screenshot = driver.getScreenshotAs(OutputType.FILE);  File destination = new File(SCREENSHOT\_DIR + step + ".png");  Files.copy(screenshot.toPath(), destination.toPath());  }  } |
|  | from appium import webdriver  from selenium.webdriver.common.by import By  from selenium.webdriver.support.ui import WebDriverWait  from selenium.webdriver.support import expected\_conditions as EC  import os  import shutil  SCREENSHOT\_DIR = "C:\\MASTER OF COMPUTER APPLICATIONS (MCA)\\MCA SEM 3\\MOBILE TESTING\\Pra 10 screenshots\\"  def take\_screenshot(driver, step):  """Take a screenshot and save it with the given step name."""  screenshot\_path = os.path.join(SCREENSHOT\_DIR, f"{step}.png")  if driver.get\_screenshot\_as\_file(screenshot\_path):  print(f"Screenshot taken: {screenshot\_path}")  else:  print(f"Failed to take screenshot: {step}")  def main():    desired\_caps = {  "deviceName": "Android Emulator",  "platformName": "Android",  "platformVersion": "14.0", # Ensure it matches your emulator  "appPackage": "com.google.android.gm",  "appActivity": "com.google.android.gm.ConversationListActivityGmail",  "automationName": "UiAutomator2"  }  driver = webdriver.Remote("http://127.0.0.1:4723 ", desired\_caps)  try:  wait = WebDriverWait(driver, 120, poll\_frequency=1)    got\_it\_button = wait.until(  EC.visibility\_of\_element\_located((By.ID, "com.google.android.gm:id/welcome\_tour\_got\_it"))  )  got\_it\_button.click()  take\_screenshot(driver, "got\_it\_button\_clicked")  take\_me\_to\_gmail\_button = wait.until(  EC.visibility\_of\_element\_located((By.ID, "com.google.android.gm:id/action\_done"))  )  take\_me\_to\_gmail\_button.click()  take\_screenshot(driver, "take\_me\_to\_gmail\_button\_clicked")  allow\_button = wait.until(  EC.visibility\_of\_element\_located((By.ID, "com.android.permissioncontroller:id/permission\_allow\_button"))  )  allow\_button.click()    got\_it\_button\_1 = wait.until(  EC.visibility\_of\_element\_located((By.XPATH, "//android.widget.Button[@text='Got it']"))  )  got\_it\_button\_1.click()  wait.until(EC.presence\_of\_element\_located((By.ID, "com.google.android.gm:id/compose\_button")))    compose\_button = wait.until(  EC.visibility\_of\_element\_located((By.ID, "com.google.android.gm:id/compose\_button"))  )  compose\_button.click()  take\_screenshot(driver, "compose\_button\_clicked")    to\_field = wait.until(  EC.visibility\_of\_element\_located((By.XPATH, "//android.widget.EditText[@text='To']"))  )  to\_field.click()  to\_field.send\_keys("23034211064@gnu.ac.in")  subject\_field = wait.until(  EC.visibility\_of\_element\_located((By.XPATH, "//android.widget.EditText[contains(@resource-id, 'subject')]"))  )  subject\_field.send\_keys("This is Practical 10")  body\_field = wait.until(  EC.visibility\_of\_element\_located((By.XPATH, "//android.widget.EditText[contains(@resource-id, 'body')]"))  )  body\_field.send\_keys("I am Appium")  send\_button = wait.until(  EC.visibility\_of\_element\_located((By.ID, "com.google.android.gm:id/send"))  )  send\_button.click()  take\_screenshot(driver, "email\_sent")  except Exception as e:  print(f"An error occurred: {e}")  finally:  driver.quit()  if \_\_name\_\_ == "\_\_main\_\_":    os.makedirs(SCREENSHOT\_DIR, exist\_ok=True)  main() |
| Input | NA |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Output | |  |  | | --- | --- | | Take me to Gmail | Got it Button clicked  . | | Compose button clicked | Enter details | |

|  |  |
| --- | --- |
| **Practical No: 11** | **Automate the login process for the Facebook App on an Android**  **Device using Appium and Java.** |
|  | <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>Practical11</groupId>  <artifactId>pra11</artifactId>  <version>0.0.1-SNAPSHOT</version>    <dependencies>  <!-- Appium Java Client -->  <dependency>  <groupId>io.appium</groupId>  <artifactId>java-client</artifactId>  <version>9.2.2</version>  </dependency>  <!-- Selenium Java -->  <dependency>  <groupId>org.seleniumhq.selenium</groupId>  <artifactId>selenium-java</artifactId>  <version>4.25.0</version>  </dependency>  </dependencies>  </project> |
|  | package pra11;  import io.appium.java\_client.android.AndroidDriver;  import org.openqa.selenium.By;  import org.openqa.selenium.WebElement;  import org.openqa.selenium.remote.DesiredCapabilities;  import org.openqa.selenium.support.ui.ExpectedConditions;  import org.openqa.selenium.support.ui.WebDriverWait;  import java.net.MalformedURLException;  import java.net.URL;  import java.time.Duration;  public class Pra11 {  private static AndroidDriver driver;  public static void main(String[] args) throws MalformedURLException, InterruptedException {  // Set Desired Capabilities  DesiredCapabilities caps = new DesiredCapabilities();  caps.setCapability("deviceName", "Android Emulator");  caps.setCapability("platformName", "Android");  caps.setCapability("platformVersion", "14.0");  caps.setCapability("appPackage", "com.facebook.katana");  caps.setCapability("appActivity", "com.facebook.katana.LoginActivity");  caps.setCapability("automationName", "UiAutomator2");    driver = new AndroidDriver(new URL("http://127.0.0.1:4723"), caps);    WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(30));    WebElement emailField = wait.until(  ExpectedConditions.visibilityOfElementLocated(By.id("com.facebook.katana:id/login\_username"))  );  emailField.sendKeys("7487944191");    WebElement passwordField = wait.until(  ExpectedConditions.visibilityOfElementLocated(By.id("com.facebook.katana:id/login\_password"))  );  passwordField.sendKeys("1234");    WebElement loginButton = wait.until(  ExpectedConditions.elementToBeClickable(By.id("com.facebook.katana:id/login\_login"))  );  loginButton.click();  System.out.println("Login Successfully");  Thread.sleep(5000);    driver.quit();  }  } |
|  | from appium import webdriver  from selenium.webdriver.common.by import By  from selenium.webdriver.support.ui import WebDriverWait  from selenium.webdriver.support import expected\_conditions as EC  import time  def main():  # Set Desired Capabilities  desired\_caps = {  "deviceName": "Android Emulator",  "platformName": "Android",  "platformVersion": "14.0",  "appPackage": "com.facebook.katana",  "appActivity": "com.facebook.katana.LoginActivity",  "automationName": "UiAutomator2"  }  # Initialize the Appium Driver  driver = webdriver.Remote("http://127.0.0.1:4723 ", desired\_caps)  try:    wait = WebDriverWait(driver, 30)    email\_field = wait.until(  EC.visibility\_of\_element\_located((By.ID, "com.facebook.katana:id/login\_username"))  )  email\_field.send\_keys("your\_email\_or\_phone\_number")  password\_field = wait.until(  EC.visibility\_of\_element\_located((By.ID, "com.facebook.katana:id/login\_password"))  )  password\_field.send\_keys("your\_password")  login\_button = wait.until(  EC.element\_to\_be\_clickable((By.ID, "com.facebook.katana:id/login\_login"))  )  login\_button.click()    time.sleep(5) # Adjust based on how fast the login completes  except Exception as e:  print(f"An error occurred: {e}")  finally:  driver.quit()  if \_\_name\_\_ == "\_\_main\_\_":  main() |
| **Input** | NA |
| **Output** | |  |  |  | | --- | --- | --- | |  |  |  | |

|  |  |
| --- | --- |
| **Practical No: 12** | **Automate the registration process in Android App using Appium**  **and Java.** |
|  | <?xml version="1.0" encoding="utf-8"?>  <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  xmlns:tools="http://schemas.android.com/tools"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  tools:context=".MainActivity">  <EditText  android:id="@+id/username"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:hint="Enter Username"  android:inputType="textPersonName"  app:layout\_constraintBottom\_toTopOf="@+id/email"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintHorizontal\_bias="0.5"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toTopOf="parent"  app:layout\_constraintVertical\_bias="0.5" />  <EditText  android:id="@+id/email"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:hint="Enter Email"  android:inputType="textEmailAddress"  app:layout\_constraintBottom\_toTopOf="@+id/password"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintHorizontal\_bias="0.5"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toBottomOf="@+id/username"  app:layout\_constraintVertical\_bias="0.5" />  <EditText  android:id="@+id/password"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:hint="Enter Password"  android:inputType="textPassword"  app:layout\_constraintBottom\_toTopOf="@+id/registerButton"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintHorizontal\_bias="0.5"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toBottomOf="@+id/email"  app:layout\_constraintVertical\_bias="0.5" />  <Button  android:id="@+id/registerButton"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="Register"  app:layout\_constraintBottom\_toBottomOf="parent"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintHorizontal\_bias="0.5"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toBottomOf="@+id/password" />  </androidx.constraintlayout.widget.ConstraintLayout> |
|  | package com.example.myapplication;  import androidx.appcompat.app.AppCompatActivity;  import android.os.Bundle; import android.widget.Button; import android.widget.EditText; import android.widget.Toast;  public class MainActivity extends AppCompatActivity {   @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_main*);   // Initialize UI elements  EditText username = findViewById(R.id.*username*);  EditText email = findViewById(R.id.*email*);  EditText password = findViewById(R.id.*password*);  Button registerButton = findViewById(R.id.*registerButton*);   // Handle button click  registerButton.setOnClickListener(v -> {  String user = username.getText().toString();  String mail = email.getText().toString();  String pass = password.getText().toString();   if (user.isEmpty() || mail.isEmpty() || pass.isEmpty()) {  Toast.*makeText*(this, "All fields are required", Toast.*LENGTH\_SHORT*).show();  } else {  Toast.*makeText*(this, "Registration Successful", Toast.*LENGTH\_SHORT*).show();  }  });  } } |
|  | <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>Practical12</groupId>  <artifactId>pra12</artifactId>  <version>0.0.1-SNAPSHOT</version>    <dependencies>  <!-- Appium Java Client -->  <dependency>  <groupId>io.appium</groupId>  <artifactId>java-client</artifactId>  <version>9.2.2</version>  </dependency>  <!-- Selenium Java -->  <dependency>  <groupId>org.seleniumhq.selenium</groupId>  <artifactId>selenium-java</artifactId>  <version>4.25.0</version>  </dependency>  </dependencies>  </project> |
|  | package pra12;  import io.appium.java\_client.android.AndroidDriver;  import org.openqa.selenium.By;  import org.openqa.selenium.WebElement;  import org.openqa.selenium.remote.DesiredCapabilities;  import java.net.MalformedURLException;  import java.net.URL;  import java.time.Duration;  @SuppressWarnings("deprecation")  public class pra12 {  private static AndroidDriver driver;  public static void main(String[] args) {  try {    DesiredCapabilities caps = new DesiredCapabilities();  caps.setCapability("deviceName", "Android Emulator");  caps.setCapability("platformName", "Android");  caps.setCapability("platformVersion", "14.0");  caps.setCapability("appPackage", "com.example.myapplication");  caps.setCapability("appActivity", "com.example.myapplication.MainActivity");  caps.setCapability("automationName", "UiAutomator2");  caps.setCapability("app", "C:\\MASTER OF COMPUTER APPLICATIONS (MCA)\\MCA SEM 3\\MOBILE TESTING\\new app practical 10\\app\\build\\outputs\\apk\\debug\\register.apk");    driver = new AndroidDriver(new URL("http://127.0.0.1:4723"), caps);    driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));    WebElement usernameField = driver.findElement(By.id("com.example.myapplication:id/username")); // Adjust with your actual ID  usernameField.sendKeys("JackSparrow");  System.out.println("username entered");    WebElement emailField = driver.findElement(By.id("com.example.myapplication:id/email")); // Adjust with your actual ID  emailField.sendKeys("jacksparrow@example.com");  System.out.println("email entered");    WebElement passwordField = driver.findElement(By.id("com.example.myapplication:id/password")); // Adjust with your actual ID  passwordField.sendKeys("password123");  System.out.println("password entered");    WebElement registerButton = driver.findElement(By.id("com.example.myapplication:id/registerButton")); // Adjust with your actual ID  registerButton.click();  System.out.println("Registration form filled successfully.");  } catch (MalformedURLException e) {  e.printStackTrace();  } catch (Exception e) {  e.printStackTrace();  } finally {    if (driver != null) {  driver.quit();  }  }  }  } |
|  | from appium import webdriver  from selenium.webdriver.common.by import By  import time  def main():  try:  # Set Desired Capabilities  desired\_caps = {  "deviceName": "Android Emulator",  "platformName": "Android",  "platformVersion": "14.0",  "appPackage": "com.example.myapplication",  "appActivity": "com.example.myapplication.MainActivity",  "automationName": "UiAutomator2",  "app": "C:\\MASTER OF COMPUTER APPLICATIONS (MCA)\\MCA SEM 3\\MOBILE TESTING\\new app practical 10\\app\\build\\outputs\\apk\\debug\\register.apk"  }  # Initialize the Appium Driver  driver = webdriver.Remote("http://127.0.0.1:4723 ", desired\_caps)  # Set implicit wait to avoid synchronization issues  driver.implicitly\_wait(10)  username\_field = driver.find\_element(By.ID, "com.example.myapplication:id/username")  username\_field.send\_keys("JackSparrow")  print("Username entered")    email\_field = driver.find\_element(By.ID, "com.example.myapplication:id/email")  email\_field.send\_keys("jacksparrow@example.com")  print("Email entered")  password\_field = driver.find\_element(By.ID, "com.example.myapplication:id/password")  password\_field.send\_keys("password123")  print("Password entered")    register\_button = driver.find\_element(By.ID, "com.example.myapplication:id/registerButton")  register\_button.click()  print("Registration form filled successfully.")  except Exception as e:  print(f"An error occurred: {e}")  finally:  if driver:  driver.quit()  if \_\_name\_\_ == "\_\_main\_\_":  main() |
| **Input** | NA |
| **Output** |  |

|  |  |
| --- | --- |
| **Practical No: 13** | **Automate the scrolling and swiping process for the Android App**  **using Appium and Java.** |
|  | <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>Practical13</groupId>  <artifactId>pra13</artifactId>  <version>0.0.1-SNAPSHOT</version>    <dependencies>  <!-- Appium Java Client -->  <dependency>  <groupId>io.appium</groupId>  <artifactId>java-client</artifactId>  <version>8.0.0</version>  </dependency>  <!-- Selenium Java -->  <dependency>  <groupId>org.seleniumhq.selenium</groupId>  <artifactId>selenium-java</artifactId>  <version>4.25.0</version>  </dependency>  </dependencies>  </project> |
|  | <?xml version="1.0" encoding="utf-8"?> <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  xmlns:tools="http://schemas.android.com/tools"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  tools:context=".MainActivity">   <Button  android:id="@+id/testButton"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="Click Me"  android:layout\_alignParentTop="true"  android:layout\_centerHorizontal="true" />   <ListView  android:id="@+id/listView"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_below="@id/testButton"  android:layout\_marginTop="16dp" />  </RelativeLayout> |
|  | package com.example.practical13;  import androidx.appcompat.app.AppCompatActivity;  import android.os.Bundle;  import android.view.View;  import android.widget.ArrayAdapter;  import android.widget.Button;  import android.widget.ListView;  import android.widget.Toast;  import java.util.ArrayList;  import java.util.List;  public class MainActivity extends AppCompatActivity {  private Button testButton;  private ListView listView;  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main);  testButton = findViewById(R.id.testButton);  listView = findViewById(R.id.listView);  // Set up button click listener  testButton.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  Toast.makeText(MainActivity.this, "Button Clicked!", Toast.LENGTH\_SHORT).show();  }  });  // Set up list view with dummy data  List<String> dataList = new ArrayList<>();  for (int i = 1; i <= 50; i++) {  dataList.add("Item " + i);  }  ArrayAdapter<String> adapter = new ArrayAdapter<>(  this, android.R.layout.simple\_list\_item\_1, dataList);  listView.setAdapter(adapter);  }  } |
|  | package pra13;  import io.appium.java\_client.AppiumDriver;  import io.appium.java\_client.android.AndroidDriver;  import io.appium.java\_client.touch.WaitOptions;  import io.appium.java\_client.touch.offset.PointOption;  import io.appium.java\_client.touch.TouchAction;  import org.openqa.selenium.By;  import org.openqa.selenium.WebElement;  import org.openqa.selenium.remote.DesiredCapabilities;  import java.net.MalformedURLException;  import java.net.URL;  import java.time.Duration;  public class pra13 {    private static AndroidDriver driver;  public static void main(String[] args) {  try {  // Set up Desired Capabilities  DesiredCapabilities capabilities = new DesiredCapabilities();  capabilities.setCapability("platformName", "Android");  capabilities.setCapability("deviceName", "Android Emulator");  capabilities.setCapability("appPackage", "com.example.practical13");  capabilities.setCapability("appActivity", ".MainActivity");  capabilities.setCapability("automationName", "UiAutomator2");  capabilities.setCapability("app", "C:\\MASTER OF COMPUTER APPLICATIONS (MCA)\\MCA SEM 3\\MOBILE TESTING\\Practical13\\app\\build\\outputs\\apk\\debug\\pra.apk");    driver = new AndroidDriver(new URL("http://127.0.0.1:4723"), capabilities);  // Wait for the app to load  Thread.sleep(5000);  // Click the button  WebElement button = driver.findElement(By.id("com.example.practical13:id/testButton"));  button.click();  System.out.println("Button clicked successfully!");  // Scroll down the list view  scrollDown();  } catch (MalformedURLException | InterruptedException e) {  e.printStackTrace();  } finally {  if (driver != null) {  driver.quit();  }  }  }  // Use TouchAction to scroll down and log the action  public static void scrollDown() {  try {  System.out.println("Starting scroll down...");  // Use the correct type for TouchAction  TouchAction<AndroidDriver> touchAction = new TouchAction<>(driver);  // Get screen size for dynamic swipe coordinates  int screenWidth = driver.manage().window().getSize().width;  int screenHeight = driver.manage().window().getSize().height;  // Perform the swipe from bottom to top  touchAction  .press(PointOption.point(screenWidth / 2, (int) (screenHeight \* 0.8))) // Start point (bottom)  .waitAction(WaitOptions.waitOptions(Duration.ofMillis(500))) // Wait for swipe  .moveTo(PointOption.point(screenWidth / 2, (int) (screenHeight \* 0.2))) // End point (top)  .release()  .perform();  System.out.println("Scroll down completed successfully!");  } catch (Exception e) {  System.out.println("Scroll action failed: " + e.getMessage());  }  }  } |
|  | from appium import webdriver  from appium.webdriver.common.touch\_action import TouchAction  from selenium.webdriver.common.by import By  import time  def main():  try:  # Set up Desired Capabilities  desired\_caps = {  "platformName": "Android",  "deviceName": "Android Emulator",  "appPackage": "com.example.practical13",  "appActivity": ".MainActivity",  "automationName": "UiAutomator2",  "app": "C:\\MASTER OF COMPUTER APPLICATIONS (MCA)\\MCA SEM 3\\MOBILE TESTING\\Practical13\\app\\build\\outputs\\apk\\debug\\pra.apk"  }  # Initialize the Appium driver  driver = webdriver.Remote("http://127.0.0.1:4723 ", desired\_caps)  # Wait for the app to load  time.sleep(5)  # Find and click the button  button = driver.find\_element(By.ID, "com.example.practical13:id/testButton")  button.click()  print("Button clicked successfully!")  # Scroll down the list view  scroll\_down(driver)  except Exception as e:  print(f"An error occurred: {e}")  finally:  # Quit the driver to close the app  if driver:  driver.quit()  def scroll\_down(driver):  try:  print("Starting scroll down...")  # Get the screen size for dynamic swipe coordinates  screen\_size = driver.get\_window\_size()  screen\_width = screen\_size['width']  screen\_height = screen\_size['height']  action = TouchAction(driver)  action.press(x=screen\_width // 2, y=int(screen\_height \* 0.8)) \  .wait(ms=500) \  .move\_to(x=screen\_width // 2, y=int(screen\_height \* 0.2)) \  .release() \  .perform()  print("Scroll down completed successfully!")  except Exception as e:  print(f"Scroll action failed: {e}")  if \_\_name\_\_ == "\_\_main\_\_":  main() |
| **Input** | NA |
| **Output** | |  |  | | --- | --- | |  |  | |

|  |  |
| --- | --- |
| **Practical No: 14** | **Write a code to desired if the device locked screen is on or off.** |
|  | <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>Practical14</groupId>  <artifactId>pra14</artifactId>  <version>0.0.1-SNAPSHOT</version>    <dependencies>  <!-- Appium Java Client -->  <dependency>  <groupId>io.appium</groupId>  <artifactId>java-client</artifactId>  <version>9.2.2</version>  </dependency>  <!-- Selenium Java -->  <dependency>  <groupId>org.seleniumhq.selenium</groupId>  <artifactId>selenium-java</artifactId>  <version>4.25.0</version>  </dependency>  </dependencies>  </project> |
|  | package pra14;  import io.appium.java\_client.android.AndroidDriver;  import org.openqa.selenium.remote.DesiredCapabilities;  import java.net.MalformedURLException;  import java.net.URL;  public class Pra14 {  public static <MobileElement> void main(String[] args) {  DesiredCapabilities capabilities = new DesiredCapabilities();  DesiredCapabilities capabilities1 = new DesiredCapabilities();  capabilities1.setCapability("platformName", "Android");  capabilities1.setCapability("deviceName", "Android Emulator"); // Replace with your device name  capabilities1.setCapability("appPackage", "com.example.practical13");  capabilities1.setCapability("appActivity", ".MainActivity");  capabilities1.setCapability("automationName", "UiAutomator2");  capabilities1.setCapability("app", "C:\\MASTER OF COMPUTER APPLICATIONS (MCA)\\MCA SEM 3\\MOBILE TESTING\\Practical13\\app\\build\\outputs\\apk\\debug\\pra.apk");  AndroidDriver driver = null;  try {  driver = new AndroidDriver(new URL("http://127.0.0.1:4723"), capabilities1);  // Check if the device is locked  boolean isLocked = driver.isDeviceLocked();  if (isLocked) {  System.out.println("The device screen is locked.");  } else {  System.out.println("The device screen is unlocked.");  }  } catch (MalformedURLException e) {  e.printStackTrace();  } finally {  if (driver != null) {  driver.quit(); // Close the app  }  }  }  } |
|  | from appium import webdriver  from selenium.webdriver.common.by import By  import time  def main():    desired\_caps = {  "platformName": "Android",  "deviceName": "Android Emulator", # Replace with your device name  "appPackage": "com.example.practical13",  "appActivity": ".MainActivity",  "automationName": "UiAutomator2",  "app": "C:\\MASTER OF COMPUTER APPLICATIONS (MCA)\\MCA SEM 3\\MOBILE TESTING\\Practical13\\app\\build\\outputs\\apk\\debug\\pra.apk"  }  # Initialize the Appium driver  driver = None  try:  driver = webdriver.Remote("http://127.0.0.1:4723 ", desired\_caps)    is\_locked = driver.is\_locked()  if is\_locked:  print("The device screen is locked.")  else:  print("The device screen is unlocked.")  except Exception as e:  print(f"An error occurred: {e}")  finally:  if driver:  driver.quit()  if \_\_name\_\_ == "\_\_main\_\_":  main() |
| **Input** | NA |
| **Output** |  |

|  |  |
| --- | --- |
| **Practical No: 15** | **Write a code to desired if the device sound or different modes is**  **on or off.** |
|  | <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>Practical15</groupId>  <artifactId>pra15</artifactId>  <version>0.0.1-SNAPSHOT</version>    <dependencies>  <!-- Appium Java Client -->  <dependency>  <groupId>io.appium</groupId>  <artifactId>java-client</artifactId>  <version>9.2.2</version>  </dependency>  <!-- Selenium Java -->  <dependency>  <groupId>org.seleniumhq.selenium</groupId>  <artifactId>selenium-java</artifactId>  <version>4.25.0</version>  </dependency>  </dependencies>  </project> |
|  | package pra15;  import io.appium.java\_client.android.AndroidDriver;  import org.openqa.selenium.remote.DesiredCapabilities;  import java.net.MalformedURLException;  import java.net.URL;  import java.util.Map;  public class Pra15 {  public static void main(String[] args) {  DesiredCapabilities capabilities11 = new DesiredCapabilities();  capabilities11.setCapability("platformName", "Android");  capabilities11.setCapability("deviceName", "Android Emulator");  capabilities11.setCapability("appPackage", "com.example.practical15");  capabilities11.setCapability("appActivity", ".MainActivity");  capabilities11.setCapability("automationName", "UiAutomator2");  capabilities11.setCapability("app",  "C:\\MASTER OF COMPUTER APPLICATIONS (MCA)\\MCA SEM 3\\MOBILE TESTING\\Practical15\\app\\build\\outputs\\apk\\debug\\testapp.apk");  capabilities11.setCapability("allowInsecure", "adb\_shell");  AndroidDriver driver = null;  try {  driver = new AndroidDriver(new URL("http://127.0.0.1:4723"), capabilities11);  // Execute shell command to get full audio dump  String command = "dumpsys audio";  String audioDump = driver.executeScript("mobile: shell",  Map.of("command", command)).toString();  // Print raw output for debugging  System.out.println("Audio Dump Output:\n" + audioDump);  // Check if the output contains the expected modes  if (audioDump.contains("MODE\_NORMAL")) {  System.out.println("The device is in NORMAL mode.");  } else if (audioDump.contains("ZEN\_MODE\_OFF")) {  System.out.println("The device is in ZEN mode (Do Not Disturb OFF).");  } else if (audioDump.contains("Ringer mode:")) {  if (audioDump.contains("NORMAL")) {  System.out.println("The ringer mode is NORMAL.");  } else if (audioDump.contains("SILENT")) {  System.out.println("The ringer mode is SILENT.");  } else if (audioDump.contains("VIBRATE")) {  System.out.println("The ringer mode is VIBRATE.");  } else {  System.out.println("Unknown ringer mode.");  }  } else {  System.out.println("Unknown audio mode.");  }  } catch (MalformedURLException e) {  e.printStackTrace();  } finally {  if (driver != null) {  driver.quit(); // Close the app  }  }  }  } |
|  | from appium import webdriver  import time  def main():  # Set up Desired Capabilities  desired\_caps = {  "platformName": "Android",  "deviceName": "Android Emulator",  "appPackage": "com.example.practical15",  "appActivity": ".MainActivity",  "automationName": "UiAutomator2",  "app": "C:\\MASTER OF COMPUTER APPLICATIONS (MCA)\\MCA SEM 3\\MOBILE TESTING\\Practical15\\app\\build\\outputs\\apk\\debug\\testapp.apk",  "allowInsecure": ["adb\_shell"]  }  # Initialize the Appium driver  driver = None  try:  driver = webdriver.Remote("http://127.0.0.1:4723 ", desired\_caps)  # Execute shell command to get the full audio dump  command = "dumpsys audio"  audio\_dump = driver.execute\_script("mobile: shell", {"command": command})  # Print raw output for debugging  print(f"Audio Dump Output:\n{audio\_dump}")  # Check for audio modes in the output  if "MODE\_NORMAL" in audio\_dump:  print("The device is in NORMAL mode.")  elif "ZEN\_MODE\_OFF" in audio\_dump:  print("The device is in ZEN mode (Do Not Disturb OFF).")  elif "Ringer mode:" in audio\_dump:  if "NORMAL" in audio\_dump:  print("The ringer mode is NORMAL.")  elif "SILENT" in audio\_dump:  print("The ringer mode is SILENT.")  elif "VIBRATE" in audio\_dump:  print("The ringer mode is VIBRATE.")  else:  print("Unknown ringer mode.")  else:  print("Unknown audio mode.")  except Exception as e:  print(f"An error occurred: {e}")  finally:  # Quit the driver to close the app  if driver:  driver.quit()  if \_\_name\_\_ == "\_\_main\_\_":  main() |
| **Input** | **NA** |
| **Output** |  |

|  |  |
| --- | --- |
| **Practical No: 16** | **Write a code to handle popup in Android App (ApiDemos.apk)**  **with model and nonmodel reply using UIautomator2.** |
|  | <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>Practical16</groupId>  <artifactId>pra16</artifactId>  <version>0.0.1-SNAPSHOT</version>    <dependencies>  <!-- Appium Java Client -->  <dependency>  <groupId>io.appium</groupId>  <artifactId>java-client</artifactId>  <version>9.2.2</version>  </dependency>  <!-- Selenium Java -->  <dependency>  <groupId>org.seleniumhq.selenium</groupId>  <artifactId>selenium-java</artifactId>  <version>4.25.0</version>  </dependency>  </dependencies>  </project> |
|  | package pra16;  import io.appium.java\_client.android.AndroidDriver;  import org.openqa.selenium.remote.DesiredCapabilities;  import org.openqa.selenium.By;  import org.openqa.selenium.WebElement;  import java.net.URL;  import java.net.MalformedURLException;  import java.util.concurrent.TimeUnit;  public class Pra16 {  private static AndroidDriver driver;  public static void main(String[] args) {  try {    setUp();      triggerPopup();    handleModalPopup();    driver.quit();  } catch (Exception e) {  e.printStackTrace();  }  }    public static void setUp() throws MalformedURLException {  DesiredCapabilities capabilities = new DesiredCapabilities();  capabilities.setCapability("platformName", "Android");  capabilities.setCapability("deviceName", "Android Emulator");  capabilities.setCapability("appPackage", "com.example.practical15");  capabilities.setCapability("appActivity", ".MainActivity");  capabilities.setCapability("automationName", "UiAutomator2");  capabilities.setCapability("app",  "C:\\MASTER OF COMPUTER APPLICATIONS (MCA)\\MCA SEM 3\\MOBILE TESTING\\Practical15\\app\\build\\outputs\\apk\\debug\\testapp.apk");      driver = new AndroidDriver(new URL("http://localhost:4723"), capabilities);  driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);  }    public static void triggerPopup() {    driver.findElement(By.id("com.example.practical15:id/show\_dialog\_button")).click();    }    public static void handleModalPopup() {  try {    WebElement okButton = driver.findElement(By.id("android:id/button1"));  okButton.click();  System.out.println("Popup handled successfully.");  } catch (Exception e) {  System.out.println("No popup found or unable to handle popup.");  e.printStackTrace();  }  }  } |
|  | from appium import webdriver from selenium.common.exceptions import NoSuchElementException import time  class Pra16:  def \_\_init\_\_(self):  self.driver = None   def set\_up(self):  desired\_caps = {  "platformName": "Android",  "deviceName": "Android Emulator",  "appPackage": "com.example.practical15",  "appActivity": ".MainActivity",  "automationName": "UiAutomator2",  "app": "C:\\MASTER OF COMPUTER APPLICATIONS (MCA)\\MCA SEM 3\\MOBILE TESTING\\Practical15\\app\\build\\outputs\\apk\\debug\\testapp.apk"  }   self.driver = webdriver.Remote("http://localhost:4723 ", desired\_caps)  self.driver.implicitly\_wait(10)   def trigger\_popup(self):  *"""Trigger the popup by clicking a button."""* self.driver.find\_element\_by\_id("com.example.practical15:id/show\_dialog\_button").click()   def handle\_modal\_popup(self):  try:  ok\_button = self.driver.find\_element\_by\_id("android:id/button1")  ok\_button.click()  print("Popup handled successfully.")  except NoSuchElementException:  print("No popup found or unable to handle popup.")   def quit\_driver(self): if self.driver:  self.driver.quit()  if \_\_name\_\_ == "\_\_main\_\_":  pra16 = Pra16()  try:  pra16.set\_up()  pra16.trigger\_popup()  pra16.handle\_modal\_popup()  except Exception as e:  print(f"An error occurred: {e}")  finally:  pra16.quit\_driver() |
| **Input** | NA |
| **Output** |  |

|  |  |
| --- | --- |
| **Practical No: 17** | **Write code to automate basic interactions, such as tapping a**  **button, entering text into a field, or swiping on the screen. (gmail,**  **facebook, instagram, whatsapp)** |
|  | <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>Practical17</groupId>  <artifactId>pra17</artifactId>  <version>0.0.1-SNAPSHOT</version>    <dependencies>  <!-- Appium Java Client -->  <dependency>  <groupId>io.appium</groupId>  <artifactId>java-client</artifactId>  <version>9.2.2</version>  </dependency>  <!-- Selenium Java -->  <dependency>  <groupId>org.seleniumhq.selenium</groupId>  <artifactId>selenium-java</artifactId>  <version>4.25.0</version>  </dependency>  </dependencies>  </project> |
|  | // For Whatsapp  package pra17;  import io.appium.java\_client.android.AndroidDriver;  import org.openqa.selenium.By;  import org.openqa.selenium.remote.DesiredCapabilities;  import org.openqa.selenium.support.ui.ExpectedConditions;  import org.openqa.selenium.support.ui.WebDriverWait;  import java.net.URL;  import java.util.HashMap;  import java.util.concurrent.TimeUnit;  public class pra17 {  private static AndroidDriver driver;  public static void main(String[] args) {  try {    automateWhatsApp();    driver.quit();  } catch (Exception e) {  e.printStackTrace();  }  }  // Set up driver for each app  public static void setUp(String appPackage, String appActivity) throws Exception {  DesiredCapabilities caps = new DesiredCapabilities();  caps.setCapability("platformName", "Android");  caps.setCapability("deviceName", "Android Emulator");  caps.setCapability("appPackage", "com.whatsapp");  caps.setCapability("appActivity", "com.whatsapp.Main");  caps.setCapability("noReset", true);  caps.setCapability("automationName", "UiAutomator2");  driver = new AndroidDriver<>(new URL("http://localhost:4723"), caps);  driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);  }  // Automate WhatsApp  public static void automateWhatsApp() throws Exception {  System.out.println("Launching WhatsApp...");  setUp("com.whatsapp", ".Main");  performBasicInteractions(By.id("com.whatsapp:id/menuitem\_search"),  By.id("com.whatsapp:id/search\_src\_text"));  }      // Perform basic interactions (tap button, enter text, swipe)  public static void performBasicInteractions(By buttonLocator, By textFieldLocator) {  try {  WebDriverWait wait = new WebDriverWait(driver, 15);  // Tap a button  MobileElement button = wait.until(ExpectedConditions.elementToBeClickable(buttonLocator));  button.click();  System.out.println("Tapping a Search button...");  // Enter text in a field  MobileElement textField = wait.until(ExpectedConditions.presenceOfElementLocated(textFieldLocator));  System.out.println("Entering text into a field...");  // Swipe the screen  swipeScreen();  } catch (Exception e) {  System.out.println("Failed to interact with the app.");  e.printStackTrace();  }  }  // Swipe on the screen (scroll down)  public static void swipeScreen() {  try {  HashMap<String, Object> swipe = new HashMap<>();  swipe.put("direction", "down");  swipe.put("percent", 0.7);  driver.executeScript("mobile: swipeGesture", swipe);  System.out.println("Swiping on the screen...");  } catch (Exception e) {  System.out.println("Failed to swipe on the screen.");  e.printStackTrace();  }  }  }  // For Facebook  package pra17;  import io.appium.java\_client.android.AndroidDriver;  import org.openqa.selenium.By;  import org.openqa.selenium.remote.DesiredCapabilities;  import org.openqa.selenium.support.ui.ExpectedConditions;  import org.openqa.selenium.support.ui.WebDriverWait;  import java.net.URL;  import java.util.HashMap;  import java.util.concurrent.TimeUnit;  public class pra17 {  private static AndroidDriver driver;  public static void main(String[] args) {  try {  // Automate Facebook app  automateFacebook();  driver.quit(); // Close the driver after the operation  } catch (Exception e) {  e.printStackTrace();  }  }  // Setup driver with app-specific capabilities  public static void setUp(String appPackage, String appActivity) throws Exception {  DesiredCapabilities caps = new DesiredCapabilities();  caps.setCapability("platformName", "Android");  caps.setCapability("deviceName", "Android Emulator");  caps.setCapability("automationName", "UiAutomator2");  caps.setCapability("appPackage", "com.facebook.katana");  caps.setCapability("appActivity", "com.facebook.katana.LoginActivity"); caps.setCapability("noReset", true);  driver = new AndroidDriver<>(new URL("http://localhost:4723/wd/hub"), caps);  driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);  }  // Automate Facebook interactions  public static void automateFacebook() throws Exception {  System.out.println("Launching Facebook...");  setUp("com.facebook.katana", "com.facebook.katana.LoginActivity");  performBasicInteractions(  By.id("com.facebook.katana:id/login\_username"),  By.id("com.facebook.katana:id/login\_password")  );  }  // Perform basic interactions (tap, enter text, swipe)  public static void performBasicInteractions(By usernameFieldLocator, By passwordFieldLocator) {  try {  WebDriverWait wait = new WebDriverWait(driver, 15);  // Enter text into the username field  MobileElement usernameField = wait.until(ExpectedConditions.presenceOfElementLocated(usernameFieldLocator));  usernameField.sendKeys("test\_user");  System.out.println("Entered text into the username field.");  // Tap the password field  MobileElement passwordField = wait.until(ExpectedConditions.elementToBeClickable(passwordFieldLocator));  passwordField.click();  System.out.println("Tapped the password field.");  // Swipe on the screen (example swipe action)  swipeScreen();  } catch (Exception e) {  System.out.println("Failed to interact with the app.");  e.printStackTrace();  }  }  // Swipe action using TouchAction  public static void swipeScreen() {  try {  TouchAction<?> action = new TouchAction<>(driver);  action  .press(PointOption.point(500, 1000))  .moveTo(PointOption.point(500, 500))  .release()  .perform();  System.out.println("Swiped on the screen.");  } catch (Exception e) {  System.out.println("Failed to perform swipe action.");  e.printStackTrace();  }  }    }   // for Instagram  package pra17;  import io.appium.java\_client.android.AndroidDriver;  import org.openqa.selenium.By;  import org.openqa.selenium.remote.DesiredCapabilities;  import org.openqa.selenium.support.ui.ExpectedConditions;  import org.openqa.selenium.support.ui.WebDriverWait;  import java.net.URL;  import java.util.HashMap;  import java.util.concurrent.TimeUnit;  public class pra17 {  private static AndroidDriver driver;  public static void main(String[] args) {  try {  // Automate Instagram app  automateInstagram();  driver.quit(); // Close the driver after operation  } catch (Exception e) {  e.printStackTrace();  }  }  // Setup driver with app-specific capabilities  public static void setUp(String appPackage, String appActivity) throws Exception {  DesiredCapabilities caps = new DesiredCapabilities();  caps.setCapability("platformName", "Android");  caps.setCapability("deviceName", "Android Emulator");  caps.setCapability("automationName", "UiAutomator2");  caps.setCapability("appPackage", "com.instagram.android");  caps.setCapability("appActivity", "com.instagram.android.activity.MainTabActivity");  caps.setCapability("noReset", true); // Keeps user session  driver = new AndroidDriver<>(new URL("http://localhost:4723/wd/hub"), caps);  driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);  }  // Automate Instagram interactions  public static void automateInstagram() throws Exception {  System.out.println("Launching Instagram...");  setUp("com.instagram.android", "com.instagram.mainactivity.MainActivity");  performBasicInteractions(  By.id("com.instagram.android:id/login\_username"),  By.id("com.instagram.android:id/login\_password")  );  }  // Perform basic interactions (tap, enter text, swipe)  public static void performBasicInteractions(By usernameFieldLocator, By passwordFieldLocator) {  try {  WebDriverWait wait = new WebDriverWait(driver, 15);  // Enter text into the username field  MobileElement usernameField = wait.until(ExpectedConditions.presenceOfElementLocated(usernameFieldLocator));  usernameField.sendKeys("user123");  System.out.println("Entered text into the username field.");  // Tap the password field  MobileElement passwordField = wait.until(ExpectedConditions.elementToBeClickable(passwordFieldLocator));  passwordField.click();  System.out.println("Tapped the password field.");  // Swipe on the screen (example swipe action)  swipeScreen();  } catch (Exception e) {  System.out.println("Failed to interact with the app.");  e.printStackTrace();  }  }  // Swipe action using TouchAction  public static void swipeScreen() {  try {  TouchAction<?> action = new TouchAction<>(driver);  action  .press(PointOption.point(500, 1000)) // Start point (x, y)  .moveTo(PointOption.point(500, 500)) // End point (x, y)  .release()  .perform();  System.out.println("Swiped the screen.");  } catch (Exception e) {  System.out.println("Failed to perform swipe action.");  e.printStackTrace();  }  }    }  // For Gmail  package pra17;  import io.appium.java\_client.android.AndroidDriver;  import org.openqa.selenium.By;  import org.openqa.selenium.remote.DesiredCapabilities;  import org.openqa.selenium.support.ui.ExpectedConditions;  import org.openqa.selenium.support.ui.WebDriverWait;  import java.net.URL;  import java.util.HashMap;  import java.util.concurrent.TimeUnit;  public class pra17 {  private static AndroidDriver driver;  public static void main(String[] args) {  try {  // Automate Gmail app interactions  automateGmail();  driver.quit(); // Close the driver after operation  } catch (Exception e) {  e.printStackTrace();  }  }  // Setup driver with Gmail-specific capabilities  public static void setUp(String appPackage, String appActivity) throws Exception {  DesiredCapabilities caps = new DesiredCapabilities();  caps.setCapability("platformName", "Android");  caps.setCapability("deviceName", "Android Emulator");  caps.setCapability("automationName", "UiAutomator2");  caps.setCapability("appPackage", "com.google.android.gm");  caps.setCapability("appActivity", "com.google.android.gm.ConversationListActivityGmail"); caps.setCapability("noReset", true); // Keeps the Gmail session active  driver = new AndroidDriver<>(new URL("http://localhost:4723/wd/hub"), caps);  driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);  }  // Automate Gmail interactions  public static void automateGmail() throws Exception {  System.out.println("Launching Gmail...");  setUp("com.google.android.gm", "com.google.android.gm.ConversationListActivityGmail");  performBasicInteractions(  By.id("com.google.android.gm:id/compose\_button"),  By.id("com.google.android.gm:id/to")  );  }  // Perform basic interactions (tap, enter text, swipe)  public static void performBasicInteractions(By buttonLocator, By textFieldLocator) {  try {  WebDriverWait wait = new WebDriverWait(driver, 15);  // Tap the compose button  MobileElement composeButton = wait.until(ExpectedConditions.elementToBeClickable(buttonLocator));  composeButton.click();  System.out.println("Tapped the Compose button.");  // Enter text into the recipient field  MobileElement recipientField = wait.until(ExpectedConditions.presenceOfElementLocated(textFieldLocator));  recipientField.sendKeys("example@gmail.com");  System.out.println("Entered recipient email.");  // Swipe the screen (example swipe)  swipeScreen();  } catch (Exception e) {  System.out.println("Failed to interact with Gmail.");  e.printStackTrace();  }  }  // Swipe action using TouchAction  public static void swipeScreen() {  try {  TouchAction<?> action = new TouchAction<>(driver);  action  .press(PointOption.point(500, 1000))  .moveTo(PointOption.point(500, 500))  .release()  .perform();  System.out.println("Swiped on the screen.");  } catch (Exception e) {  System.out.println("Failed to perform swipe action.");  e.printStackTrace();  }  }    } |
|  | from appium import webdriver from appium.webdriver.common.appiumby import AppiumBy from selenium.webdriver.support.ui import WebDriverWait from selenium.webdriver.support import expected\_conditions as EC import time  def setUp(app\_package, app\_activity):  caps = {  "platformName": "Android",  "deviceName": "Android Emulator",  "appPackage": app\_package,  "appActivity": app\_activity,  "noReset": True,  "automationName": "UiAutomator2"  }  return webdriver.Remote("http://localhost:4723 ", caps)  def automate\_whatsapp(driver):  print("Launching WhatsApp...")  wait = WebDriverWait(driver, 10)  search\_button = wait.until(EC.element\_to\_be\_clickable((AppiumBy.ID, "com.whatsapp:id/menuitem\_search")))  search\_button.click()  print("Tapped search button.")   search\_field = wait.until(EC.presence\_of\_element\_located((AppiumBy.ID, "com.whatsapp:id/search\_src\_text")))  search\_field.send\_keys("Hello")  print("Entered text into search field.")  time.sleep(2)  def main():  driver = setUp("com.whatsapp", "com.whatsapp.Main")  try:  automate\_whatsapp(driver)  finally:  driver.quit()  if \_\_name\_\_ == "\_\_main\_\_":  main()  def automate\_facebook(driver):  print("Launching Facebook...")  wait = WebDriverWait(driver, 10)   username\_field = wait.until(EC.presence\_of\_element\_located((AppiumBy.ID, "com.facebook.katana:id/login\_username")))  username\_field.send\_keys("test\_user")  print("Entered username.")   password\_field = wait.until(EC.element\_to\_be\_clickable((AppiumBy.ID, "com.facebook.katana:id/login\_password")))  password\_field.click()  print("Tapped password field.")  time.sleep(2)  def main():  driver = setUp("com.facebook.katana", "com.facebook.katana.LoginActivity")  try:  automate\_facebook(driver)  finally:  driver.quit()  if \_\_name\_\_ == "\_\_main\_\_":  main()  def automate\_instagram(driver):  print("Launching Instagram...")  wait = WebDriverWait(driver, 10)   username\_field = wait.until(EC.presence\_of\_element\_located((AppiumBy.ID, "com.instagram.android:id/login\_username")))  username\_field.send\_keys("user123")  print("Entered username.")   password\_field = wait.until(EC.element\_to\_be\_clickable((AppiumBy.ID, "com.instagram.android:id/login\_password")))  password\_field.click()  print("Tapped password field.")  time.sleep(2)  def main():  driver = setUp("com.instagram.android", "com.instagram.mainactivity.MainActivity")  try:  automate\_instagram(driver)  finally:  driver.quit()  if \_\_name\_\_ == "\_\_main\_\_":  main()  def automate\_gmail(driver):  print("Launching Gmail...")  wait = WebDriverWait(driver, 10)  compose\_button = wait.until(EC.element\_to\_be\_clickable((AppiumBy.ID, "com.google.android.gm:id/compose\_button")))  compose\_button.click()  print("Tapped Compose button.")  to\_field = wait.until(EC.presence\_of\_element\_located((AppiumBy.ID, "com.google.android.gm:id/to")))  to\_field.send\_keys("example@gmail.com")  print("Entered recipient email.")  time.sleep(2)  def main():  driver = setUp("com.google.android.gm", "com.google.android.gm.ConversationListActivityGmail")  try:  automate\_gmail(driver)  finally:  driver.quit()  if \_\_name\_\_ == "\_\_main\_\_":  main() |
| **Input** | NA |
| **Output** | |  |  | | --- | --- | |  |  | |  |  | |

|  |  |
| --- | --- |
| **Practical No: 18** | **Write a script in Java to create a test case to perform Drag and**  **Drop on elements in App using Appium.** |
|  | <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>Practical18</groupId>  <artifactId>pra18</artifactId>  <version>0.0.1-SNAPSHOT</version>    <dependencies>  <!-- Appium Java Client -->  <dependency>  <groupId>io.appium</groupId>  <artifactId>java-client</artifactId>  <version>9.2.2</version>  </dependency>  <!-- Selenium Java -->  <dependency>  <groupId>org.seleniumhq.selenium</groupId>  <artifactId>selenium-java</artifactId>  <version>4.25.0</version>  </dependency>  </dependencies>  </project> |
|  | <?xml version="1.0" encoding="utf-8"?> <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:padding="16dp">   <!-- Draggable View (Source) -->  <TextView  android:id="@+id/source"  android:layout\_width="150dp"  android:layout\_height="150dp"  android:background="#FF5722"  android:gravity="center"  android:text="Drag Me"  android:textColor="#FFFFFF"  android:textSize="18sp"  android:layout\_alignParentTop="true"  android:layout\_centerHorizontal="true"  android:layout\_marginTop="100dp"/>   <!-- Drop Target View -->  <TextView  android:id="@+id/target"  android:layout\_width="150dp"  android:layout\_height="150dp"  android:background="#4CAF50"  android:gravity="center"  android:text="Drop Here"  android:textColor="#FFFFFF"  android:textSize="18sp"  android:layout\_alignParentBottom="true"  android:layout\_centerHorizontal="true"  android:layout\_marginBottom="100dp"/>  </RelativeLayout> |
|  | package com.example.practical15;  import androidx.appcompat.app.AppCompatActivity;  import android.os.Bundle;  import android.view.DragEvent;  import android.view.View;  import android.widget.TextView;  import android.widget.Toast;  public class MainActivity extends AppCompatActivity {  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main);  // Get the source and target views  TextView source = findViewById(R.id.source);  TextView target = findViewById(R.id.target);  // Set a long click listener on the source view to start the drag  source.setOnLongClickListener(view -> {  View.DragShadowBuilder shadowBuilder = new View.DragShadowBuilder(view);  view.startDrag(null, shadowBuilder, view, 0);  return true;  });  // Set a drag listener on the target view to handle the drop  target.setOnDragListener((view, dragEvent) -> {  switch (dragEvent.getAction()) {  case DragEvent.ACTION\_DRAG\_STARTED:  return true; // Indicate that we accept the drag event  case DragEvent.ACTION\_DRAG\_ENTERED:  target.setBackgroundColor(0xFF9E9E9E); // Change color to indicate active drop area  return true;  case DragEvent.ACTION\_DRAG\_EXITED:  target.setBackgroundColor(0xFF4CAF50); // Revert to original color  return true;  case DragEvent.ACTION\_DROP:  View draggedView = (View) dragEvent.getLocalState();  draggedView.setVisibility(View.INVISIBLE); // Hide the dragged view  Toast.makeText(this, "Dropped Successfully!", Toast.LENGTH\_SHORT).show();  return true;  case DragEvent.ACTION\_DRAG\_ENDED:  target.setBackgroundColor(0xFF4CAF50); // Reset color after drop  return true;  default:  return false;  }  });  }  } |
|  | package pra18;  import io.appium.java\_client.AppiumBy;  import io.appium.java\_client.android.AndroidDriver;  import io.appium.java\_client.touch.offset.ElementOption;  import io.appium.java\_client.TouchAction;  import org.openqa.selenium.WebElement;  import org.openqa.selenium.remote.DesiredCapabilities;  import java.net.MalformedURLException;  import java.net.URL;  import java.time.Duration;  public class pra18 {  private static AndroidDriver driver;    public static void main(String[] args) {  try {  // Step 1: Initialize Appium Driver with Desired Capabilities  setUp();  // Step 2: Perform Drag and Drop  performDragAndDrop();  } catch (Exception e) {  e.printStackTrace();  } finally {  // Step 3: Close the driver  if (driver != null) {  driver.quit();  }  }}    private static void setUp() throws MalformedURLException {  DesiredCapabilities capabilities11 = new DesiredCapabilities();  capabilities11.setCapability("platformName", "Android");  capabilities11.setCapability("deviceName", "Android Emulator");  capabilities11.setCapability("appPackage", "com.example.practical15");  capabilities11.setCapability("appActivity", ".MainActivity");  capabilities11.setCapability("automationName", "UiAutomator2");  capabilities11.setCapability("app",  "C:\\MASTER OF COMPUTER APPLICATIONS (MCA)\\MCA SEM 3\\MOBILE TESTING\\Practical15\\app\\build\\outputs\\apk\\debug\\testapp.apk");    // Initialize Android Driver  driver = new AndroidDriver(new URL("http://127.0.0.1:4723"), capabilities11);  driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));  }  private static void performDragAndDrop() {  // Locate the source (draggable view)  WebElement source = driver.findElement(AppiumBy.id("com.example.practical15:id/source"));  // Locate the target (drop area)  WebElement target = driver.findElement(AppiumBy.id("com.example.practical15:id/target"));  // Perform drag and drop action using TouchAction  TouchAction<?> action = new TouchAction<>(driver)  .longPress(ElementOption.element(source))  .moveTo(ElementOption.element(target))  .release();  action.perform();    System.out.println("Drag and Drop performed successfully!");  }    } |
|  | from appium import webdriver from appium.webdriver.common.appiumby import AppiumBy from appium.webdriver.common.touch\_action import TouchAction import time  def set\_up():  # Desired Capabilities  desired\_caps = {  "platformName": "Android",  "deviceName": "Android Emulator",  "appPackage": "com.example.practical15",  "appActivity": ".MainActivity",  "automationName": "UiAutomator2",  "app": "C:\\MASTER OF COMPUTER APPLICATIONS (MCA)\\MCA SEM 3\\MOBILE TESTING\\Practical15\\app\\build\\outputs\\apk\\debug\\testapp.apk"  }   # Initialize the driver  driver = webdriver.Remote("http://127.0.0.1:4723 ", desired\_caps)  driver.implicitly\_wait(10)  return driver  def perform\_drag\_and\_drop(driver):  try:  # Locate the source (draggable view)  source = driver.find\_element(AppiumBy.ID, "com.example.practical15:id/source")   # Locate the target (drop area)  target = driver.find\_element(AppiumBy.ID, "com.example.practical15:id/target")   # Perform drag and drop action using TouchAction  action = TouchAction(driver)  action.long\_press(source).move\_to(target).release().perform()   print("Drag and Drop performed successfully!")   except Exception as e:  print(f"Error during drag and drop: {e}")  def main():  driver = None  try:  # Step 1: Set up Appium driver  driver = set\_up()   # Step 2: Perform drag and drop  perform\_drag\_and\_drop(driver)   finally:  # Step 3: Close the driver  if driver:  driver.quit()  if \_\_name\_\_ == "\_\_main\_\_":  main() |
| **Input** | NA |
| **Output** | |  |  | | --- | --- | |  |  | |

|  |  |
| --- | --- |
| **Practical No: 19** | **Write a script in Java to create a test case to Automate**  **Miscellaneous Activities like orientation & KeyEvents using Appium.** |
|  | <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>Practical19</groupId>  <artifactId>pra19</artifactId>  <version>0.0.1-SNAPSHOT</version>    <dependencies>  <!-- Appium Java Client -->  <dependency>  <groupId>io.appium</groupId>  <artifactId>java-client</artifactId>  <version>9.2.2</version>  </dependency>  <!-- Selenium Java -->  <dependency>  <groupId>org.seleniumhq.selenium</groupId>  <artifactId>selenium-java</artifactId>  <version>4.25.0</version>  </dependency>  </dependencies>  </project> |
|  | package pra19;  import io.appium.java\_client.AppiumBy;  import io.appium.java\_client.android.AndroidDriver;  import org.openqa.selenium.remote.DesiredCapabilities;  import java.net.MalformedURLException;  import java.net.URL;  import java.time.Duration;  public class pra19 {  private static AndroidDriver driver;  public static void main(String[] args) {  try {  setUp(); // Initialize driver  changeOrientation(); // Change orientation  sendKeyEvents(); // Perform key events  } catch (Exception e) {  e.printStackTrace();  } finally {  if (driver != null) {  driver.quit(); // Close driver  }  }  }    private static void setUp() throws MalformedURLException {  DesiredCapabilities capabilities11 = new DesiredCapabilities();  capabilities11.setCapability("platformName", "Android");  capabilities11.setCapability("deviceName", "Android Emulator");  capabilities11.setCapability("appPackage", "com.example.practical19");  capabilities11.setCapability("appActivity", ".MainActivity");  capabilities11.setCapability("automationName", "UiAutomator2");  capabilities11.setCapability("app",  "C:\\MASTER OF COMPUTER APPLICATIONS (MCA)\\MCA SEM 3\\MOBILE TESTING\\Practical19\\app\\build\\outputs\\apk\\debug\\app.apk");    driver = new AndroidDriver(new URL("http://127.0.0.1:4723"), capabilities11);  driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));  }  private static void changeOrientation() {  System.out.println("Changing to landscape...");  driver.rotate(org.openqa.selenium.ScreenOrientation.LANDSCAPE);  try { Thread.sleep(3000); } catch (InterruptedException ignored) {}  System.out.println("Changing back to portrait...");  driver.rotate(org.openqa.selenium.ScreenOrientation.PORTRAIT);  }  private static void sendKeyEvents() {  System.out.println("Pressing HOME button...");  driver.pressKey(new io.appium.java\_client.android.nativekey.KeyEvent(io.appium.java\_client.android.nativekey.AndroidKey.HOME));  try { Thread.sleep(3000); } catch (InterruptedException ignored) {}  System.out.println("Pressing BACK button...");  driver.pressKey(new io.appium.java\_client.android.nativekey.KeyEvent(io.appium.java\_client.android.nativekey.AndroidKey.BACK));  }  } |
|  | from appium import webdriver from appium.webdriver.common.appiumby import AppiumBy from appium.webdriver.common.touch\_action import TouchAction from appium.webdriver.common.mobileby import MobileBy import time  def set\_up():  # Desired Capabilities  desired\_caps = {  "platformName": "Android",  "deviceName": "Android Emulator",  "appPackage": "com.example.practical19",  "appActivity": ".MainActivity",  "automationName": "UiAutomator2",  "app": "C:\\MASTER OF COMPUTER APPLICATIONS (MCA)\\MCA SEM 3\\MOBILE TESTING\\Practical19\\app\\build\\outputs\\apk\\debug\\app.apk"  }   # Initialize the driver  driver = webdriver.Remote("http://127.0.0.1:4723 ", desired\_caps)  driver.implicitly\_wait(10)  return driver  def change\_orientation(driver):  print("Changing to landscape...")  driver.orientation = "LANDSCAPE"  time.sleep(3) # Wait for 3 seconds   print("Changing back to portrait...")  driver.orientation = "PORTRAIT"  def send\_key\_events(driver):  print("Pressing HOME button...")  driver.press\_keycode(3)    time.sleep(3) # Wait for 3 seconds   print("Pressing BACK button...")  driver.press\_keycode(4) # Keycode for BACK  def main():  driver = None  try:  # Step 1: Set up Appium driver  driver = set\_up()   # Step 2: Change device orientation  change\_orientation(driver)   # Step 3: Send key events  send\_key\_events(driver)   finally:  # Close the driver  if driver:  driver.quit()  if \_\_name\_\_ == "\_\_main\_\_":  main() |
| **Input** | **NA** |
| **Output** |  |

|  |  |
| --- | --- |
| **Practical No: 20** | **Write a script in Java to create a test case for Verifying toast**  **messages for error validations using Appium.** |
|  | <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>Practical20</groupId>  <artifactId>pra20</artifactId>  <version>0.0.1-SNAPSHOT</version>    <dependencies>  <!-- Appium Java Client -->  <dependency>  <groupId>io.appium</groupId>  <artifactId>java-client</artifactId>  <version>9.2.2</version>  </dependency>  <!-- Selenium Java -->  <dependency>  <groupId>org.seleniumhq.selenium</groupId>  <artifactId>selenium-java</artifactId>  <version>4.25.0</version>  </dependency>  </dependencies>  </project> |
|  | package pra20;  import io.appium.java\_client.AppiumBy;  import io.appium.java\_client.android.AndroidDriver;  import org.openqa.selenium.WebElement;  import org.openqa.selenium.remote.DesiredCapabilities;  import org.openqa.selenium.support.ui.ExpectedConditions;  import org.openqa.selenium.support.ui.WebDriverWait;  import java.net.MalformedURLException;  import java.net.URL;  import java.time.Duration;  public class pra20 {  private static AndroidDriver driver;  public static void main(String[] args) {  try {  setUp(); // Initialize driver  triggerToastMessage(); // Trigger the error toast message  verifyToastMessage("Expected error message"); // Verify the toast message  } catch (Exception e) {  e.printStackTrace();  } finally {  if (driver != null) {  driver.quit(); // Close driver  }  }  }  private static void setUp() throws MalformedURLException {  DesiredCapabilities capabilities = new DesiredCapabilities();  capabilities.setCapability("platformName", "Android");  capabilities.setCapability("deviceName", "Android Emulator");  capabilities.setCapability("appPackage", "com.example.practical19");  capabilities.setCapability("appActivity", ".MainActivity");  capabilities.setCapability("automationName", "UiAutomator2");  capabilities.setCapability("app",  "C:\\MASTER OF COMPUTER APPLICATIONS (MCA)\\MCA SEM 3\\MOBILE TESTING\\Practical19\\app\\build\\outputs\\apk\\debug\\app.apk");  driver = new AndroidDriver(new URL("http://127.0.0.1:4723"), capabilities);  driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));  }  private static void triggerToastMessage() {  // Explicit wait to ensure elements are available before interacting  WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));  WebElement inputField = wait.until(ExpectedConditions.visibilityOfElementLocated(AppiumBy.id("com.example.practical19:id/inputField")));  WebElement submitButton = wait.until(ExpectedConditions.visibilityOfElementLocated(AppiumBy.id("com.example.practical19:id/submitButton")));  // Clear the input field and click submit to trigger the toast  inputField.clear();  submitButton.click();  }  private static void verifyToastMessage(String expectedMessage) {  // Locate the toast message using XPath; using "contains" for flexibility  WebElement toastMessage = driver.findElement(AppiumBy.xpath("//android.widget.Toast[1]"));  // Retrieve the toast message text and verify it  String actualMessage = toastMessage.getAttribute("name");  if (actualMessage.equals(expectedMessage)) {  System.out.println("Toast message verification passed!");  } else {  System.out.println("Toast message verification failed!");  System.out.println("Expected: " + expectedMessage);  System.out.println("Actual: " + actualMessage);  }  }  } |
|  | <?xml version="1.0" encoding="utf-8"?> <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:padding="16dp">   <EditText  android:id="@+id/inputField"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:hint="Enter some text" />   <Button  android:id="@+id/submitButton"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="Submit"  android:layout\_below="@id/inputField"  android:layout\_marginTop="16dp"  android:layout\_alignParentEnd="true" /> </RelativeLayout> |
|  | package com.example.practical19;  import androidx.appcompat.app.AppCompatActivity;  import android.os.Bundle; import android.view.View; import android.widget.Button; import android.widget.EditText; import android.widget.Toast;  public class MainActivity extends AppCompatActivity {  private EditText inputField; // Declare inputField here  private Button submitButton;  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_main*);  // Correctly cast inputField as EditText  inputField = (EditText) findViewById(R.id.*inputField*);  submitButton = (Button) findViewById(R.id.*submitButton*);   submitButton.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  // Check if inputField is empty and show a toast message  if (inputField.getText().toString().isEmpty()) {  Toast.*makeText*(MainActivity.this, "Input cannot be empty!", Toast.*LENGTH\_SHORT*).show();  } else {  Toast.*makeText*(MainActivity.this, "Input submitted!", Toast.*LENGTH\_SHORT*).show();  }  }  });  } } |
|  | from appium import webdriver  from appium.webdriver.common.appiumby import AppiumBy  from appium.webdriver.common.mobileby import MobileBy  from selenium.webdriver.support.ui import WebDriverWait  from selenium.webdriver.support import expected\_conditions as EC  import time  def set\_up():  # Desired Capabilities  desired\_caps = {  "platformName": "Android",  "deviceName": "Android Emulator",  "appPackage": "com.example.practical19",  "appActivity": ".MainActivity",  "automationName": "UiAutomator2",  "app": "C:\\MASTER OF COMPUTER APPLICATIONS (MCA)\\MCA SEM 3\\MOBILE TESTING\\Practical19\\app\\build\\outputs\\apk\\debug\\app.apk"  }  # Initialize the driver  driver = webdriver.Remote("http://127.0.0.1:4723/wd/hub", desired\_caps)  driver.implicitly\_wait(10)  return driver  def trigger\_toast\_message(driver):  # Explicit wait to ensure elements are available before interacting  wait = WebDriverWait(driver, 10)  input\_field = wait.until(EC.visibility\_of\_element\_located(AppiumBy.ID, "com.example.practical19:id/inputField"))  submit\_button = wait.until(EC.visibility\_of\_element\_located(AppiumBy.ID, "com.example.practical19:id/submitButton"))  # Clear the input field and click submit to trigger the toast  input\_field.clear()  submit\_button.click()  def verify\_toast\_message(driver, expected\_message):  # Locate the toast message using XPath; using "contains" for flexibility  toast\_message = driver.find\_element(AppiumBy.XPATH, "//android.widget.Toast[1]")  # Retrieve the toast message text and verify it  actual\_message = toast\_message.get\_attribute("name")  if actual\_message == expected\_message:  print("Toast message verification passed!")  else:  print("Toast message verification failed!")  print(f"Expected: {expected\_message}")  print(f"Actual: {actual\_message}")  def main():  driver = None  try:  # Step 1: Set up Appium driver  driver = set\_up()  # Step 2: Trigger the toast message  trigger\_toast\_message(driver)  # Step 3: Verify the toast message  verify\_toast\_message(driver, "Expected error message") # Change this to your actual expected message  finally:  # Close the driver  if driver:  driver.quit()  if \_\_name\_\_ == "\_\_main\_\_":  main() |
| **Input** | NA |
| **Output** |  |