

INDEX

No.	Definition / Aim	CO	Date	Signature
1	Install and configure Python, Android Studio, Android SDK, Android Emulator, Appium Server, Appium Inspector and required Python packages for mobile automation.	CO1		
2	Create an Android Virtual Device (AVD), start the emulator, configure developer options, and prepare it for automated testing.	CO1		
3	Install a native Android application and prepare the testing environment for native, hybrid, and mobile web application testing.	CO1		
4	Install an APK on an Android emulator and perform basic functional testing such as launching, navigation, input, and closing the application.	CO1		
5	Design practical test cases and test data for a sample Android application covering functional, usability, compatibility, and interruption scenarios.	CO1		
6	Install Appium and the Python Appium client, configure the Android automation environment, and verify that Appium can communicate with the Android device/emulator.	CO2		
7	Create a Python program that establishes an Appium session with an Android emulator using appropriate desired capabilities/options.	CO2		
8	Use Appium Inspector to inspect an Android application and identify element properties such as resource-id, text, class name, accessibility ID, and XPath.	CO2		
9	Develop a Python-Appium test script to locate username and password fields, enter test data, click the Login button, and verify the result.	CO2		
10	Develop an Appium automation script to perform multiple navigation actions and verify expected text, buttons, screens, and application states.	CO2		
11	Develop a Python-Appium test script to launch the Android Calculator and automate arithmetic operations such as addition, subtraction, multiplication, and division.	CO3		
12	Develop an Appium Python script to locate text fields and buttons, enter values, perform button clicks, clear fields, and verify application output.	CO3		

Ganpat University - A M Patel Institute of Computer Studies

No.	Definition / Aim	CO	Date	Signature
13	Create a Python-Appium automation script to interact with checkboxes, radio buttons, switches, and other selectable mobile UI controls and verify their states.	CO3		
14	Automate an Android application using ID, Accessibility ID, Class Name, XPath, and other suitable locators and compare their practical usage.	CO3		
15	Develop a robust Python-Appium test by handling waits, element-not-found errors, session errors, application launch problems, and other common automation issues.	CO3		
16	Capture an Android application screen using UiAutomatorViewer and identify UI objects, resource IDs, text, class names, package names, and element bounds.	CO4		
17	Configure ADB and execute practical commands to identify devices, install/uninstall APKs, launch applications, capture logs, and perform basic device operations.	CO4		
18	Enable USB debugging and connect a real Android device to the computer using ADB, then verify the device and perform basic testing operations.	CO4		
19	Develop a Python-based automation workflow that uses ADB and Appium to prepare the device, launch an application, execute test steps, and collect test results.	CO4		
20	Develop and execute a complete Python-Appium test suite covering application launch, element identification, user input, navigation, validation, error handling, and test result reporting.	CO4		

Pra-01	Install and configure Python, Android Studio, Android SDK, Android Emulator, Appium Server, Appium Inspector and required Python packages for mobile automation.
Code Files	
Input	
Output	
Screenshot	