****

1. **Introduction**  
   This report provides an overview of a test automation framework and the test cases run for the LevelSuperMind mobile application. The automation is done using Appium with Java for running and reporting tests, and the framework uses TestNG.  
     
   **2. Tools and Technologies Used**  
   Appium - Mobile Automation.  
     
   Java- The programming language used to write test scripts.  
     
   TestNG- The test framework used for test case management and execution.  
     
   AndroidDriver- Interacts with the Android elements.  
     
   Maven- If any dependency management.  
     
   IDE- IntelliJ IDEA or Eclipse(for development).  
     
   **3. Test Automation Framework**The framework has been created in order to automate the functional areas of the LevelSuperMind mobile application.  
     
   Capabilities Class- This manages settings like device and app.  
     
   Test Cases- Stored into methods using TestNG annotations.  
     
   **4. Test Cases**Below is a detailed overview of the test cases executed:  
     
   Test Case 1: Open the App  
   Description: Verifying that the LevelSuperMind app opens without any issue.  
     
   Steps:  
     
   Open the app.  
     
   Wait for the app to start.  
     
   Expected Result: It should open the app successfully.  
     
   Test Case 2: Click on Sign-In Button  
   Description: Check if the user can click on the Sign-In button.  
     
   **Steps:**  
   Identify the Sign-In button.  
     
   Click on the Sign-In button.  
     
   **Expected Result:** The Sign-In screen should appear.  
     
   Test Case 3: Enter Phone Number  
   Description: Verify that the user can enter a phone number.  
     
   Steps:  
     
   Get the element for the phone number input field.  
     
   Input a valid phone number (6267021603).  
     
   **Expected Result**: Successfully identified phone number input.  
     
   Test Case 4: Click on Arrow Button  
   Description: Check the ability of the user to click on the arrow button in order to proceed.  
     
   **Steps:**  
     
   Find the arrow button.  
     
   Click on the arrow button.  
     
   **Expected Results**: The app should navigate to the next screen.  
     
   Test Case 5: Click on Arrow Button Again  
   Description: Check if the user is able to click again to navigate along their path.  
     
   **Steps**:  
     
   Find the arrow button.  
     
   Click on the arrow button.  
     
   **Expected Results:** The app should go to the next screen.  
     
   Test Case 6: Click on Options  
   Description: This checks that a user can click on the options menu.  
     
   **Steps:**  
   Identify the options menu.  
     
   Click on the options menu.  
     
   **Expected Results**- The options menu should be opened.  
     
   Test Case 7: Click on Logout Button  
   Description: Here we check for Logout button interaction by the user.  
     
   **Steps**:  
     
   Find and scroll the logout button.  
     
   Click on the Logout button.  
    **Expected Result:** Logout confirmation screen should appear.  
     
   Test Case 8: Confirm Logout  
   Description: Check logout confirmation from the user   
     
   **Steps**:  
     
   Find Logout confirmation button.  
     
   Click on Logout confirmation button.  
     
   **Expected Result**: Should log off to login screen.  
     
   5. Code Structure  
   The code is organized into several parts:   
     
   5.1. Capabilities Class  
     
   Purpose: Manage the capability set up for the device and app.  
     
   Main Methods:  
     
   level\_superMind(): Sets up the driver for Android with all required capabilities.   
     
   5.2. Test Class (RunLevelSuperMind)  
     
   Function: Contains all the test cases.  
     
   Key Annotations:  
     
   @BeforeTest: Responsible for app and driver initialization.  
     
   @Test: Assigns priorities to individual test cases.  
     
   Major Methods:  
     
   setup(): Drives the setup and implicit waits.  
     
   Test methods (run(), start(), phoneNumber(), etc.): Execute the test steps.  
   **7. Challenges and Solutions**  
   Challenge 1: Dynamic elements with different XPaths.  
     
   Solution: Using relative XPaths or accessibility IDs.  
     
   Challenge 2: Some screens are slow to load.  
     
   Solution: Instead of using Thread.sleep(), use explicit waits(WebDriverWait).  
    **8. Future Enhancements**  
   More test cases would be included for other features, like profile updates, settings.   
     
   External data sources- Excel, JSON will be implemented for data-driven testing.   
     
   CI/CD pipeline integration for automated testing.   
     
   **9. Conclusion**  
   The test automation framework for LevelSuperMind App will assist in validating the application's functioning on assumptions. Appium, Java, and TestNG are an excellent, scalable, and strong combination for mobile automation testing.  
     
   **10. Appendix**  
   **Dependencies:   
     
   Appium Java Client  
     
   TestNG  
     
   Selenium WebDriver**