INDEX

1. Introduction

1.1 Background Study: In this ever-evolving digital landscape, mobile devices have become an integral part of our daily lives. With their extensive capabilities and wide-ranging functionalities, smartphones have become indispensable tools for communication, productivity, and entertainment. However, the increasing reliance on mobile technology also brings forth potential security risks and vulnerabilities. Introducing an innovative mobile security tool designed to address these very concerns. It is a cutting-edge application that aims to empower individuals and organizations with the ability to proactively assess and enhance the security of their mobile devices.

Driven by the ethos of responsible and ethical cybersecurity practices, This tool offers a comprehensive suite of features, providing users with insights into potential vulnerabilities, weak points, and security loopholes that malicious actors could exploit. The tool facilitates penetration testing, allowing users to simulate real-world attacks and identify areas for improvement in their mobile security infrastructure.

Mobile devices, particularly smartphones, have become indispensable in modern society, leading to an increased need for robust security measures to protect sensitive data. However, the rising number of cyber threats and vulnerabilities poses significant challenges to mobile security. Ethical hacking tools have emerged as proactive solutions to identify and mitigate potential security risks. Existing research highlights the importance of responsible ethical hacking practices and legal compliance while conducting security assessments on mobile devices. While several ethical hacking tools are available, this research focuses on the specific capabilities and impact of this tool in enhancing mobile security.

1.2 Aim: The aim of this project is to enhance mobile device security through the use of ethical hacking tools, with a focus on investigating the effectiveness, ethical considerations, and applicability of a specific tool. This project seeks to contribute to the understanding of how this tools can identify vulnerabilities in mobile devices and applications, prioritize them based on severity, and provide valuable insights for improving overall mobile security.

This aim statement encapsulates the core objective of the project, which is to improve mobile device security through ethical hacking practices while emphasizing the importance of data protection and user privacy.

2. Literature Review

2.1 Penetration Testing for Mobile Cloud Computing Applications.

Penetration testing of mobile applications has become more complex and expensive due to several parameters, such as the platform, device heterogeneity, context event types, and offloading. Numerous studies have been published in the MCC domain, whereas few studies have addressed the common issues and challenges of MCC testing. However, current studies do not address MCC and penetration testing. Therefore, revisiting MCC and penetration testing domains is essential to overcoming the inherent complexity and reducing costs. Motivated by the importance of revisiting these domains, this paper pursues two objectives: to provide a comprehensive systematic literature review (SLR) of the MCC, security and penetration testing domains and to establish the requirements for penetration testing of MCC applications.

2.2 Penetration Testing for Android Smartphones.

One major challenge faced by Android users today is the security of the operating system especially during setup. The use of smartphones for communication, social networking, mobile banking and payment systems has all tripled and many have depended on it for their daily transactions. Android OS on smartphones is so popular today that it has beaten the most popular mobile operating systems, like RIM, iOS, Windows Mobile and even Symbian, which ruled the mobile market for more than a decade. This paper performs penetration testing of Android-based Smartphones using an application program designed to simplify port-scanning techniques for information gathering and vulnerability attack.

2.3 A Review on Web Application Vulnerability Assessment and Penetration Testing.

With the increase in the number of internet users, web applications, user data there is an increase in the number of hackers all over the world. It is becoming challenging for organizations to ensure the security of the data of their employees and their customers around the world. Any cyber-attack on the organization will drastically affect the reputation of the organization as well as the loss of trust from the users or customers. Customers will not invest in these organizations who have encountered a cyber threat or attack. Hence, enabling regular security testing and checks by the penetration testers or security analysts help in preparing the organization from any security threat by testing network and applications.

2.4 An Over view of Penetration Testing.

Penetration testing is an effort to attack a system using similar techniques and tools adopted by real hackers. The goal of penetration testing is to call to light as many existing vulnerabilities as possible, then come up with practical solutions to remediate the problems; thus, enhance the system security. The paper introduces concepts and definitions related to penetration testing, together with different models and methodologies to conduct a penetration test. A wide range of penetration testing state-of-the-art, as well as related tools both commercial and free open source available on the market are also presented in relatively rich details.

2.5 Penetration Testing and Vulnerability Assessments: A Professional Approach.

Attacks against computer systems and the data contained within these systems are becoming increasingly frequent and evermore sophisticated. So-called “zero-day” exploits can be purchased on black markets and Advanced Persistent Threats (APTs) can lead to exfiltration of data over extended periods. Organizations wishing to ensure security of their systems may look towards adopting appropriate measures to protect themselves against potential security breaches. One such measure is to hire the services of penetration testers (or “pen-tester”) to find vulnerabilities present in the organization’s network, and provide recommendations as to how best to mitigate such risks. This paper discusses the definition and role of the modern pen-tester and summarizes current standards and professional qualifications in the UK. The paper further identifies issues arising from pen-testers, highlighting differences from what is generally expected of their role in industry to what is demanded by professional qualifications.

2.6 About Penetration Testing.

Students generally learn red teaming, sometimes called penetration testing or ethical hacking, as "breaking into your own system to see how hard it is to do so". Contrary to this simplistic view, a penetration test requires a detailed analysis of the threats and potential attackers in order to be most valuable. Using the results of penetration testing requires proper interpretation. Neither testers nor sponsors should assert that the penetration test has found all possible flaws, or that the failure to find flaws means that the system is secure. All types of testing can show only the presence of flaws and never the absence of them. The best that testers can say is that the specific flaws they looked for and failed to find are not present: this can give some idea of the overall security of the system's design and implementation.

3. Objectives and Scope

3.1 Objectives:

1. **Security Assessment:**
   - **Identify and assess vulnerabilities:** Use tools and methodologies to identify potential weaknesses in systems, networks, and applications.
   - **Evaluate security measures:** Assess the effectiveness of existing security controls and mechanisms.

2. **Penetration Testing:**
   - **Simulate real-world attacks:** Conduct controlled and authorized attempts to exploit vulnerabilities and assess the system's resistance to attacks.
   - **Test resilience:** Determine how well a system withstands various forms of penetration and identify areas for improvement.

3. **Policy Compliance:**
   - **Assess security policies:** Review and analyze existing security policies against industry standards and regulatory requirements.
   - **Evaluate compliance:** Ensure that the organization is adhering to legal and regulatory frameworks related to information security.

4. **Social Engineering Testing:**
   - **Assessment:** Simulate social engineering attacks to test the vulnerability of employees.
   - **Awareness programs:** Provide feedback and suggestions for improving security awareness programs.

5. **Network Security:**
   - **Assess infrastructure:** Evaluate the security of routers, switches, and other network components.
   - **Test security controls:** Check the effectiveness of firewalls, intrusion detection and prevention systems, and other security measures.

3.2 Scope:

- **IoT Security Testing:** As the Internet of Things (IoT) continues to expand, the need for testing the security of connected devices and ecosystems will rise. Penetration testing software will need to adapt to assess the security of smart devices, home automation systems, industrial IoT, and other interconnected technologies.

- **Cloud Security Assessments:** With the increasing adoption of cloud computing, penetration testing will focus more on assessing the security of cloud-based infrastructures, platforms, and services. Future penetration testing software should be equipped to evaluate configurations, identity and access management, and data protection in cloud environments.

- **AI and Machine Learning Security:** As artificial intelligence (AI) and machine learning (ML) technologies become more prevalent, penetration testing software will need to address the unique security challenges associated with these domains. This includes testing for vulnerabilities in AI algorithms, models, and systems.

- **Automated Threat Simulation:** Future penetration testing software is likely to leverage advanced automation and AI-driven technologies to simulate sophisticated cyber threats. Automated threat simulations can provide more realistic and dynamic assessments of an organization's security posture.

- **DevSecOps Integration:** Integration with DevSecOps practices will become more crucial, ensuring that security is integrated into the software development lifecycle from the beginning. Penetration testing software will need to seamlessly integrate with continuous integration/continuous deployment (CI/CD) pipelines and provide rapid feedback to development teams.

- **Blockchain Security Assessments:** As blockchain technology is increasingly adopted in various industries, penetration testing will need to address the security of blockchain networks, smart contracts, and decentralized applications. Assessing the vulnerabilities unique sto blockchain ecosystems will be essential.

3.3 Applicability:

- **Organizational Security:** This tool can be applied within organizations to conduct thorough security assessments on employees' mobile devices used for work-related purposes. This ensures compliance with security policies, identifies potential vulnerabilities, and helps prevent unauthorized access to sensitive corporate data.

- **Mobile App Development:** App developers can employ this tool to evaluate the security of their applications during the development lifecycle. By simulating potential attack scenarios, developers can identify and address security weaknesses before releasing their apps to the public, thereby building more secure products.

- **Personal Device Security:** Individual users can benefit from this tool by assessing their personal mobile devices for vulnerabilities. This empowers users to take proactive measures to protect their data, detect potential threats, and maintain a higher level of personal privacy.

- **Device Manufacturers:** Phone manufacturers and mobile device makers can incorporate this tool in their quality assurance processes. By conducting security assessments before launching products to the market, manufacturers can enhance the security of their devices and build consumer trust.

- **Mobile Service Providers:** Telecommunications companies and mobile service providers can employ the project's best practices to enhance the security of their networks and services, thereby providing a more secure environment for their customers.

- **Security Consulting Firms:** Security consulting firms can integrate the project's tools and methodologies into their service offerings, providing clients with comprehensive mobile security assessments.

- **Legal and Regulatory Bodies:** Government bodies responsible for cybersecurity and data protection can use the project's outcomes to develop or refine legal and regulatory frameworks related to mobile device security assessments.

- **Ethical Hacking Community:** The ethical hacking community can benefit from the project's insights and collaborate to further improve mobile security practices.

4. Technology Survey

1. **Python:** Python is an OOPs (Object Oriented Programming) based, high level, interpreted programming language. It is a robust, highly useful language focused on rapid application development (RAD). Python helps in easy writing and execution of codes. Python can implement the same logic with as much as 1/5th code as compared to other OOPs languages.

   Python provides a huge list of benefits to all. The usage of Python is such that it cannot be limited to only one activity. Its growing popularity has allowed it to enter into some of the most popular and complex processes like Artificial Intelligence (AI), Machine Learning (ML), natural language processing, data science etc. Python has a lot of libraries for every need of this project. For JIA, libraries used are speech recognition to recognize voice, Pyttsx for text to speech, selenium for web automation etc.

2. **Android Debug Bridge (ADB):** ADB stands for Android Debug Bridge, a powerful command line tool that you can use to debug your Android phone or tablet and send a large number of commands to control behavior on the device, allowing for the installation of apps and the logging of processes. To achieve that, the tool is composed of three distinct parts.

   First, there is the client interface that lives on the machine you use for developing or debugging that sends commands to your device or emulator through the command line terminal. Second, there is a daemon (ADBD) that actually executes the commands you send using the client. It runs in the background on all devices and emulators equipped with ADB. Lastly, there is a server on your development machine that establishes the connection to your device or emulator.

   The Android Debug Bridge is included as part of Android Studio, Google's IDE (Integrated Development Environment) for Android app development, but you can also download it as a standalone tool if you don't need the full IDE. (The full IDE installation is very large.)

3. **Scrcpy:** (pronounced "**screen copy** ") is a free, open-source, and cross-platform application used to display and control an Android device from your Linux desktop computer. It works on Linux, Windows, and macOS, and allows you to control a device connected via a USB or wirelessly (over TCP/IP). It features mirroring with Android device screen off, configurable screen display quality, recording, copy and paste in both directions, using an Android device as a webcam (Linux only), physical keyboard and mouse simulation, OTG mode, and much more.

5. System and Other Requirements

5.1 Requirement Definition:

5.1.1 Functional Requirements:

- **Vulnerability Identification**: This tool should be capable of scanning mobile devices, operating systems, and applications to identify potential vulnerabilities.
- **Vulnerability Assessment:** The tool should assess the severity and potential impact of identified vulnerabilities to prioritize them for mitigation.
- **Penetration Testing:** IT should simulate real-world attack scenarios to test the device's defenses and identify possible points of exploitation.
- **Report Generation:** The tool should generate detailed reports that include information about identified vulnerabilities, their severity, and recommendations for mitigation.

5.1.2 Non-Functional Requirements:

- **Performance**: This tool should efficiently scan and assess vulnerabilities without causing significant performance degradation on the tested devices.
- **Security:** The tool should prioritize user data security and confidentiality during vulnerability assessments, ensuring that no sensitive information is compromised.
- **Compatibility:** This tool should be compatible with a wide range of mobile devices, operating systems, and applications.
- **Accuracy:** The tool should accurately identify vulnerabilities and assess their severity to provide reliable results.
- **Usability:** This tool should have clear and well-documented functionalities, enabling users to navigate and utilize its features effectively.

5.2 Problem Definition: The problem at hand is the need to fortify mobile device security effectively while adhering to ethical principles and legal regulations. The challenge encompasses several dimensions:

- **Identifying Vulnerabilities:** The dynamic nature of mobile technology means that new vulnerabilities continuously emerge. There is a need for these tools to effectively identify and catalog these vulnerabilities to maintain a strong security posture.

- **Assessing Severity:** Once vulnerabilities are identified, organizations need to assess their severity and potential impact accurately. This is crucial for prioritizing mitigation efforts and allocating resources efficiently.

- **Ethical Use:** Ethical hacking tools like this must be employed responsibly and ethically. Organizations must ensure that security assessments are conducted with explicit consent and adhere to ethical hacking guidelines.

- **Legal Compliance:** Compliance with relevant legal frameworks is non-negotiable. Organizations need to navigate complex legal requirements to conduct mobile security assessments within the bounds of the law.

5.3 Hardware Requirements:

- Windows Operating system.
- Minimum 4gb ram.
- Intel i3 processor or above.
- Minimum 20gb memory.
- USB cable.

5.4 Software Requirements:

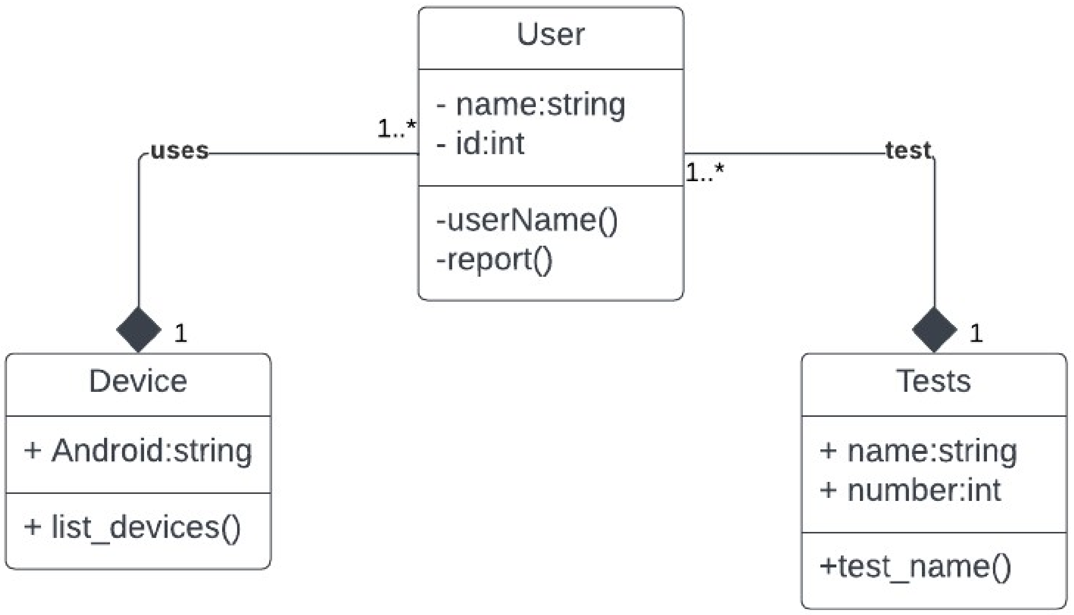
- Python3 : Python 3.10 or Newer
- ADB: Android Debug Bridge (ADB) from Android SDK Platform Tools
- Scrcpy

6. System Design

6.1 Class Diagram:

The class diagram depicts a static view of an application. It represents the types of objects residing in the system and the relationships between them. A class consists of its objects, and it may inherit from other classes.

A class diagram is used to visualize, describe, document various aspects of the system, and also construct executable software code. It shows the attributes, classes, functions, and relationships to give an overview of the software system. It constitutes class names, attributes, and functions in a separate compartment that helps in software development. Since it is a collection of classes, interfaces, associations, collaborations, and constraints, it is termed as a structural diagram.



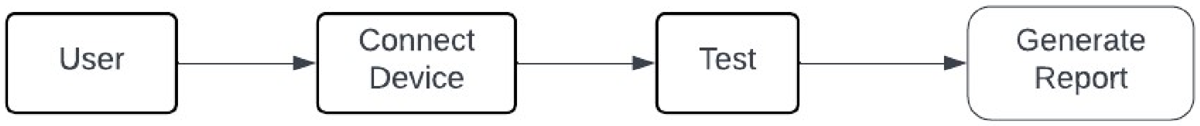
The above diagram shows three classes device, user, and tests. Tests that can be performed on the device by the user. The Device class has android attribute that specifies the OS of the device, the user has name and id attribute that defines the user whereas tests have name and number attributes that specifies the types of different tests available.

6.2 Data Flow Diagram:

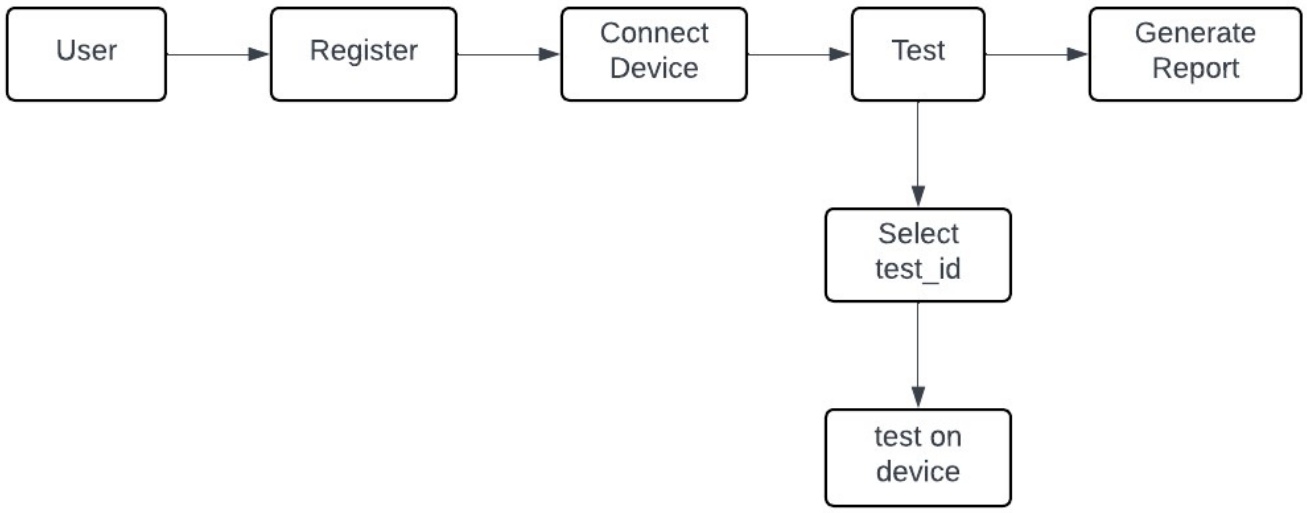
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.

The objective of a DFD is to show the scope and boundaries of a system. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called as a data flow graph or bubble chart.

- DFD Level 0 Diagram:



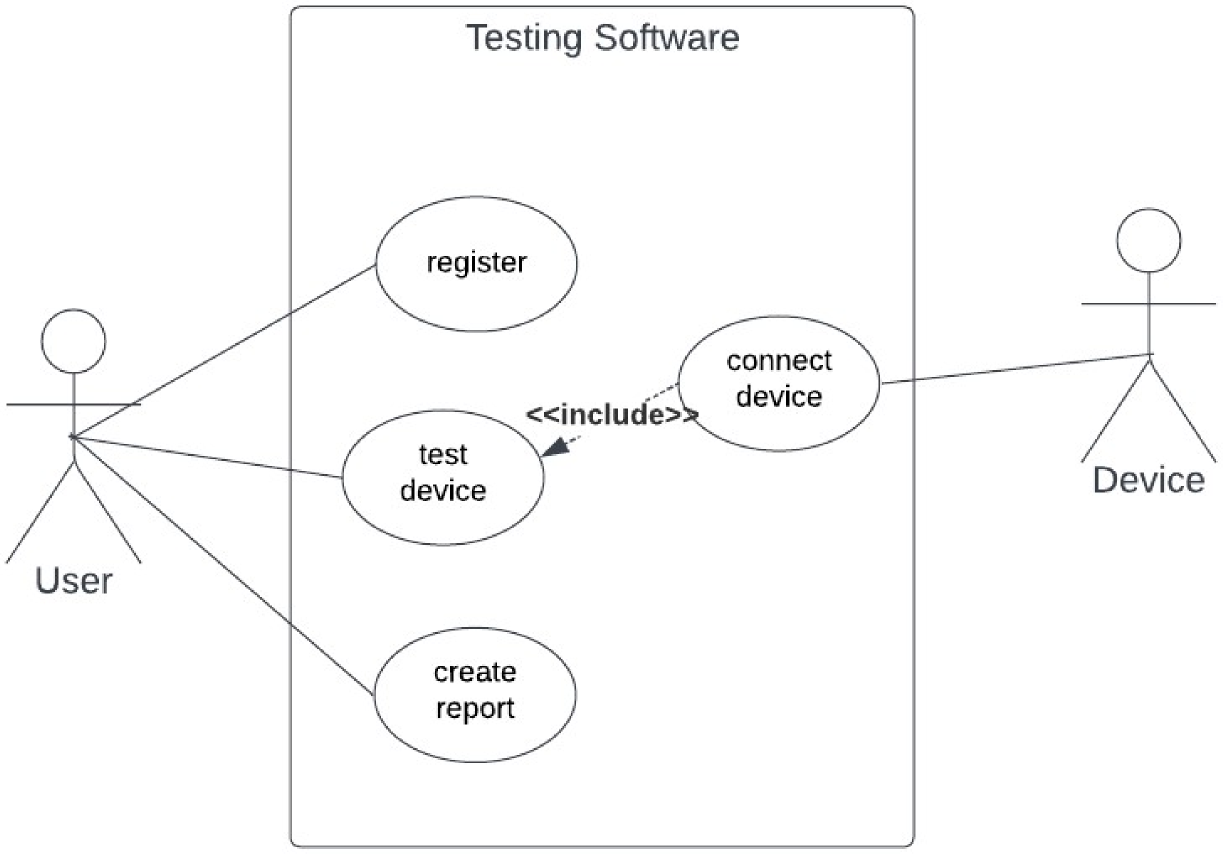
- DFD Level 1 Diagram:



6.3 Use case Diagram:

UML, use-case diagrams model the behavior of a system and help to capture the requirements of the system. Use case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally.

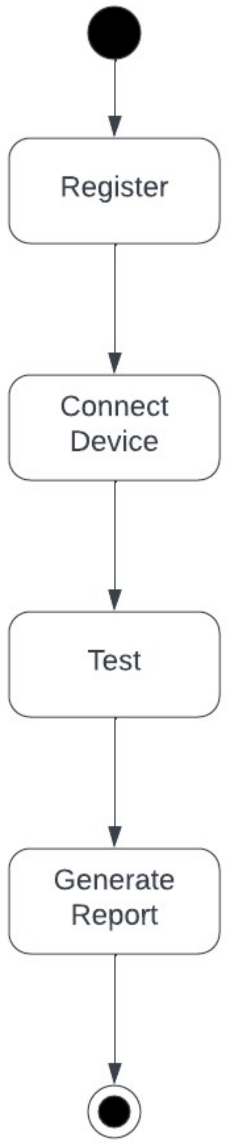
Use-case diagrams illustrate and define the context and requirements of either an entire system or the important parts of the system. You can model a complex system with a single use-case diagram, or create many use-case diagrams to model the components of the system. You would typically develop use-case diagrams in the early phases of a project and refer to them throughout the development process.



There are two actors one is the user and other is the device on the test is performed. The register himself, connect the device, test device, and create the report.

6.4 Activity Diagram:

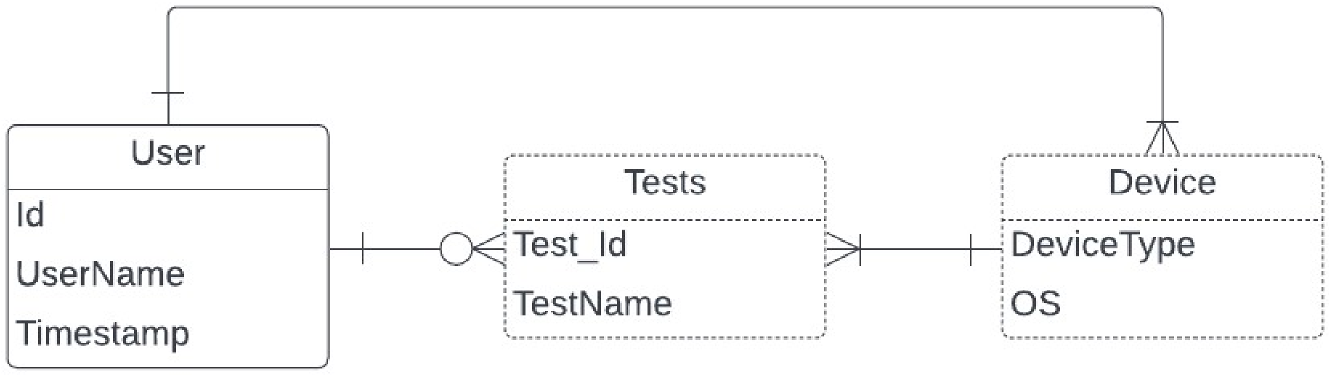
In UML, the activity diagram is used to demonstrate the flow of control within the system rather than the implementation. It models the concurrent and sequential activities. The activity diagram helps in envisioning the workflow from one activity to another. It put emphasis on the condition of flow and the order in which it occurs.



The activity diagram starts with user registering himself and connects the device to the system and perform multiple tests and generates the reports based on the test applied on the devices.

6.5 ER Diagram:

An Entity Relationship Diagram (ER Diagram) pictorially explains the relationship between entities to be stored in a database. Fundamentally, the ER Diagram is a structural design of the database. It acts as a framework created with specialized symbols for the purpose of defining the relationship between the database entities. ER diagram is created based on three principal components: entities, attributes, and relationships.



There are three entities user, tests, and device. The user has its own values like id, username and timestamp that can be used to store any information about the user like id = 44, username = sohail. User can connect a device which has its own type and operating system and can perform multiple tests on the device the test has its own unique id and name which is used to select the test.

7. Code

- **main.py**

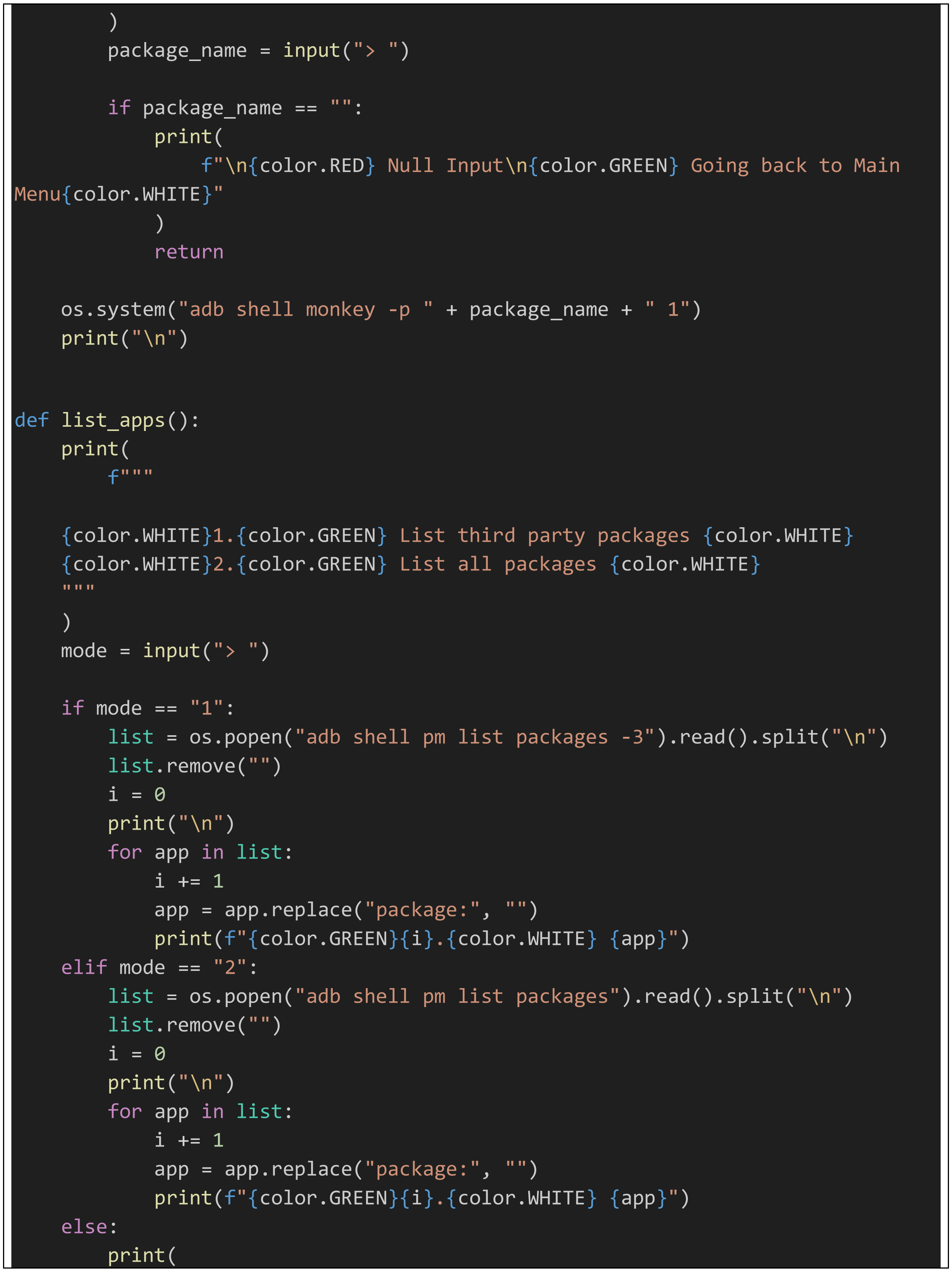
```python
import os
import random
import socket
import time
import subprocess
import platform
import datetime
from datetime import datetime
from modules import banner
from modules import color
from modules import nmap


def start():
    # Creating Downloaded-Files folder if it does not exist
    try:
        # Creates a folder to store pulled files
        os.mkdir("Downloaded-Files")
    except:
        pass

    # Checking OS
    global operating_system, opener
    operating_system = platform.system()
    if operating_system == "Windows":
        # Windows specific configuration
        windows_config()
    else:
        # macOS only
        if operating_system == "Darwin":
            opener = "open"

        # On Linux and macOS both
        import readline  # Arrow Key

        check_packages()  # Checking for required packages
```

```python
def windows_config():
    global clear, opener  # , move
    clear = "cls"
    opener = "start"
    # move = 'move'


def check_packages():
    adb_status = subprocess.call(["which", "adb"])
    scrcpy_status = subprocess.call(["which", "scrcpy"])
    metasploit_status = subprocess.call(["which", "msfconsole"])
    nmap_status = subprocess.call(["which", "nmap"])

    if (
        adb_status != 0
        or metasploit_status != 0
        or scrcpy_status != 0
        or nmap_status != 0
    ):
        print(
            f"\n{color.RED}ERROR : The following required software are NOT
installed!\n"
        )

        count = 0  # Count variable for indexing

        if adb_status != 0:
            count = count + 1
            print(f"{color.YELLOW}{count}. {color.YELLOW}ADB{color.WHITE}")

        if metasploit_status != 0:
            count = count + 1
            print(f"{color.YELLOW}{count}. Metasploit-Framework{color.WHITE}")

        if scrcpy_status != 0:
            count = count + 1
            print(f"{color.YELLOW}{count}. Scrcpy{color.WHITE}")

        if nmap_status != 0:
            count = count + 1
            print(f"{color.YELLOW}{count}. Nmap{color.WHITE}")

        print(f"\n{color.CYAN}Please install the above listed
software.{color.WHITE}\n")
```

```
        choice = input(
            f"\n{color.GREEN}Do you still want to continue to PhoneSploit
Pro?{color.WHITE}     Y / N > "
        ).lower()
        if choice == "y" or choice == "":
            return
        elif choice == "n":
            exit_phonesploit_pro()
            return
        else:
            while choice != "y" and choice != "n" and choice != "":
                choice = input("\nInvalid choice!, Press Y or N > ").lower()
                if choice == "y" or choice == "":
                    return
                elif choice == "n":
                    exit_phonesploit_pro()
                    return

def userName():
        name = input("Enter your name: ").capitalize()
        id = generate_random_number()
        timeStamp = datetime.now().strftime("%Y-%m-%d %I:%M:%S")
        save_name_to_file(timeStamp,name, id)

def generate_random_number():
    return random.randint(1000, 10000)

def save_name_to_file(timeStamp, name, id):
    filename = 'UserLog.txt'

    with open(filename, 'a') as file:
        file.write(f"TimeStamp: {timeStamp}, Name: {name}, Id: {id}\n")

    print(f"Welcome {name} your unique is {id}.")

def display_menu():
    """Displays banner and menu"""
    print(selected_banner, page)


def clear_screen():
    """Clears the screen and display menu"""
    os.system(clear)
    display_menu()
```

```
def list_devices():
    print("\n")
    os.system("adb devices -l")
    print("\n")


def disconnect():
    print("\n")
    os.system("adb disconnect")
    print("\n")


def exit_phonesploit_pro():
    global run_phonesploit_pro
    run_phonesploit_pro = False
    print("\nExiting...\n")


def get_shell():
    print("\n")
    os.system("adb shell")


def get_screenshot():
    global screenshot_location
    # Getting a temporary file name to store time specific results
    instant = datetime.datetime.now()
    file_name = f"screenshot-{instant.year}-{instant.month}-{instant.day}-
{instant.hour}-{instant.minute}-{instant.second}.png"
    os.system(f"adb shell screencap -p /sdcard/{file_name}")
    if screenshot_location == "":
        print(
            f"\n{color.YELLOW}Enter location to save all screenshots, Press
'Enter' for default{color.WHITE}"
        )
        screenshot_location = input("> ")
    if screenshot_location == "":
        screenshot_location = "Downloaded-Files"
        print(
            f"\n{color.PURPLE}Saving screenshot to PhoneSploit-
Pro/{screenshot_location}\n{color.WHITE}"
        )
    else:
        print(
            f"\n{color.PURPLE}Saving screenshot to
{screenshot_location}\n{color.WHITE}"
        )

    os.system(f"adb pull /sdcard/{file_name} {screenshot_location}")

    # Asking to open file
    choice = input(
        f"\n{color.GREEN}Do you want to Open the file?     Y / N {color.WHITE}>
"
    ).lower()
    if choice == "y" or choice == "":
        os.system(f"{opener} {screenshot_location}/{file_name}")

    elif not choice == "n":
        while choice != "y" and choice != "n" and choice != "":
            choice = input("\nInvalid choice!, Press Y or N > ").lower()
            if choice == "y" or choice == "":
                os.system(f"{opener} {screenshot_location}/{file_name}")

    print("\n")


def screenrecord():
    global screenrecord_location
    # Getting a temporary file name to store time specific results
    instant = datetime.datetime.now()
    file_name = f"vid-{instant.year}-{instant.month}-{instant.day}-
{instant.hour}-{instant.minute}-{instant.second}.mp4"

    duration = input(
        f"\n{color.CYAN}Enter the recording duration (in seconds) >
{color.WHITE}"
    )
    print(f"\n{color.YELLOW}Starting Screen Recording...\n{color.WHITE}")
    os.system(
        f"adb shell screenrecord --verbose --time-limit {duration}
/sdcard/{file_name}"
    )

    if screenrecord_location == "":
        print(
            f"\n{color.YELLOW}Enter location to save all videos, Press 'Enter'
for default{color.WHITE}"
        )
        screenrecord_location = input("> ")
    if screenrecord_location == "":
        screenrecord_location = "Downloaded-Files"
        print(
            f"\n{color.PURPLE}Saving video to PhoneSploit-
Pro/{screenrecord_location}\n{color.WHITE}"
        )
    else:
        print(f"\n{color.PURPLE}Saving video to
{screenrecord_location}\n{color.WHITE}")

    os.system(f"adb pull /sdcard/{file_name} {screenrecord_location}")

    # Asking to open file
    choice = input(
        f"\n{color.GREEN}Do you want to Open the file?     Y / N {color.WHITE}>
"
    ).lower()
    if choice == "y" or choice == "":
        os.system(f"{opener} {screenrecord_location}/{file_name}")

    elif not choice == "n":
        while choice != "y" and choice != "n" and choice != "":
            choice = input("\nInvalid choice!, Press Y or N > ").lower()
            if choice == "y" or choice == "":
                os.system(f"{opener} {screenrecord_location}/{file_name}")
    print("\n")


def pull_file():
    global pull_location
    print(
        f"\n{color.CYAN}Enter file path           {color.YELLOW}Example :
/sdcard/Download/sample.jpg{color.WHITE}"
    )
    location = input("\n> /sdcard/")
    # Checking if specified file or folder exists in Android
    if os.system(f"adb shell test -e /sdcard/{location}") == 0:
        pass
    else:
        print(
            f"{color.RED}\n[Error]{color.GREEN} Specified location does not
exist {color.GREEN}"
        )
        return
    if pull_location == "":
        print(
            f"\n{color.YELLOW}Enter location to save all files, Press 'Enter'
for default{color.WHITE}"
        )
        pull_location = input("> ")
    if pull_location == "":
        pull_location = "Downloaded-Files"
        print(
            f"\n{color.PURPLE}Saving file to PhoneSploit-
Pro/{pull_location}\n{color.WHITE}"
        )
    else:
        print(f"\n{color.PURPLE}Saving file to {pull_location}\n{color.WHITE}")
    os.system(f"adb pull /sdcard/{location} {pull_location}")

    # Asking to open file
    choice = input(
        f"\n{color.GREEN}Do you want to Open the file?     Y / N {color.WHITE}>
"
    ).lower()

    # updating location = file_name if it existed inside a folder
    # Example : sdcard/DCIM/longtime.jpg -> longtime.jpg
    file_path = location.split("/")
    location = file_path[len(file_path) - 1]

    # processing request
    if choice == "y" or choice == "":
        os.system(f"{opener} {pull_location}/{location}")

    elif not choice == "n":
        while choice != "y" and choice != "n" and choice != "":
            choice = input("\nInvalid choice!, Press Y or N > ").lower()
            if choice == "y" or choice == "":
                os.system(f"{opener} {pull_location}/{location}")


def push_file():
    location = input(f"\n{color.CYAN}Enter file path in computer{color.WHITE} >
")

    if location == "":
        print(
            f"\n{color.RED} Null Input\n{color.GREEN} Going back to Main
Menu{color.WHITE}"
        )
        return
    else:
        if operating_system == "Windows":
            file_status = int(
                os.popen(f"if exist {location} (echo 0) ELSE (echo 1)").read()
            )
        else:
            file_status = os.system(f"test -e {location}")
        if file_status == 0:
            pass
        else:
            print(
                f"{color.RED}\n[Error]{color.GREEN} Specified location does not
exist {color.GREEN}"
            )
            return
        destination = input(
            f"\n{color.CYAN}Enter destination
path              {color.YELLOW}Example : /sdcard/Documents{color.WHITE}\n>
/sdcard/"
        )
        os.system("adb push " + location + " /sdcard/" + destination)

def uninstall_app():
    print(
        f"""
    {color.WHITE}1.{color.GREEN} Select from App List
    {color.WHITE}2.{color.GREEN} Enter Package Name Manually
    {color.WHITE}"""
    )

    mode = input("> ")
    if mode == "1":
        # Listing third party apps
        list = os.popen("adb shell pm list packages -3").read().split("\n")
        list.remove("")
        i = 0
        print("\n")
        for app in list:
            i += 1
            app = app.replace("package:", "")
            print(f"{color.GREEN}{i}.{color.WHITE} {app}")
        # Selection of app
        app = input("\nEnter Selection > ")
        if app.isdigit():
            if int(app) <= len(list) and int(app) > 0:
                package = list[int(app) - 1].replace("package:", "")
                print(f"\n{color.RED}Uninstalling
{color.YELLOW}{package}{color.WHITE}")
                os.system("adb uninstall " + package)
            else:
                print(
                    f"\n{color.RED} Invalid selection\n{color.GREEN} Going back
to Main Menu{color.WHITE}"
                )
                return
        else:
            print(
                f"\n{color.RED} Expected an Integer Value\n{color.GREEN} Going
back to Main Menu{color.WHITE}"
            )
            return

    elif mode == "2":
        print(
            f"\n{color.CYAN}Enter package name     {color.WHITE}Example :
com.spotify.music "
        )
        package_name = input("> ")

        if package_name == "":
            print(
                f"\n{color.RED} Null Input\n{color.GREEN} Going back to Main
Menu{color.WHITE}"
            )
        else:
            os.system("adb uninstall " + package_name)
    else:
        print(
            f"\n{color.RED} Invalid selection\n{color.GREEN} Going back to Main
Menu{color.WHITE}"
        )
        return

    print("\n")
```

```
def launch_app():
    print(
        f"""
    {color.WHITE}1.{color.GREEN} Select from App List
    {color.WHITE}2.{color.GREEN} Enter Package Name Manually
    {color.WHITE}"""
    )

    mode = input("> ")
    if mode == "1":
        # Listing third party apps
        list = os.popen("adb shell pm list packages -3").read().split("\n")
        list.remove("")
        i = 0
        print("\n")
        for app in list:
            i += 1
            app = app.replace("package:", "")
            print(f"{color.GREEN}{i}.{color.WHITE} {app}")

        # Selection of app
        app = input("\nEnter Selection > ")
        if app.isdigit():
            if int(app) <= len(list) and int(app) > 0:
                package_name = list[int(app) - 1].replace("package:", "")
            else:
                print(
                    f"\n{color.RED} Invalid selection\n{color.GREEN} Going back
to Main Menu{color.WHITE}"
                )
                return
        else:
            print(
                f"\n{color.RED} Expected an Integer Value\n{color.GREEN} Going
back to Main Menu{color.WHITE}"
            )
            return

    elif mode == "2":
        ## Old
        print(
            f"\n{color.CYAN}Enter package name :     {color.WHITE}Example :
com.spotify.music "
```



```python
        )
        package_name = input("> ")

        if package_name == "":
            print(
                f"\n{color.RED} Null Input\n{color.GREEN} Going back to Main
Menu{color.WHITE}"
            )
            return

    os.system("adb shell monkey -p " + package_name + " 1")
    print("\n")


def list_apps():
    print(
        f"""

    {color.WHITE}1.{color.GREEN} List third party packages {color.WHITE}
    {color.WHITE}2.{color.GREEN} List all packages {color.WHITE}
    """
    )
    mode = input("> ")

    if mode == "1":
        list = os.popen("adb shell pm list packages -3").read().split("\n")
        list.remove("")
        i = 0
        print("\n")
        for app in list:
            i += 1
            app = app.replace("package:", "")
            print(f"{color.GREEN}{i}.{color.WHITE} {app}")
    elif mode == "2":
        list = os.popen("adb shell pm list packages").read().split("\n")
        list.remove("")
        i = 0
        print("\n")
        for app in list:
            i += 1
            app = app.replace("package:", "")
            print(f"{color.GREEN}{i}.{color.WHITE} {app}")
    else:
        print(
            f"\n{color.RED} Invalid selection\n{color.GREEN} Going back to Main
Menu{color.WHITE}"
        )
    print("\n")


def reboot(key):
    print(
        f"\n{color.RED}[Warning]{color.YELLOW} Restarting will disconnect the
device{color.WHITE}"
    )
    choice = input("\nDo you want to continue?     Y / N > ").lower()
    if choice == "y" or choice == "":
        pass
    elif choice == "n":
        return
    else:
        while choice != "y" and choice != "n" and choice != "":
            choice = input("\nInvalid choice!, Press Y or N > ").lower()
            if choice == "y" or choice == "":
                pass
            elif choice == "n":
                return

    if key == "system":
        os.system("adb reboot")
    else:
        print(
            f"""
    {color.WHITE}1.{color.GREEN} Reboot to Recovery Mode
    {color.WHITE}2.{color.GREEN} Reboot to Bootloader
    {color.WHITE}3.{color.GREEN} Reboot to Fastboot Mode
    {color.WHITE}"""
        )
        mode = input("> ")
        if mode == "1":
            os.system("adb reboot recovery")
        elif mode == "2":
            os.system("adb reboot bootloader")
        elif mode == "3":
            os.system("adb reboot fastboot")
        else:
            print(
                f"\n{color.RED} Invalid selection\n{color.GREEN} Going back to
Main Menu{color.WHITE}"
```

```
            )
            return

    print("\n")


# def list_files():
    print("\n")
    os.system("adb shell ls -a /sdcard/")
    print("\n")

def copy_whatsapp():
    global pull_location
    if pull_location == "":
        print(
            f"\n{color.YELLOW}Enter location to save WhatsApp Data, Press
'Enter' for default{color.WHITE}"
        )
        pull_location = input("> ")
    if pull_location == "":
        pull_location = "Downloaded-Files"
        print(
            f"\n{color.PURPLE}Saving data to PhoneSploit-
Pro/{pull_location}\n{color.WHITE}"
        )
    else:
        print(f"\n{color.PURPLE}Saving data to {pull_location}\n{color.WHITE}")
    if (
        os.system('adb shell test -d
"/sdcard/Android/media/com.whatsapp/WhatsApp"')
        == 0
    ):
        location = "/sdcard/Android/media/com.whatsapp/WhatsApp"
    elif os.system('adb shell test -d "/sdcard/WhatsApp"') == 0:
        location = "/sdcard/WhatsApp"
    else:
        print(
            f"{color.RED}\n[Error]{color.GREEN} WhatsApp folder does not exist
{color.GREEN}"
        )
        return

    os.system(f"adb pull {location} {pull_location}")
    print("\n")
```

```python
def copy_screenshots():
    global pull_location
    if pull_location == "":
        print(
            f"\n{color.YELLOW}Enter location to save all Screenshots, Press
'Enter' for default{color.WHITE}"
        )
        pull_location = input("> ")

    if pull_location == "":
        pull_location = "Downloaded-Files"
        print(
            f"\n{color.PURPLE}Saving Screenshots to PhoneSploit-
Pro/{pull_location}\n{color.WHITE}"
        )
    else:
        print(f"\n{color.PURPLE}Saving Screenshots to
{pull_location}\n{color.WHITE}")

    # Checking if folder exists
    if os.system('adb shell test -d "/sdcard/Pictures/Screenshots"') == 0:
        location = "/sdcard/Pictures/Screenshots"
    elif os.system('adb shell test -d "/sdcard/DCIM/Screenshots"') == 0:
        location = "/sdcard/DCIM/Screenshots"
    elif os.system('adb shell test -d "/sdcard/Screenshots"') == 0:
        location = "/sdcard/Screenshots"
    else:
        print(
            f"{color.RED}\n[Error]{color.GREEN} Screenshots folder does not
exist {color.GREEN}"
        )
        return
    os.system(f"adb pull {location} {pull_location}")
    print("\n")




def copy_camera():
    global pull_location
    if pull_location == "":
        print(
            f"\n{color.YELLOW}Enter location to save all Photos, Press 'Enter'
for default{color.WHITE}"
        )
        pull_location = input("> ")
    if pull_location == "":
        pull_location = "Downloaded-Files"
        print(
            f"\n{color.PURPLE}Saving Photos to PhoneSploit-
Pro/{pull_location}\n{color.WHITE}"
        )
    else:
        print(f"\n{color.PURPLE}Saving Photos to
{pull_location}\n{color.WHITE}")

    # Checking if folder exists
    if os.system('adb shell test -d "/sdcard/DCIM/Camera"') == 0:
        location = "/sdcard/DCIM/Camera"
    else:
        print(
            f"{color.RED}\n[Error]{color.GREEN} Camera folder does not exist
{color.GREEN}"
        )
        return
    os.system(f"adb pull {location} {pull_location}")
    print("\n")


def anonymous_screenshot():
    global screenshot_location
    # Getting a temporary file name to store time specific results
    instant = datetime.datetime.now()
    file_name = f"screenshot-{instant.year}-{instant.month}-{instant.day}-
{instant.hour}-{instant.minute}-{instant.second}.png"
    os.system(f"adb shell screencap -p /sdcard/{file_name}")
    if screenshot_location == "":
        print(
            f"\n{color.YELLOW}Enter location to save all screenshots, Press
'Enter' for default{color.WHITE}"
        )
        screenshot_location = input("> ")
    if screenshot_location == "":
        screenshot_location = "Downloaded-Files"
        print(
            f"\n{color.PURPLE}Saving screenshot to PhoneSploit-
Pro/{screenshot_location}\n{color.WHITE}"
```

```
        )
    else:
        print(
            f"\n{color.PURPLE}Saving screenshot to
{screenshot_location}\n{color.WHITE}"
        )

    os.system(f"adb pull /sdcard/{file_name} {screenshot_location}")

    print(f"\n{color.YELLOW}Deleting screenshot from Target
device\n{color.WHITE}")
    os.system(f"adb shell rm /sdcard/{file_name}")

    # Asking to open file
    choice = input(
        f"\n{color.GREEN}Do you want to Open the file?     Y / N {color.WHITE}>
"
    ).lower()
    if choice == "y" or choice == "":
        os.system(f"{opener} {screenshot_location}/{file_name}")

    elif not choice == "n":
        while choice != "y" and choice != "n" and choice != "":
            choice = input("\nInvalid choice!, Press Y or N > ").lower()
            if choice == "y" or choice == "":
                os.system(f"{opener} {screenshot_location}/{file_name}")

    print("\n")


def anonymous_screenrecord():
    global screenrecord_location
    # Getting a temporary file name to store time specific results
    instant = datetime.datetime.now()
    file_name = f"vid-{instant.year}-{instant.month}-{instant.day}-
{instant.hour}-{instant.minute}-{instant.second}.mp4"

    duration = input(
        f"\n{color.CYAN}Enter the recording duration (in seconds) >
{color.WHITE}"
    )
    print(f"\n{color.YELLOW}Starting Screen Recording...\n{color.WHITE}")
    os.system(
        f"adb shell screenrecord --verbose --time-limit {duration}
/sdcard/{file_name}"
```

```
    )

    if screenrecord_location == "":
        print(
            f"\n{color.YELLOW}Enter location to save all videos, Press 'Enter' 
for default{color.WHITE}"
        )
        screenrecord_location = input("> ")
    if screenrecord_location == "":
        screenrecord_location = "Downloaded-Files"
        print(
            f"\n{color.PURPLE}Saving video to PhoneSploit-
Pro/{screenrecord_location}\n{color.WHITE}"
        )
    else:
        print(f"\n{color.PURPLE}Saving video to 
{screenrecord_location}\n{color.WHITE}")

    os.system(f"adb pull /sdcard/{file_name} {screenrecord_location}")

    print(f"\n{color.YELLOW}Deleting video from Target device\n{color.WHITE}")
    os.system(f"adb shell rm /sdcard/{file_name}")
    # Asking to open file
    choice = input(
        f"\n{color.GREEN}Do you want to Open the file?     Y / N {color.WHITE}> 
"
    ).lower()
    if choice == "y" or choice == "":
        os.system(f"{opener} {screenrecord_location}/{file_name}")

    elif not choice == "n":
        while choice != "y" and choice != "n" and choice != "":
            choice = input("\nInvalid choice!, Press Y or N > ").lower()
            if choice == "y" or choice == "":
                os.system(f"{opener} {screenrecord_location}/{file_name}")
    print("\n")


# def use_keycode():
    keycodes = True
    os.system(clear)
    print(banner.keycode_menu)
    while keycodes:
        print(f"\n {color.CYAN}99 : Clear Screen                0 : Main Menu")
        keycode_option = input(
            f"{color.RED}\n[KEYCODE] {color.WHITE}Enter selection > "
        ).lower()

        match keycode_option:
            case "0":
                keycodes = False
                display_menu()
            case "99":
                os.system(clear)
                print(banner.keycode_menu)
            case "1":
                text = input(f"\n{color.CYAN}Enter text > {color.WHITE}")
                os.system(f'adb shell input text "{text}"')
                print(f'{color.YELLOW}\nEntered {color.WHITE}"{text}"')
            case "2":
                os.system("adb shell input keyevent 3")
                print(f"{color.YELLOW}\nPressed Home Button{color.WHITE}")
            case "3":
                os.system("adb shell input keyevent 4")
                print(f"{color.YELLOW}\nPressed Back Button{color.WHITE}")
            case "4":
                os.system("adb shell input keyevent 187")
                print(f"{color.YELLOW}\nPressed Recent Apps
Button{color.WHITE}")
            case "5":
                os.system("adb shell input keyevent 26")
                print(f"{color.YELLOW}\nPressed Power Key{color.WHITE}")
            case "6":
                os.system("adb shell input keyevent 19")
                print(f"{color.YELLOW}\nPressed DPAD Up{color.WHITE}")
            case "7":
                os.system("adb shell input keyevent 20")
                print(f"{color.YELLOW}\nPressed DPAD Down{color.WHITE}")
            case "8":
                os.system("adb shell input keyevent 21")
                print(f"{color.YELLOW}\nPressed DPAD Left{color.WHITE}")
            case "9":
                os.system("adb shell input keyevent 22")
                print(f"{color.YELLOW}\nPressed DPAD Right{color.WHITE}")
            case "10":
                os.system("adb shell input keyevent 67")
                print(f"{color.YELLOW}\nPressed Delete/Backspace{color.WHITE}")
            case "11":
                os.system("adb shell input keyevent 66")
                print(f"{color.YELLOW}\nPressed Enter{color.WHITE}")
            case "12":
                os.system("adb shell input keyevent 24")
                print(f"{color.YELLOW}\nPressed Volume Up{color.WHITE}")
            case "13":
                os.system("adb shell input keyevent 25")
                print(f"{color.YELLOW}\nPressed Volume Down{color.WHITE}")
            case "14":
                os.system("adb shell input keyevent 126")
                print(f"{color.YELLOW}\nPressed Media Play{color.WHITE}")
            case "15":
                os.system("adb shell input keyevent 127")
                print(f"{color.YELLOW}\nPressed Media Pause{color.WHITE}")
            case "16":
                os.system("adb shell input keyevent 61")
                print(f"{color.YELLOW}\nPressed Tab Key{color.WHITE}")
            case "17":
                os.system("adb shell input keyevent 111")
                print(f"{color.YELLOW}\nPressed Esc Key{color.WHITE}")

            case other:
                print("\nInvalid selection!\n")

def open_photo():
    location = input(
        f"\n{color.YELLOW}Enter Photo location in computer{color.WHITE} > "
    )

    if location == "":
        print(
            f"\n{color.RED} Null Input\n{color.GREEN} Going back to Main
Menu{color.WHITE}"
        )
        return
    else:
        if location[len(location) - 1] == " ":
            location = location.removesuffix(" ")
        location = location.replace("'", "")
        location = location.replace('"', "")
        if not os.path.isfile(location):
            print(
                f"{color.RED}\n[Error]{color.GREEN} This file does not exist
{color.GREEN}"
            )
            return
        else:
            location = '"' + location + '"'
            os.system("adb push " + location + " /sdcard/")

        file_path = location.split("/")
        file_name = file_path[len(file_path) - 1]

        # Reverse slash ('\') splitting for Windows only
        global operating_system
        if operating_system == "Windows":
            file_path = file_name.split("\\")
            file_name = file_path[len(file_path) - 1]

        file_name = file_name.replace("'", "")
        file_name = file_name.replace('"', "")
        file_name = "'" + file_name + "'"
        print(file_name)
        print(f"\n{color.YELLOW}Opening Photo on
device        \n{color.WHITE}")
        os.system(
            f'adb shell am start -a android.intent.action.VIEW -d
"file:///sdcard/{file_name}" -t image/jpeg'
        )  # -n com.android.chrome/com.google.android.apps.chrome.Main
        print("\n")


def open_video():
    location = input(
        f"\n{color.YELLOW}Enter Video location in computer{color.WHITE} > "
    )

    if location == "":
        print(
            f"\n{color.RED} Null Input\n{color.GREEN} Going back to Main
Menu{color.WHITE}"
        )
        return
    else:
        if location[len(location) - 1] == " ":
            location = location.removesuffix(" ")
        location = location.replace("'", "")
        location = location.replace('"', "")
        if not os.path.isfile(location):
            print(
                f"{color.RED}\n[Error]{color.GREEN} This file does not exist
{color.GREEN}"
```

```
            )
            return
        else:
            location = '"' + location + '"'
            os.system("adb push " + location + " /sdcard/")

        file_path = location.split("/")
        file_name = file_path[len(file_path) - 1]

        # Reverse slash ('\') splitting for Windows only
        global operating_system
        if operating_system == "Windows":
            file_path = file_name.split("\\")
            file_name = file_path[len(file_path) - 1]

        file_name = file_name.replace("'", "")
        file_name = file_name.replace('"', "")
        file_name = "'" + file_name + "'"
        print(file_name)

        print(f"\n{color.YELLOW}Playing Video on
device        \n{color.WHITE}")
        os.system(
            f'adb shell am start -a android.intent.action.VIEW -d
"file:///sdcard/{file_name}" -t video/mp4'
        )
        print("\n")


def get_device_info():
    model = os.popen(f"adb shell getprop ro.product.model").read()
    manufacturer = os.popen(f"adb shell getprop
ro.product.manufacturer").read()
    chipset = os.popen(f"adb shell getprop ro.product.board").read()
    android = os.popen(f"adb shell getprop ro.build.version.release").read()
    security_patch = os.popen(
        f"adb shell getprop ro.build.version.security_patch"
    ).read()
    device = os.popen(f"adb shell getprop ro.product.vendor.device").read()
    sim = os.popen(f"adb shell getprop gsm.sim.operator.alpha").read()
    encryption_state = os.popen(f"adb shell getprop ro.crypto.state").read()
    build_date = os.popen(f"adb shell getprop ro.build.date").read()
    sdk_version = os.popen(f"adb shell getprop ro.build.version.sdk").read()
    wifi_interface = os.popen(f"adb shell getprop wifi.interface").read()
    print(
        f"""
    {color.YELLOW}Model :{color.WHITE} {model}\
    {color.YELLOW}Manufacturer :{color.WHITE} {manufacturer}\
    {color.YELLOW}Chipset :{color.WHITE} {chipset}\
    {color.YELLOW}Android Version :{color.WHITE} {android}\
    {color.YELLOW}Security Patch :{color.WHITE} {security_patch}\
    {color.YELLOW}Device :{color.WHITE} {device}\
    {color.YELLOW}SIM :{color.WHITE} {sim}\
    {color.YELLOW}Encryption State :{color.WHITE} {encryption_state}\
    {color.YELLOW}Build Date :{color.WHITE} {build_date}\
    {color.YELLOW}SDK Version :{color.WHITE} {sdk_version}\
    {color.YELLOW}WiFi Interface :{color.WHITE} {wifi_interface}\
"""
    )


def battery_info():
    battery = os.popen(f"adb shell dumpsys battery").read()
    print(
        f"""\n{color.YELLOW}Battery Information :
{color.WHITE}{battery}\n"""

def unlock_device():
    password = input(
        f"{color.YELLOW}\nEnter password or Press 'Enter' for
blank{color.WHITE} > "
    )
    os.system("adb shell input keyevent 26")
    os.system("adb shell input swipe 200 900 200 300 200")
    if not password == "":  # if password is not blank
        os.system(f'adb shell input text "{password}"')
    os.system("adb shell input keyevent 66")
    print(f"{color.GREEN}\nDevice unlocked{color.WHITE}")


def lock_device():
    os.system("adb shell input keyevent 26")
    print(f"{color.GREEN}\nDevice locked{color.WHITE}")


def mirror():
    print(
        f"""
    {color.WHITE}1.{color.GREEN} Default Mode   {color.YELLOW}(Best quality)
    {color.WHITE}2.{color.GREEN} Fast Mode      {color.YELLOW}(Low quality but 
high performance)
    {color.WHITE}3.{color.GREEN} Custom Mode    {color.YELLOW}(Tweak settings 
to increase performance)
    {color.WHITE}"""
    )
    mode = input("> ")
    if mode == "1":
        os.system("scrcpy")
    elif mode == "2":
        os.system("scrcpy -m 1024 -b 1M")
    elif mode == "3":
        print(f"\n{color.CYAN}Enter size limit {color.YELLOW}(e.g. 
1024){color.WHITE}")
        size = input("> ")
        if not size == "":
            size = "-m " + size

        print(
            f"\n{color.CYAN}Enter bit-rate {color.YELLOW}(e.g. 
2)   {color.WHITE}(Default : 8 Mbps)"
        )
        bitrate = input("> ")
        if not bitrate == "":
            bitrate = "-b " + bitrate + "M"

        print(f"\n{color.CYAN}Enter frame-rate {color.YELLOW}(e.g. 
15){color.WHITE}")
        framerate = input("> ")
        if not framerate == "":
            framerate = "--max-fps=" + framerate

        os.system(f"scrcpy {size} {bitrate} {framerate}")
    else:
        print(
            f"\n{color.RED} Invalid selection\n{color.GREEN} Going back to Main 
Menu{color.WHITE}"
        )
        return
    print("\n")


def power_off():
    print(
        f"\n{color.RED}[Warning]{color.YELLOW} Powering off device will
disconnect the device{color.WHITE}"
    )
    choice = input("\nDo you want to continue?     Y / N > ").lower()
    if choice == "y" or choice == "":
        pass
    elif choice == "n":
        return
    else:
        while choice != "y" and choice != "n" and choice != "":
            choice = input("\nInvalid choice!, Press Y or N > ").lower()
            if choice == "y" or choice == "":
                pass
            elif choice == "n":
                return
    os.system(f"adb shell reboot -p")
    print("\n")


def create_report(users):
    with open("Report.txt", "a") as log_file:
        for user in users:
            timestamp = datetime.now().strftime("%Y-%m-%d %I:%M:%S")
            log_data = f"\nTimestamp: {timestamp}\nName: {user['name']}\nID:
{user['id']}\nDescription: {user['description']}\n-----------------------------
-------------"
            log_file.write(log_data)

def report():
    num_users = int(input("Welcome! Please enter the number of users:"))

    users = []

    for _ in range(num_users):
        name = input("Name: ").capitalize()
        while name=="":
            print("Please enter your name:")
            name = input("Name: ").capitalize()
        user_id = input("ID: ")
        while user_id=="" or user_id.isalpha():
            print("Please a valid id:")
            user_id = input("ID: ")
        description = input("Description: ")
        while description=="":
            print("Description is mandatory:")
            description = input("Description: ")

        user_data = {'name': name, 'id': user_id, 'description': description}
        users.append(user_data)

    create_report(users)

    print("Log file created successfully!")

def main():
    print(f"\n {color.WHITE}cls: Clear Screen                e: Exit")
    option = input(f"\n{color.RED}[Main Menu] {color.WHITE}Enter selection >
").lower()

    match option:
        case "release":
            from modules import release
        case "e":
            exit_phonesploit_pro()
        case "cls":
            clear_screen()
        case "1":
            userName()
        case "2":
            list_devices()
        case "3":
            disconnect()
        case "4":
            mirror()
        case "5":
            get_screenshot()
        case "6":
            screenrecord()
        case "7":
            pull_file()
        case "8":
            push_file()
        case "9":
            launch_app()
        case "10":
            anonymous_screenrecord()
        case "11":
            open_link()
        case "12":
            open_photo()
        case "13":
            open_video()
        case "14":
            get_device_info()
        case "15":
            battery_info()
        case "16":
            reboot("system")
        case "17":
            reboot("advanced")
        case "18":
            lock_device()
        case "19":
            uninstall_app()
        case "20":
            list_apps()
        case "21":
            get_shell()
        case "22":
            list_files()
        case "23":
            copy_whatsapp()
        case "24":
            copy_screenshots()
        case "25":
            copy_camera()
        case "26":
            anonymous_screenshot()
        case "27":
            extract_apk()
        case "28":
            stop_adb()
        case "29":
            power_off()
        case "30":
            report()
        case other:
            print("\nInvalid selection!\n")


# Global variables
run_phonesploit_pro = True
operating_system = ""
clear = "clear"
opener = "xdg-open"
```

```
# move = 'mv'
page_number = 0
page = banner.menu[page_number]

# Locations
screenshot_location = ""
screenrecord_location = ""
pull_location = ""

# Concatenating banner color with the selected banner
selected_banner = random.choice(color.color_list) +
random.choice(banner.banner_list)

start()

if run_phonesploit_pro:
    clear_screen()
    while run_phonesploit_pro:
        try:
            main()
        except KeyboardInterrupt:
            exit_phonesploit_pro()
```

8. Testing

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. It includes a set of techniques and methods to identify defects, bugs, performance issues and providing a reliable and quality product. The goal is to identify issues as early as possible and improve the overall quality of the system.

Types of Testing:

- **Unit Testing:** Unit testing is a type of testing that is used to evaluate the individual units or components of a software system. This type of testing helps ensure that each unit or component of the system is working correctly and is able to perform its intended function.

- **Integration Testing:** Integration testing is a type of testing that is used to evaluate how well the different units or components of a software system work together. This type of testing helps to identify and resolve issues related to compatibility, performance, and data flow between the different units or components.

- **Functional Testing:** Functional testing is a type of testing that is used to evaluate how well a system or software performs the specific functions or tasks that it is designed to perform. It is done by testing the system or software with various inputs and verifying that the outputs are correct. This type of testing ensures that the system or software is working as intended and can perform the functions it was designed to perform

- **White Box Testing:** White box testing, also known as structural testing or glass-box testing, is a type of testing that examines the internal structure and implementation of a software system. It involves testing the code itself and checking that it is functioning correctly and adhering to coding standards. This type of testing helps to identify and resolve issues related to logic, control flow, and data structures within the system.

- **Black Box Testing:** Black box testing, also known as functional testing, is a type of testing that examines the external behavior and interfaces of a software system. It involves testing the system from the user's perspective, without looking at the internal structure or implementation, and checking that it is functioning correctly and meeting the requirements. This type of testing helps to identify and resolve issues related to usability, compatibility, and performance.

8.1 Test Results

Test Case 1:

Test Case Id	001
Test Type	Unit Test.
Name of Test	Verify registration of user.
Description	To check the functioning of registration.
Steps	1) Select number for registration of user. 2) Insert user name.
Input	Name: sohail
Expected Output	Welcome "username" your unique ID is 9701.
Actual Output	Welcome Sohail your unique ID is 9701.
Result	Pass.

Test Case 2:

Test Case Id	002
Test Type	Unit Test
Name of Test	Verify list of connected devices
Description	To check the List of connected devices
Steps	1) Connect a device via USB. 2) Select number for listing connected devices.
Input	None
Expected Output	List of connected devices should be displayed.
Actual Output	List of devices connected device displayed successfully.
Result	Pass

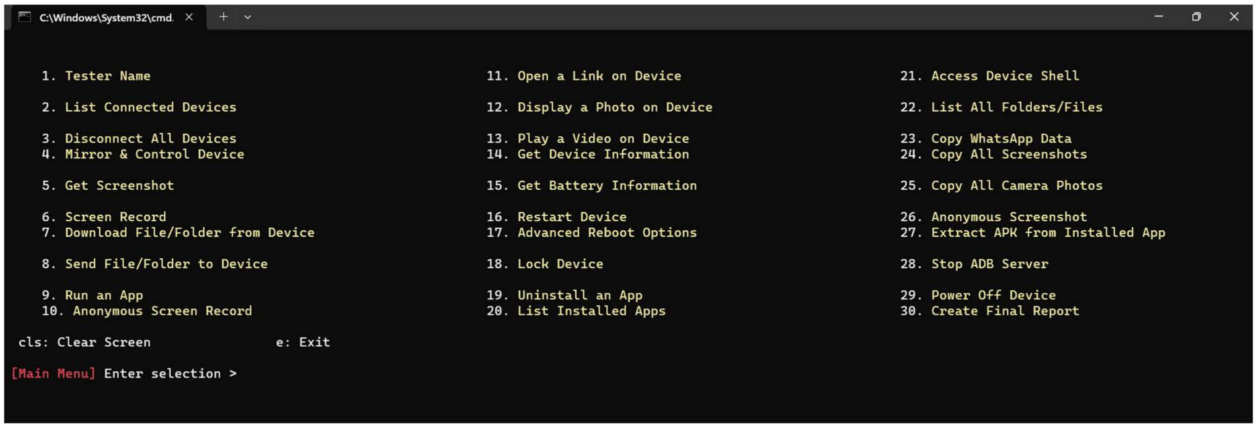
Test Case 3:

Test Case Id	003
Test Type	Functional Test.
Name of Test	Generation of report.
Description	To check whether the report is generated.
Steps	1) Select number for generating report. 2) Enter number of users. 3) Enter name. 4) Enter ID. 5) Enter Description.
Input	Number of users: 1 Name: sohail ID: 44 Description: All functions in working status.
Expected Output	Log file should be created successfully.
Actual Output	Log file created successfully.
Result	Pass.

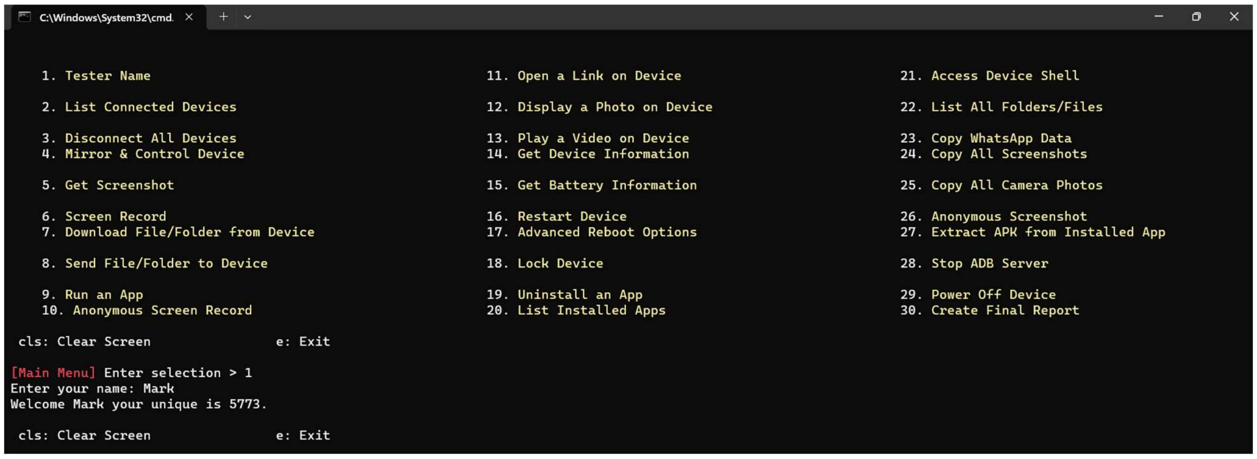
Test Case 4:

Test Case Id	004
Test Type	Unit Test.
Name of Test	Get battery information.
Description	To check whether the device battery is displayed.
Steps	1) Connect a device via USB. 2) Select number for getting battery information.
Input	None
Expected Output	Battery information should be displayed.
Actual Output	Battery information displayed successfully.
Result	Pass.

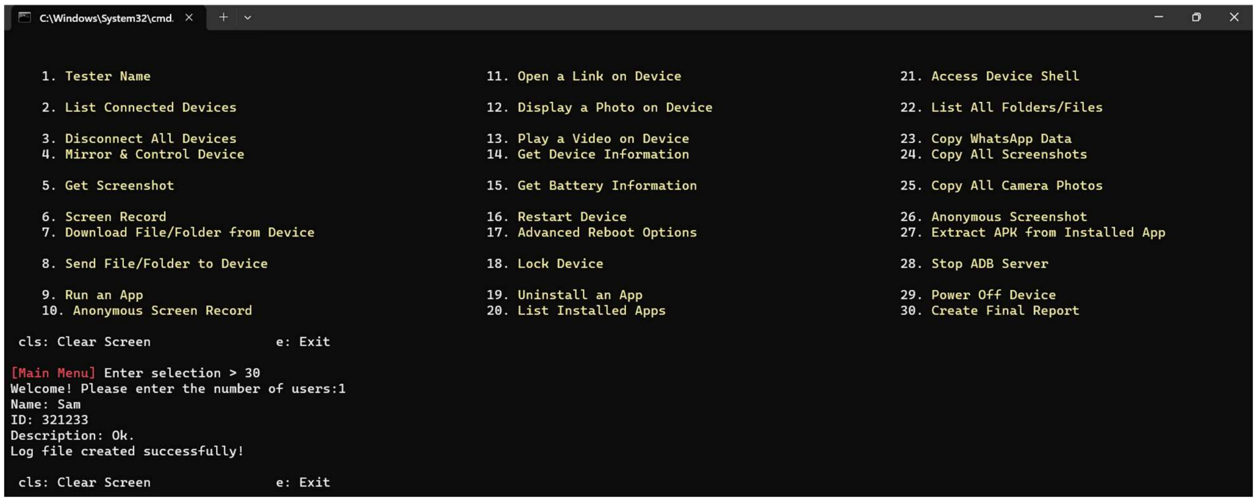
9. Screenshots



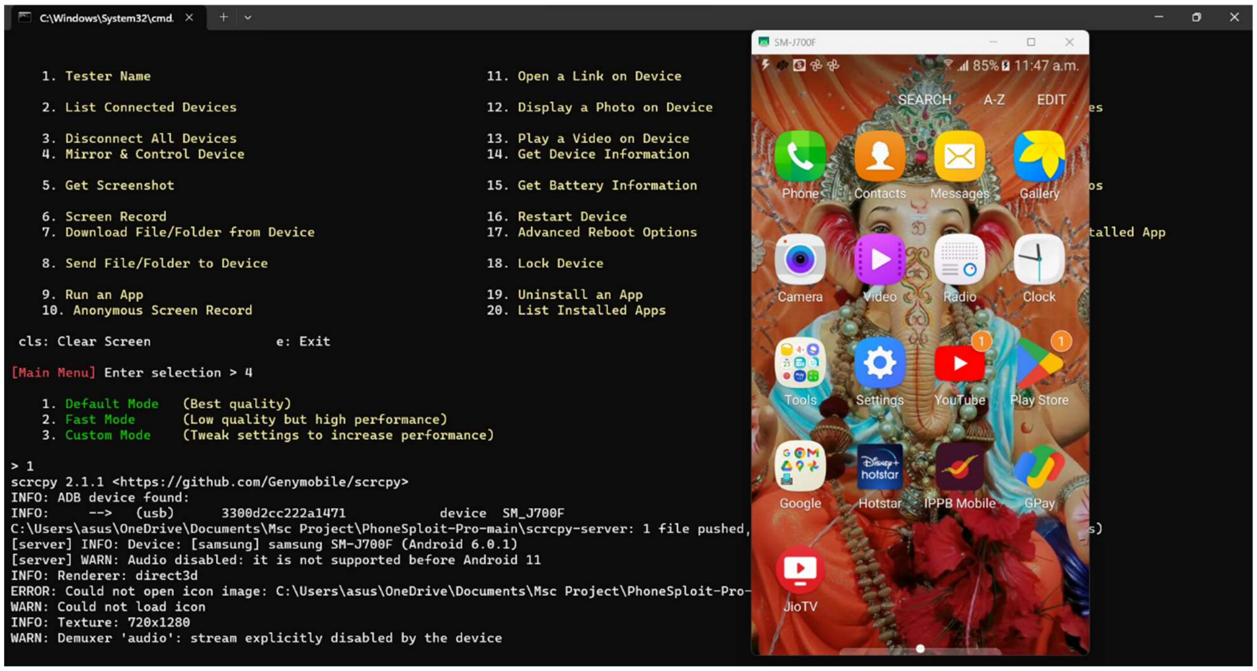
```
C:\Windows\System32\cmd.

    1. Tester Name                                         11. Open a Link on Device                        21. Access Device Shell

    2. List Connected Devices                              12. Display a Photo on Device                    22. List All Folders/Files

    3. Disconnect All Devices                              13. Play a Video on Device                       23. Copy WhatsApp Data
    4. Mirror & Control Device                             14. Get Device Information                       24. Copy All Screenshots

    5. Get Screenshot                                      15. Get Battery Information                      25. Copy All Camera Photos

    6. Screen Record                                       16. Restart Device                               26. Anonymous Screenshot
    7. Download File/Folder from Device                    17. Advanced Reboot Options                      27. Extract APK from Installed App

    8. Send File/Folder to Device                          18. Lock Device                                  28. Stop ADB Server

    9. Run an App                                          19. Uninstall an App                             29. Power Off Device
    10. Anonymous Screen Record                            20. List Installed Apps                          30. Create Final Report

  cls: Clear Screen                e: Exit

[Main Menu] Enter selection >
```



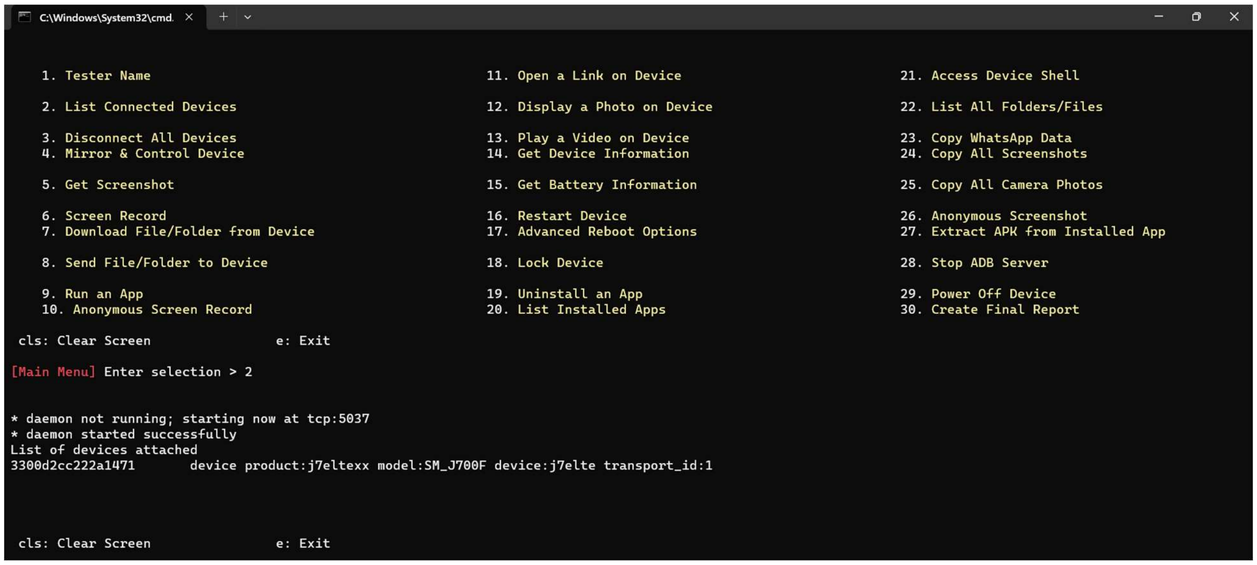
```
C:\Windows\System32\cmd.

    1. Tester Name                                         11. Open a Link on Device                        21. Access Device Shell

    2. List Connected Devices                              12. Display a Photo on Device                    22. List All Folders/Files

    3. Disconnect All Devices                              13. Play a Video on Device                       23. Copy WhatsApp Data
    4. Mirror & Control Device                             14. Get Device Information                       24. Copy All Screenshots

    5. Get Screenshot                                      15. Get Battery Information                      25. Copy All Camera Photos

    6. Screen Record                                       16. Restart Device                               26. Anonymous Screenshot
    7. Download File/Folder from Device                    17. Advanced Reboot Options                      27. Extract APK from Installed App

    8. Send File/Folder to Device                          18. Lock Device                                  28. Stop ADB Server

    9. Run an App                                          19. Uninstall an App                             29. Power Off Device
    10. Anonymous Screen Record                            20. List Installed Apps                          30. Create Final Report

  cls: Clear Screen                e: Exit

[Main Menu] Enter selection > 1
Enter your name: Mark
Welcome Mark your unique is 5773.

  cls: Clear Screen                e: Exit
```



```
C:\Windows\System32\cmd.

    1. Tester Name                                         11. Open a Link on Device                        21. Access Device Shell

    2. List Connected Devices                              12. Display a Photo on Device                    22. List All Folders/Files

    3. Disconnect All Devices                              13. Play a Video on Device                       23. Copy WhatsApp Data
    4. Mirror & Control Device                             14. Get Device Information                       24. Copy All Screenshots

    5. Get Screenshot                                      15. Get Battery Information                      25. Copy All Camera Photos

    6. Screen Record                                       16. Restart Device                               26. Anonymous Screenshot
    7. Download File/Folder from Device                    17. Advanced Reboot Options                      27. Extract APK from Installed App

    8. Send File/Folder to Device                          18. Lock Device                                  28. Stop ADB Server

    9. Run an App                                          19. Uninstall an App                             29. Power Off Device
    10. Anonymous Screen Record                            20. List Installed Apps                          30. Create Final Report

  cls: Clear Screen                e: Exit

[Main Menu] Enter selection > 30
Welcome! Please enter the number of users:1
Name: Sam
ID: 321233
Description: Ok.
Log file created successfully!

  cls: Clear Screen                e: Exit
```



```
    1. Tester Name                                        11. Open a Link on Device

    2. List Connected Devices                             12. Display a Photo on Device                  es

    3. Disconnect All Devices                             13. Play a Video on Device
    4. Mirror & Control Device                            14. Get Device Information

    5. Get Screenshot                                     15. Get Battery Information                    os

    6. Screen Record                                      16. Restart Device
    7. Download File/Folder from Device                   17. Advanced Reboot Options                    talled App

    8. Send File/Folder to Device                         18. Lock Device

    9. Run an App                                         19. Uninstall an App
   10. Anonymous Screen Record                            20. List Installed Apps

 cls: Clear Screen                 e: Exit

[Main Menu] Enter selection > 4

    1. Default Mode    (Best quality)
    2. Fast Mode       (Low quality but high performance)
    3. Custom Mode     (Tweak settings to increase performance)

> 1
scrcpy 2.1.1 <https://github.com/Genymobile/scrcpy>
INFO: ADB device found:
INFO:     -->   (usb)      3300d2cc222a1471            device  SM_J700F
C:\Users\asus\OneDrive\Documents\Msc Project\PhoneSploit-Pro-main\scrcpy-server: 1 file pushed,           s)
[server] INFO: Device: [samsung] samsung SM-J700F (Android 6.0.1)
[server] WARN: Audio disabled: it is not supported before Android 11
INFO: Renderer: direct3d
ERROR: Could not open icon image: C:\Users\asus\OneDrive\Documents\Msc Project\PhoneSploit-Pro-
WARN: Could not load icon
INFO: Texture: 720x1280
WARN: Demuxer 'audio': stream explicitly disabled by the device
```



```
    1. Tester Name                                        11. Open a Link on Device                       21. Access Device Shell

    2. List Connected Devices                             12. Display a Photo on Device                   22. List All Folders/Files

    3. Disconnect All Devices                             13. Play a Video on Device                      23. Copy WhatsApp Data
    4. Mirror & Control Device                            14. Get Device Information                      24. Copy All Screenshots

    5. Get Screenshot                                     15. Get Battery Information                     25. Copy All Camera Photos

    6. Screen Record                                      16. Restart Device                              26. Anonymous Screenshot
    7. Download File/Folder from Device                   17. Advanced Reboot Options                     27. Extract APK from Installed App

    8. Send File/Folder to Device                         18. Lock Device                                 28. Stop ADB Server

    9. Run an App                                         19. Uninstall an App                            29. Power Off Device
   10. Anonymous Screen Record                            20. List Installed Apps                         30. Create Final Report

 cls: Clear Screen                 e: Exit

[Main Menu] Enter selection > 2


* daemon not running; starting now at tcp:5037
* daemon started successfully
List of devices attached
3300d2cc222a1471       device product:j7eltexx model:SM_J700F device:j7elte transport_id:1



 cls: Clear Screen                 e: Exit
```

10. Conclusion and Future Enhancements

10.1 Conclusion: Ethical hacking, also known as penetration testing, involves testing computer systems, networks, or applications for security vulnerabilities to help identify and fix potential weaknesses. Ethical hacking and penetration testing are indispensable components of a robust cybersecurity strategy. They are integral elements of a comprehensive cybersecurity strategy, providing organizations with the tools and insights needed to fortify their digital defenses and respond effectively to the evolving threat landscape. They help organizations proactively identify and address vulnerabilities, protect sensitive data, comply with regulations, and continuously enhance their security posture in an ever-evolving threat landscape.

10.2 Future Enhancements: In future various extra features and modules can in be incorporated in this project with new software integration making it more robust and effective in testing the devices as well as can do penetration testing on cross operating system platforms like iOS, Mac, Windows, etc. These enhancements will further enhance the efficiency, security, and accessibility of the voting process, making it more inclusive and trustworthy

10.3 Bibliography:

https://github.com/AzeemIdrisi/PhoneSploit-Pro

https://www.hackerone.com/knowledge-center/7-pentesting-tools-you-must-know-about

https://www.csoonline.com/article/551957/11-penetration-testing-tools-the-pros-use.html

https://github.com/Genymobile/scrcpy

https://developer.android.com/tools/adb

https://www.geeksforgeeks.org/how-to-prepare-test-case-report-for-a-project/