Design Document for Commands Test Automation Tool

Version 1.0

Prepared by: Mr. Rahul J. Shimpi

* **Introduction:**

The OTA (Over-The-Air) Commands Test Automation Tool is a graphical user interface (GUI) application developed to automate the testing of OTA commands sent to devices. This tool simplifies the process of executing commands stored in an Excel file against a device with a specific IMEI (International Mobile Equipment Identity) number. It captures the execution results, generates test logs, and provides a summary of test execution.

* **Purpose:**

The purpose of this tool is to streamline the testing process for OTA commands by providing a user-friendly interface for executing commands, tracking results, and generating reports. It aims to improve efficiency, accuracy, and consistency and save time in OTA command testing.

* **Features:**

•Browse and select an Excel file containing the list of commands to be executed.

•Validate user inputs, including the Excel file path, IMEI, and tester name.

•Execute OTA commands sequentially against the device.

•Display real-time progress of command execution using a progress bar.

•Capture command execution results, including expected and actual responses.

•Log command execution details to a text file for reference.

•Generate an Excel report summarizing the test execution, including pass/fail status and relevant metrics.

•Display a pie chart summarizing test results for quick analysis.

•Provide options to abort execution, reset inputs, and open the output folder.

•Support for handling GUI responsiveness during execution using threading.

* **Tools and Technologies:**

•Programming Language: Python

•GUI Toolkit: Tkinter

•External Libraries: pandas, openpyxl, matplotlib, PIL (Python Imaging Library)

•External Tools: Java (for executing OTA commands via JAR file)

* **Workflow:**

1. Launch Application: Upon launching the application, the user is presented with a GUI interface.

2. Input Data: The user browses and selects an Excel file containing the list of OTA commands, enters the device IMEI, and provides their name as the tester.

3. Execute Commands: When you click the "Execute Commands" button, the application validates inputs, initiates command execution, and displays real-time progress.

4. Command Execution: The application sequentially executes each command against the device, captures results, and updates the progress bar.

5. Result Analysis: During execution, the application logs details, compares expected and actual responses, and updates the summary statistics.

6. Report Generation: Upon completion, the application generates an Excel report summarizing test execution and displays a pie chart illustrating test results.

7. User Interaction: Users can abort execution, reset inputs, or open the output folder for further analysis.

* **File Structure:**

•Main Script: ota\_commands\_test.py

•Sample Commands Excel File: sample\_commands.xlsx

•Logo Image: logo.png

•Java Executable JAR: ota-cmdutil-0.0.1-SNAPSHOT.jar

•Generated Output Files: Timestamped Excel report and test logs

* **Deployment:**

•The application can be packaged into a standalone executable using PyInstaller for easy distribution.

•Deployment on Windows, macOS, and Linux platforms is supported.

•Ensure that external files (Excel, image, JAR) are included in the distribution package.

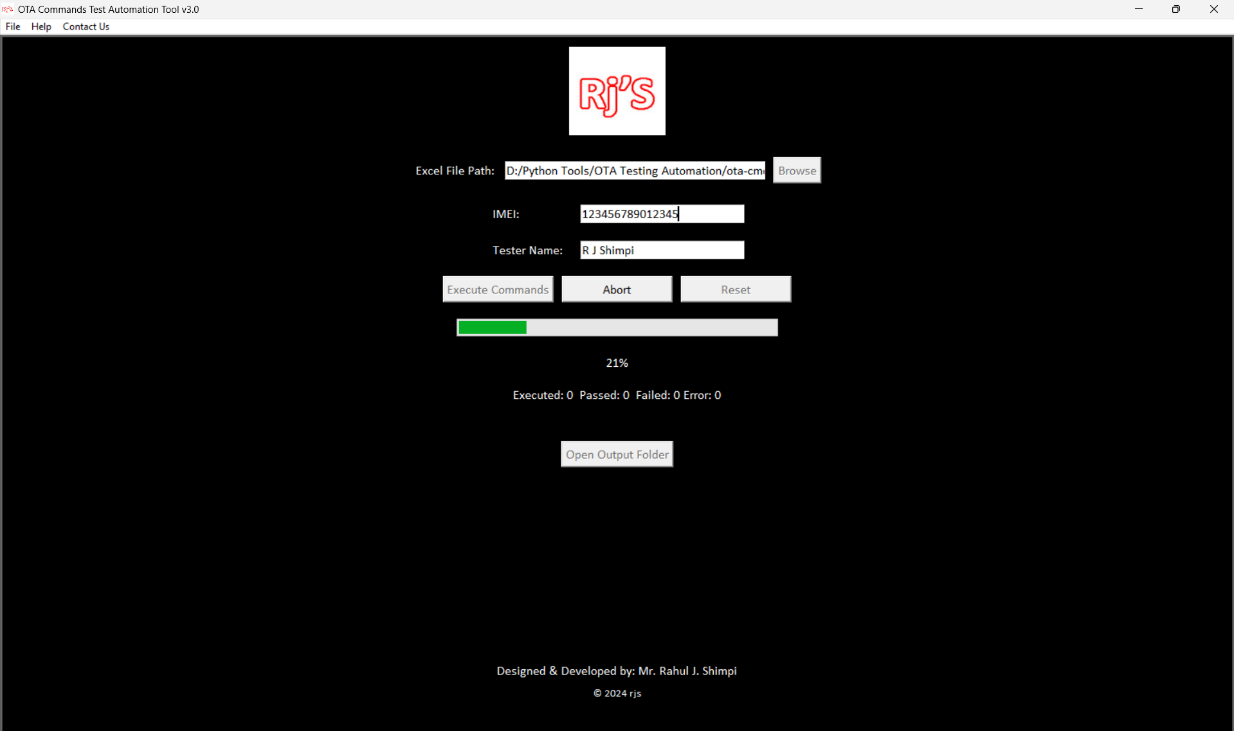
* **Conclusion:**

The OTA Commands Test Automation Tool simplifies and accelerates the testing of OTA commands, offering a robust solution for device testing teams. With its intuitive interface and comprehensive features, it streamlines the testing process, improves productivity, and ensures the quality of OTA command implementations.

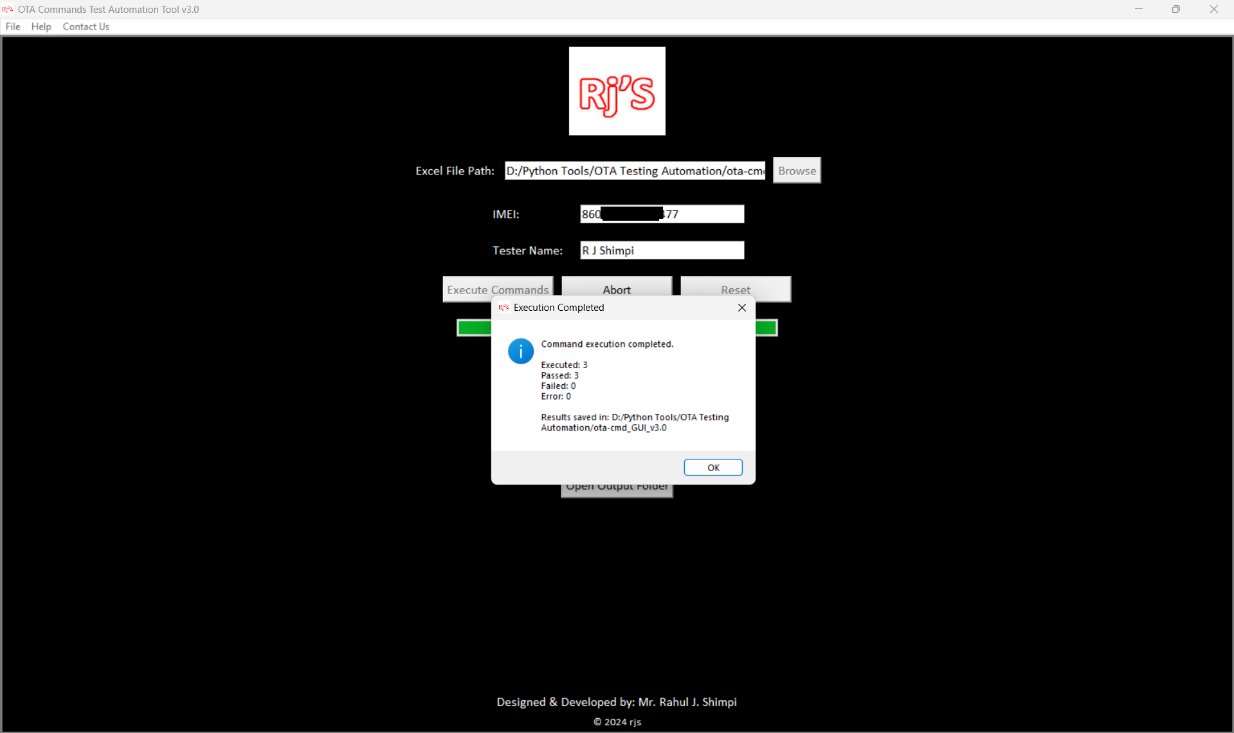
* **Screenshots:**
  + First screen:



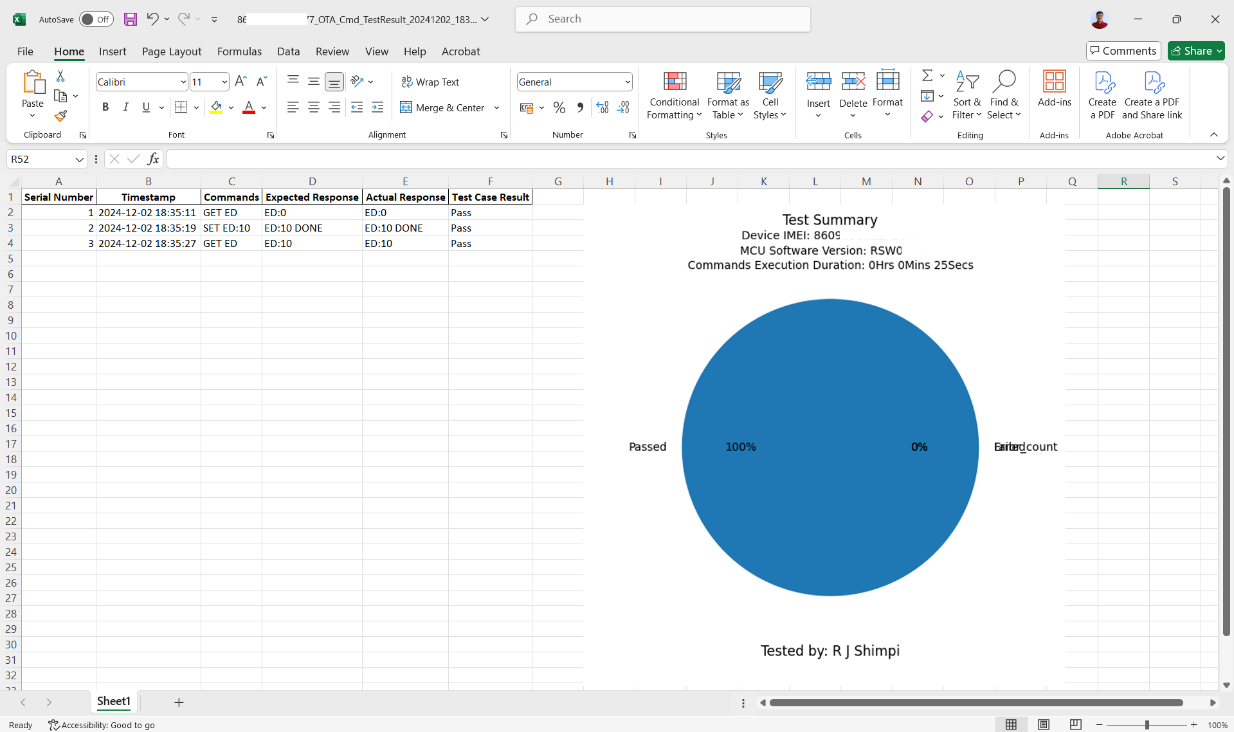
* + In progress:



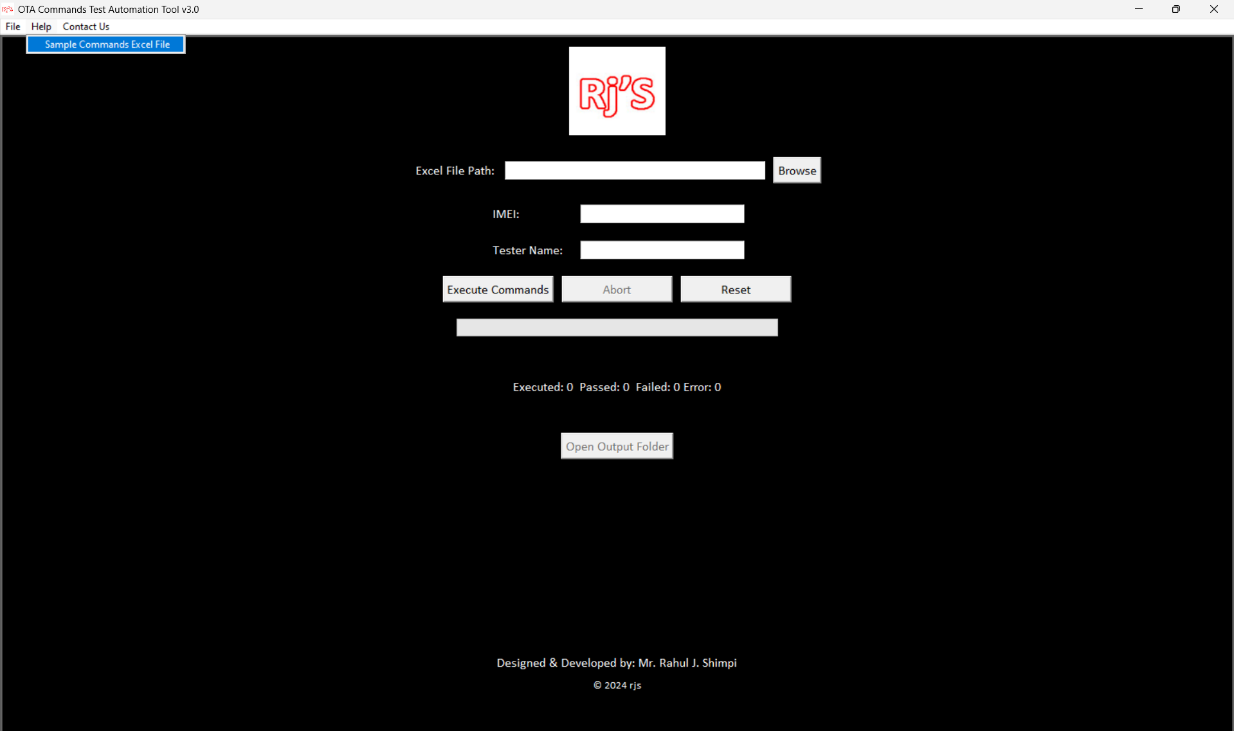
* + Completed:

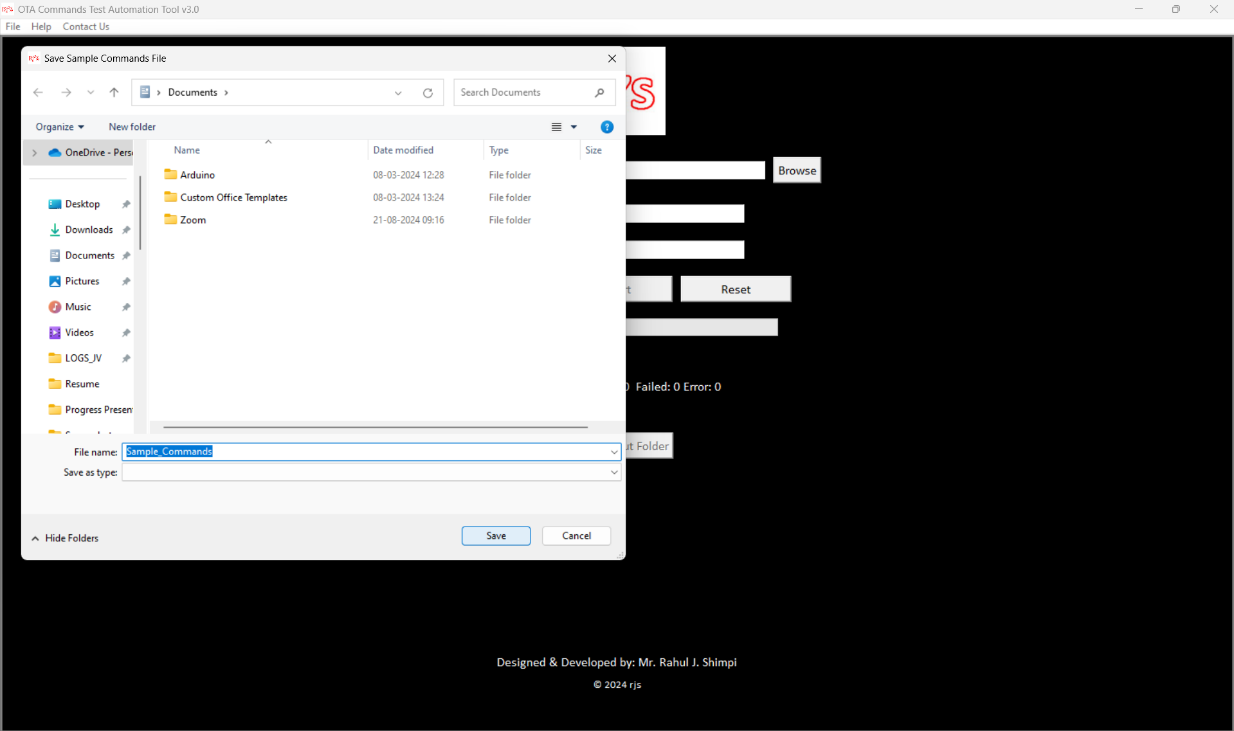


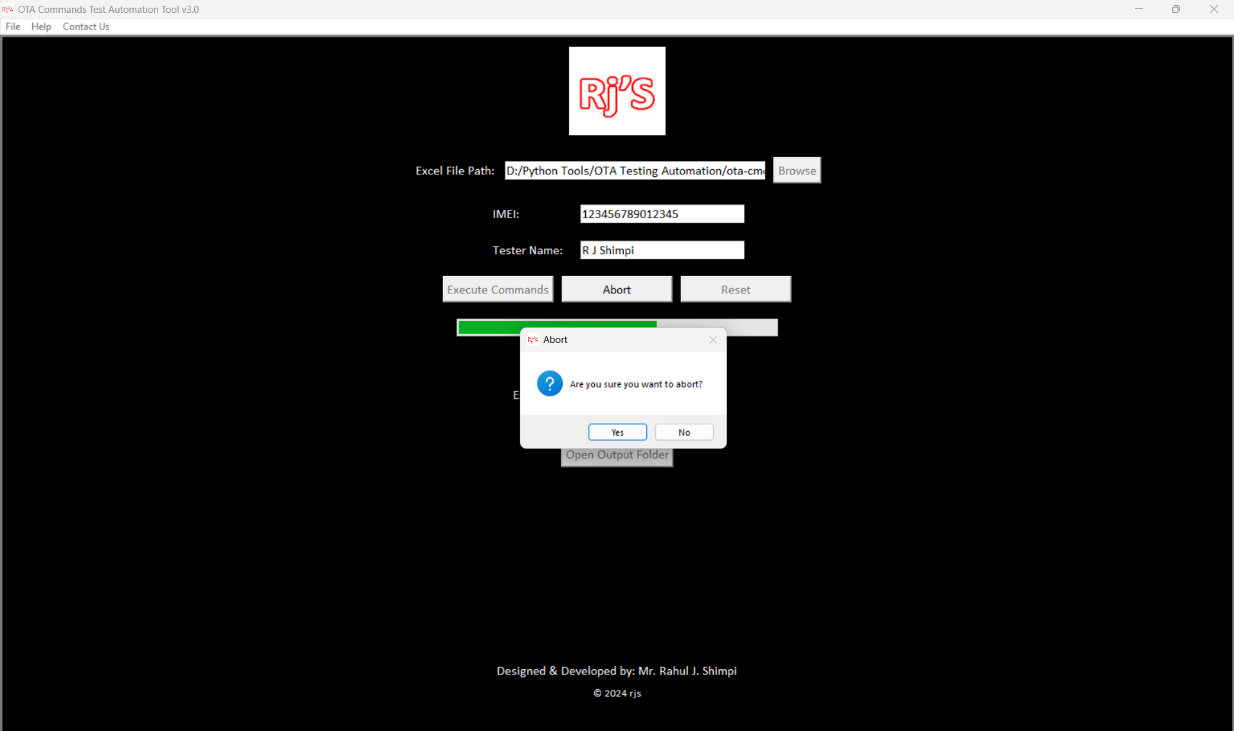
* + Result Analysis:



* + Help Tab:







* **Executable File:** 
  + Download ota-cmd\_GUI\_v1.0.exe file

Q & A