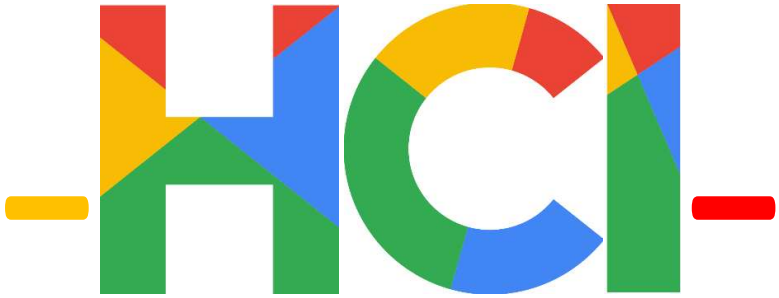


NATIONAL UNIVERSITY
of Computer & Emerging Sciences

Course: CS2007 Human-Computer Interaction

Instructor: Lecturer Zeeshan Khan

Submitted by:
Muhammad Rehan | 22P-9106
Zainab Riaz | 22P-9133

Class: BSE-3A

Date: Nov 23th, 2023.

Project Stage 4

1. Evaluate the Pinterest UI using Appium

Appium, an open-source tool, enables automated testing across various platforms—iOS, Android, and Windows desktop—for native, mobile web, and hybrid applications.

It supports multiple programming languages like Java, PHP, Perl, Python, allowing users to script in their preferred language, but we will be using java.

Reasons for choosing Appium:

- **Open Source:** Free to use and modify.
- **Cross Platform:** Works on Windows, Mac, and Linux, testing native, hybrid, or web apps on Android or iOS without needing access to source code.
- **No Reinstallation:** Unlike other tools, Appium doesn't require reinstalling the app each time it's tested.
- **Framework Support:** Offers flexibility in choosing the programming language for automation.

The initial step involved analyzing the potential problems that could be examined using an app like Appium while running a test script.

Here are the key points we will be testing through:

Pin-S1: Appium can validate the accurate display of the system status indicator or icon after saving a pin. Additionally, it can verify that the page does not refresh automatically upon returning to the top.

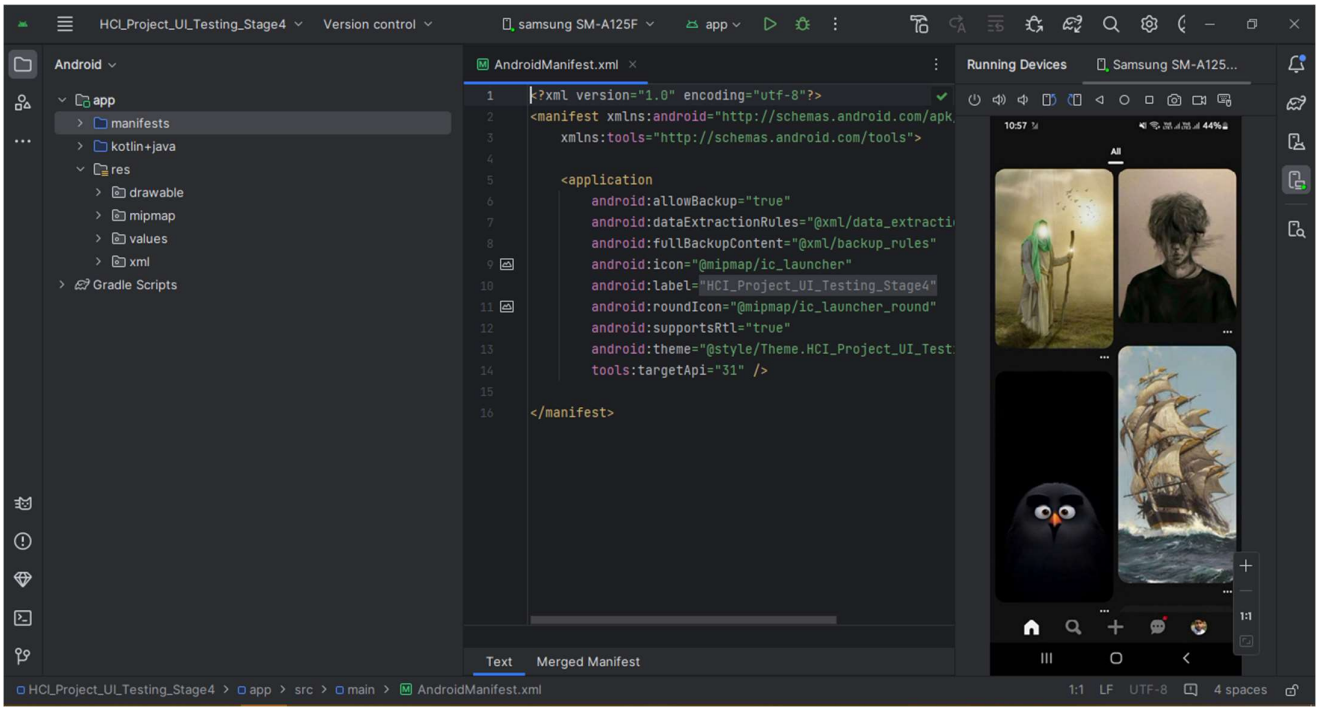
Pin-S12: Appium can assess the functionality of customizing notification preferences and confirming if notifications appear alongside the pins.

Pin-S3: Appium can test the user-specific board creation feature, aiming to enhance user recognition and recall of their pins.

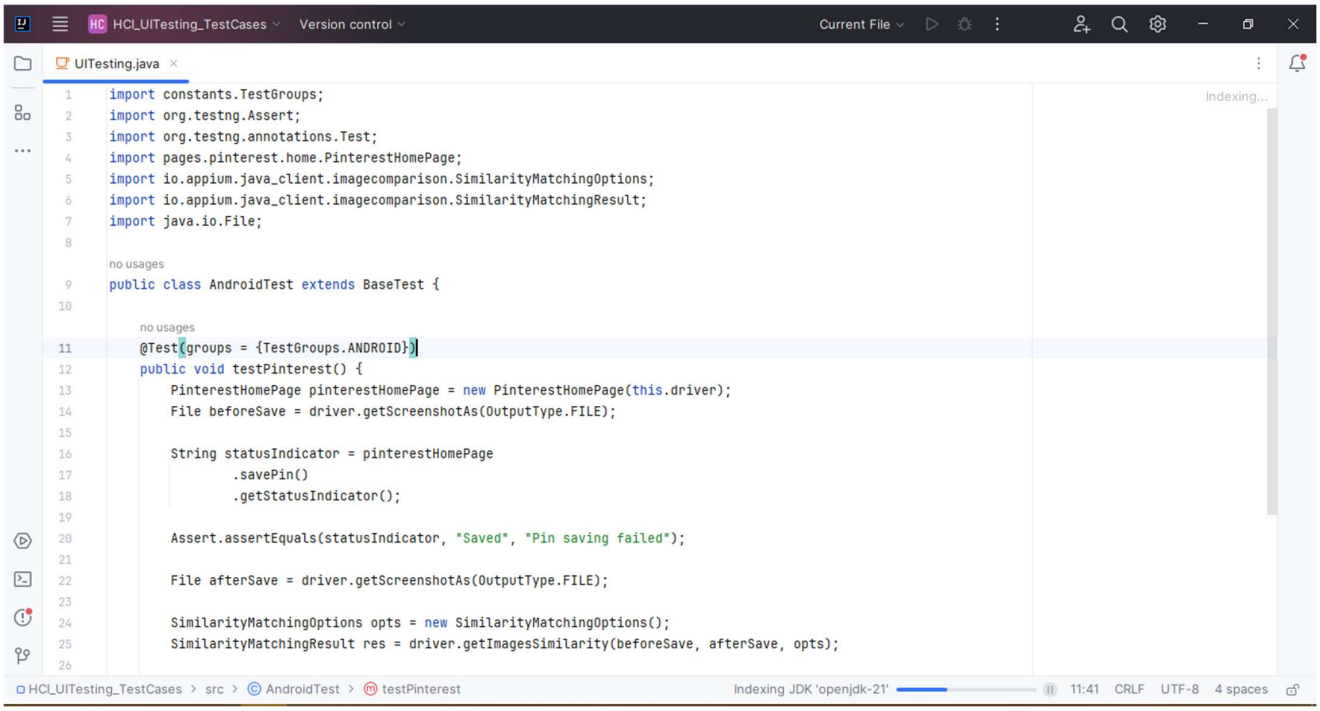
2. Describing the Testing Process

Tech Stack | Development Environment

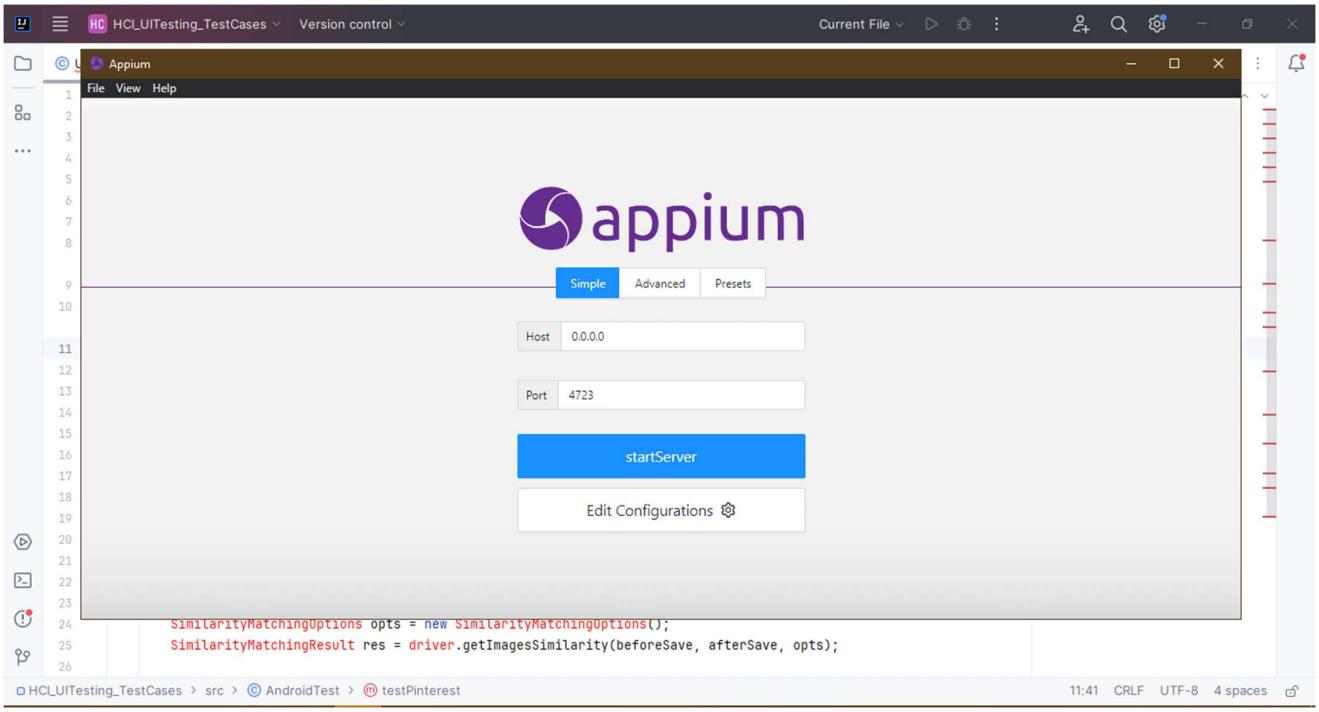
Android Studio Hedgehog 2023.1.1: This IDE was utilized to run our emulator. We also used a Samsung device for app testing. It was directly interfaced with the SDK components of Appium for device interactions.



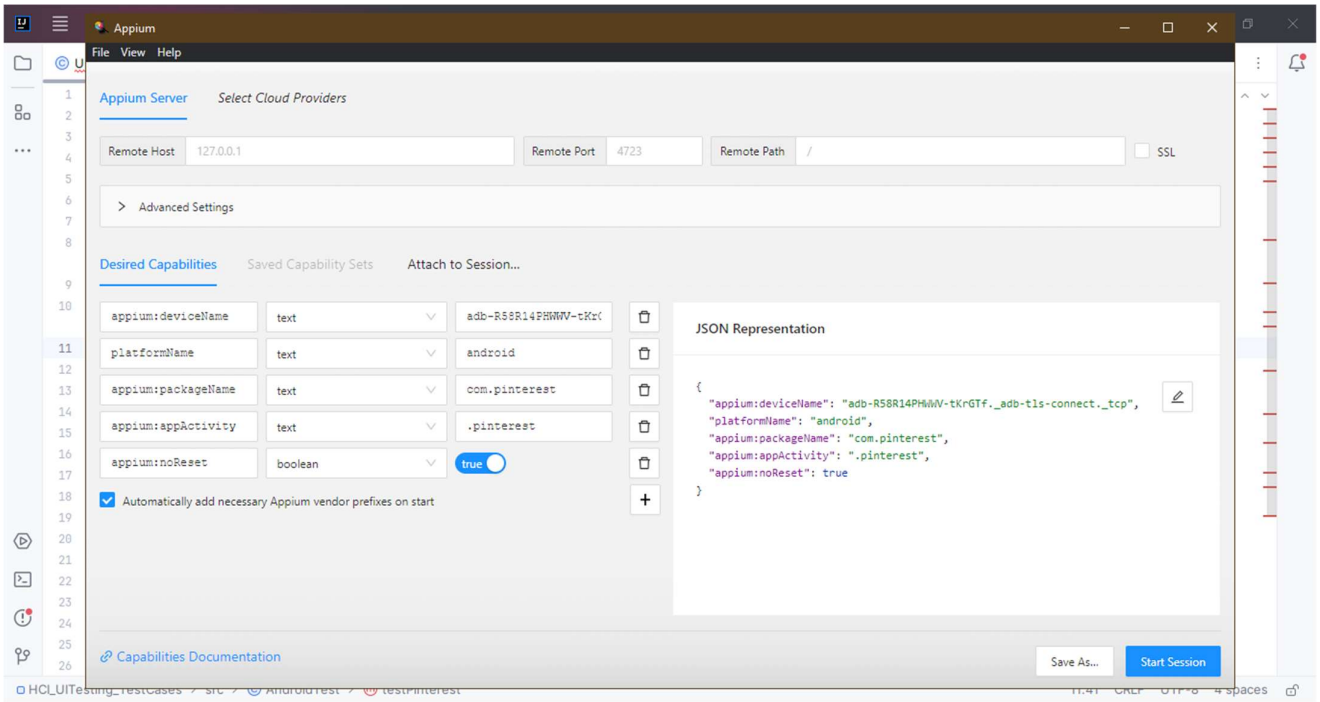
IntelliJ IDEA: This comprehensive IDE was used to write and automate our test cases. It facilitated the import of necessary libraries for writing test scripts.



Appium: Appium acted as our server setup tool, enabling **ADB** drivers to control our device through wireless debugging.



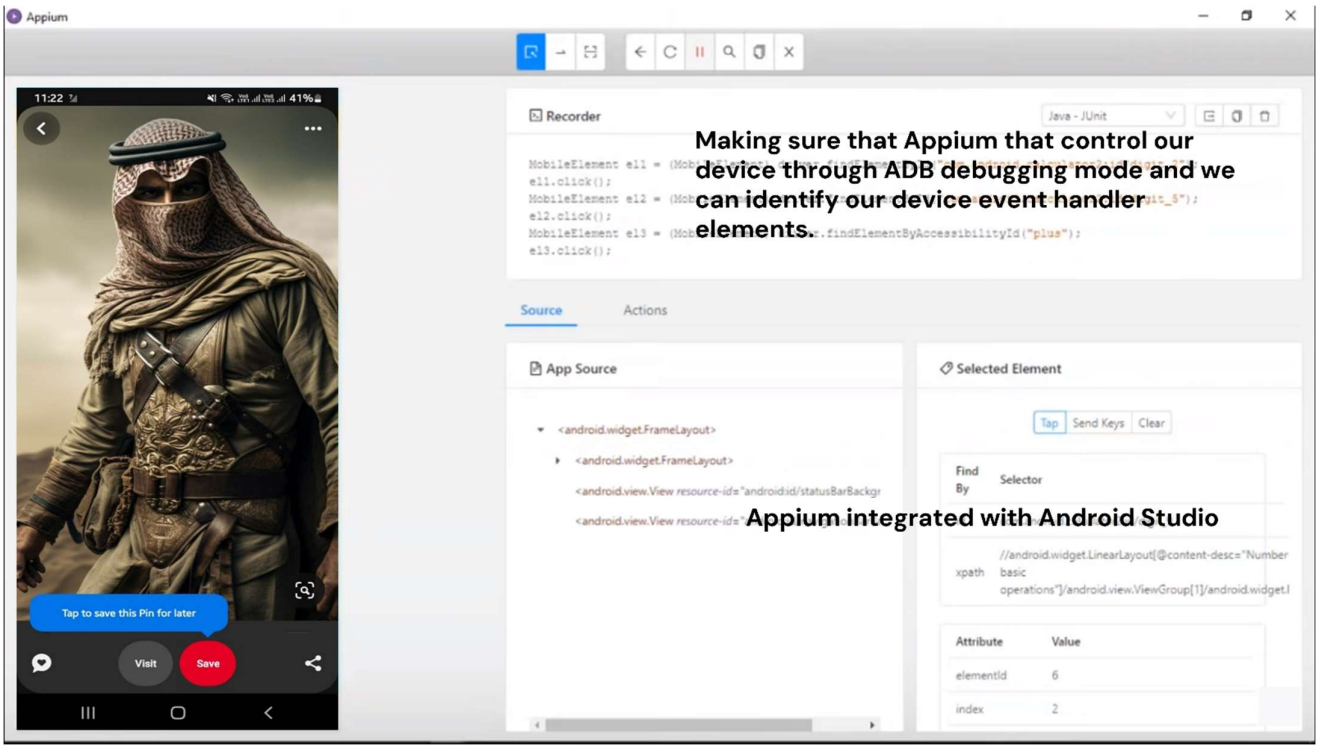
Appium Inspector: This tool was instrumental in controlling our app and executing each test.



Config:

```
{
  "appium:deviceName": "adb-R58R14PHWWV-tKrGTf._adb-tls-connect._tcp",
  "platformName": "android",
  "appium:packageName": "com.pinterest",
  "appium:appActivity": ".pinterest",
  "appium:noReset": true
}
```

Appium Inspector integrated with SDK emulator



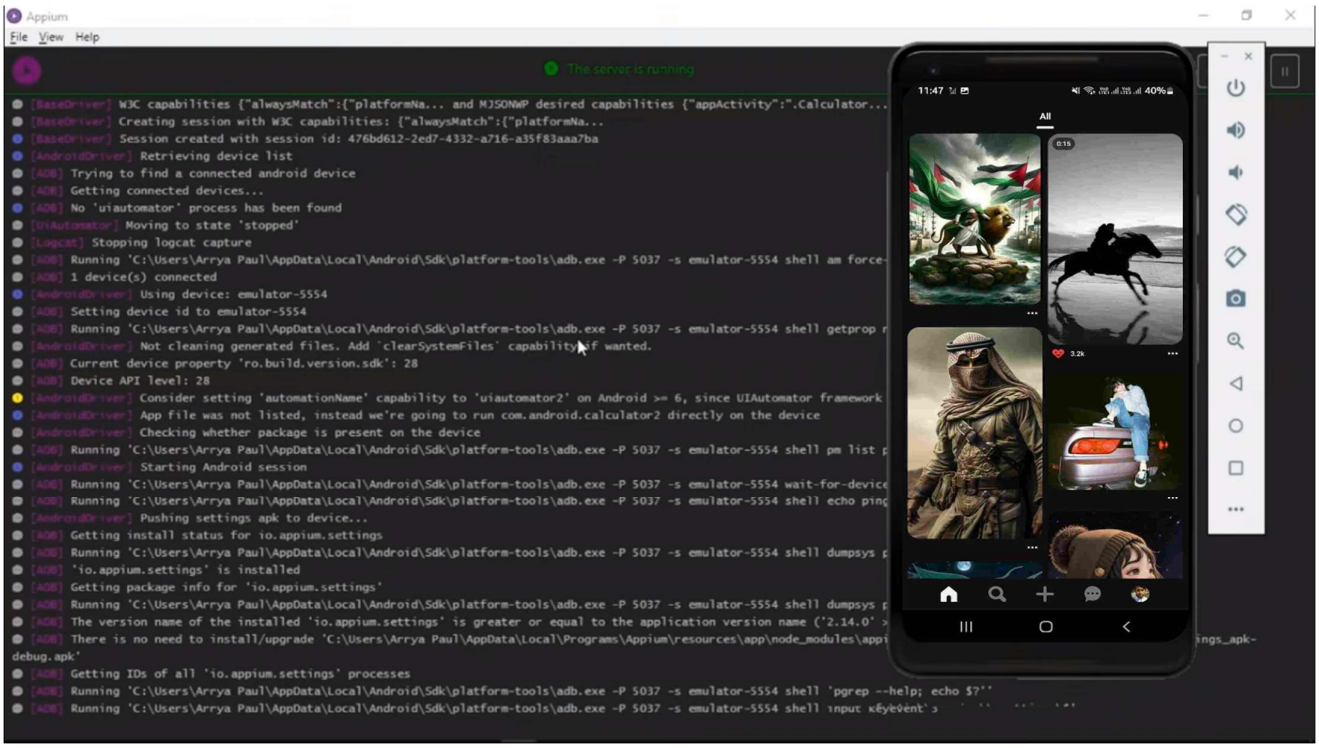
The Android Studio SDK Development Environment was leveraged to avoid the need for installing all dependencies individually, including:

- Node.js (Appium npm package)
- Java Client side
- WebDrivers
- q:io.appium a:java-client
- Lang3 library (Apache)
- Selenium server-side standalone

All the necessary libraries and dependencies were bundled at the creation of a project for **Java.SDK.Automation.**

Test Execution

The Appium inspector executed the test script **Pin-S1** on the server **://4723**.



Test Output:

Appium works in a way that we run the automation script on our IDE **IntelliJ** and it automatically respect the run the appnium server and test the scripts and output the server respoune on our terminal. And it the test fails somehow it, throws a exception on the **IDE.**

The Other Two Scripts

Pin-S2:

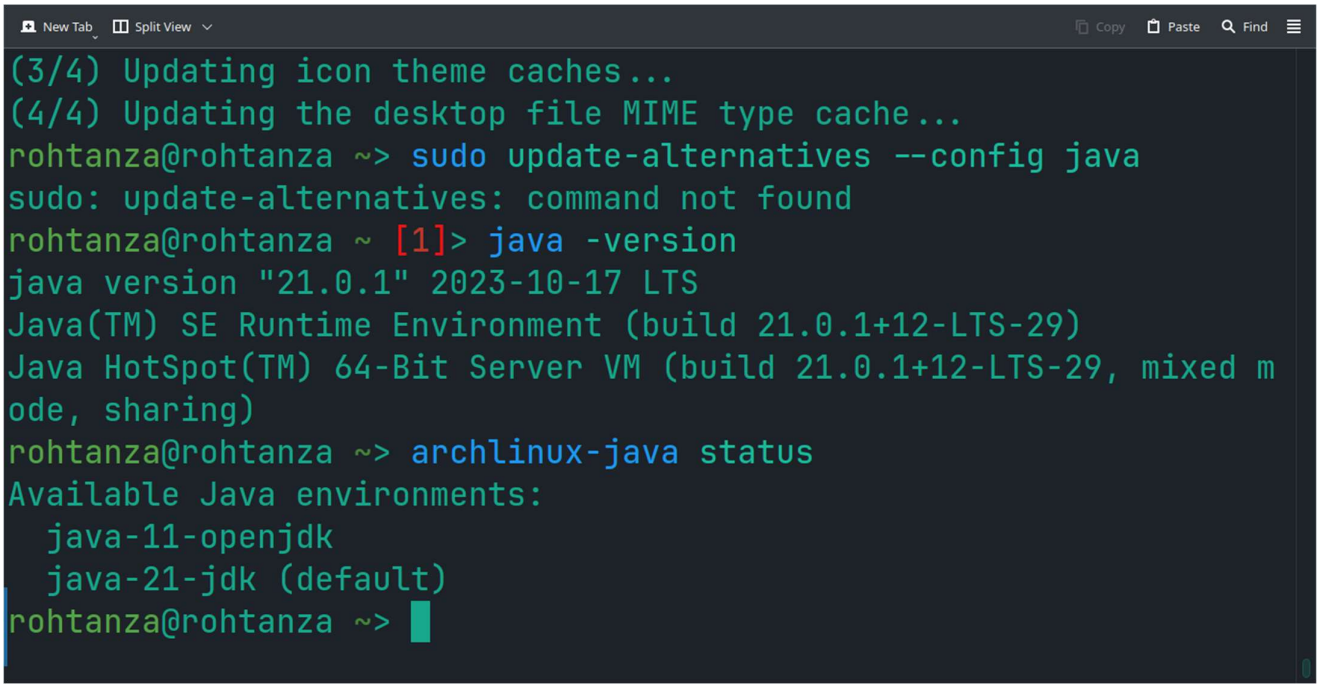
```
import io.appium.java_client.MobileElement;
import io.appium.java_client.android.AndroidDriver;
import io.appium.java_client.remote.MobileCapabilityType;
import io.appium.java_client.remote.AndroidMobileCapabilityType;
import io.appium.java_client.remote.DesiredCapabilities;
import org.openqa.selenium.remote.RemoteWebDriver;
import java.net.MalformedURLException;
import java.net.URL;

public class TestAppium {
    private AndroidDriver<MobileElement> driver;

    public void setUp() throws MalformedURLException {
        DesiredCapabilities desiredCapabilities = new DesiredCapabilities();
        desiredCapabilities.setCapability(MobileCapabilityType.DEVICE_NAME,
"adb-R58R14PHWWV-tKrGTf._adb-tls-connect._tcp");
        desiredCapabilities.setCapability(MobileCapabilityType.PLATFORM_NAME,
"android");

desiredCapabilities.setCapability(AndroidMobileCapabilityType.APP_PACKAGE,
"com.pinterest");

desiredCapabilities.setCapability(AndroidMobileCapabilityType.APP_ACTIVITY,
".pinterest");
        desiredCapabilities.setCapability("appium:noReset", true);

        driver = new AndroidDriver<>(new URL("http://localhost:4723/wd/hub"),
desiredCapabilities);
    }

    public void tearDown() {
        if (driver != null) {
            driver.quit();
        }
    }
}
```

Pin-S3:

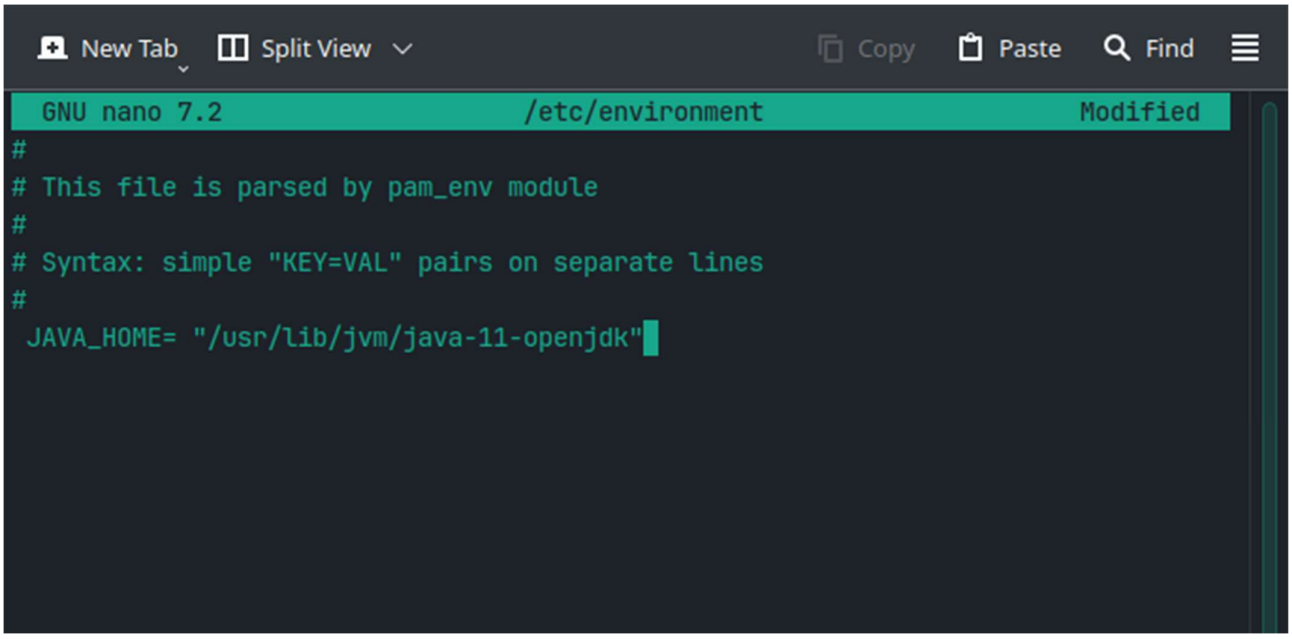
```
import org.junit.Test;
import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;
import java.util.concurrent.TimeUnit;

public class TestAppium {
   @Test
   public void testCustomizeNotificationPreferences() {
      driver.findElement(By.id("notification_preferences_button_id")).click();
      WebElement notificationSwitch =
driver.findElement(By.id("notification_switch_id"));
      notificationSwitch.click();
      WebElement notificationWithPin =
driver.findElement(By.id("notification_with_pin_id"));
      assert notificationWithPin.isDisplayed();
   }

   @Test
   public void testCreateBoardForUser() {
      driver.findElement(By.id("create_board_button_id")).click();
      WebElement boardNameInput =
driver.findElement(By.id("board_name_input_id"));
      boardNameInput.sendKeys("Test Board");
      driver.findElement(By.id("save_board_button_id")).click();
      WebElement boardName = driver.findElement(By.id("board_name_id"));
      assert boardName.getText().equals("Test Board");
   }
}
```

3. Document Any UI Issues Identified During Testing:

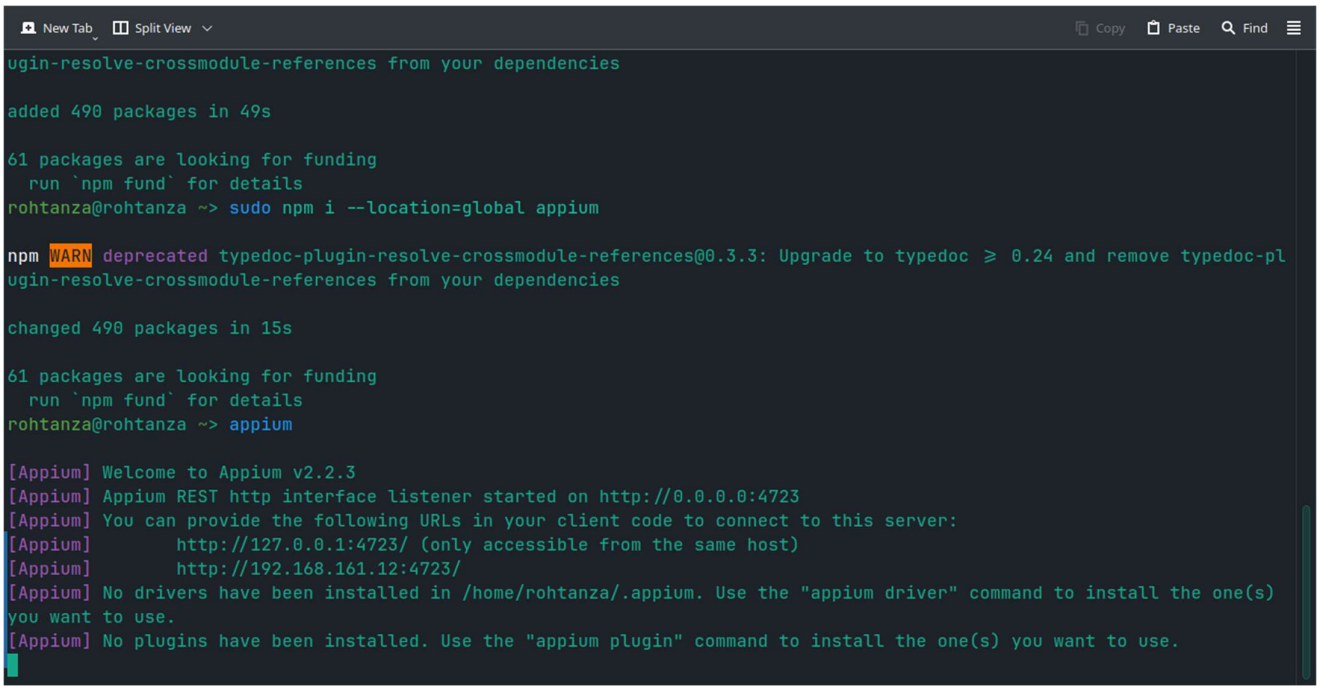
The initial challenge we encountered involved configuring the Java Development Kit (JDK). Our choice of using Manjaro proved problematic due to the complexities associated with installing software on this platform. Once the JDK was installed, we encountered difficulties in configuring the Java environment variable, essential for executing automation processes.



Upon successfully installing all the requisite components, we managed to launch the Appium server. Additionally, we installed an older version from the previous year, which functioned flawlessly for our purposes.



Upon completing the installation of all essential components, we successfully initiated the Appium server. Subsequently, we installed an older version from a year prior, which operated flawlessly.



4. Recommendations for Resolving UI Issues

To address and resolve the UI issues you mentioned, here are some recommendations:

1. **System Status Indicator or Icon Validation:**

To validate whether the system status indicator or icon is correctly displayed when a pin has been saved, you should:

- Test the indicator or icon's visibility and functionality after saving a pin. This can be done by saving a pin and checking if the indicator or icon changes as expected.
- Ensure that the indicator or icon provides feedback to the user. For instance, if the indicator or icon is supposed to turn green when a pin is saved, it should indeed turn green. This feedback helps the user understand that their action was successful.
- Test if the page does not automatically refresh when returning to the top. This can be done by saving a pin, navigating away from the page, and then returning to the top to check if the page refreshes. If it does, you'll need to modify the code to prevent this.

2. **Notification Preferences Customization:**

To test the functionality of customizing notification preferences and whether the notifications are displayed with the pins, you should:

- Allow users to customize their notification preferences. This can be done by providing options to users to choose the type of notifications they want to receive (e.g., email, push notifications, in-app notifications).
- Test the notifications with the pins. This can be done by saving a pin and checking if the notification is displayed as expected. The notification should include the details of the pin and any other relevant information.

3. **User Board Creation:**

To test the feature that allows each user to create a board for them, you should:

- Allow users to create a board. This can be done by providing a "Create Board" button or link that, when clicked, opens a form where users can input the details of their board.

Sir, in our personal opinion, Pinterest's popularity is declining steadily due to competitors like TikTok. To revitalize the app, making it open-source could breathe new life into it.