**A**

### **Project Report**

### **On**

### **PortHawk : Advance Port Scanner**

**Submitted by**

### **Nakul Kishor Nimbekar**

**Roll No.: 24343**

**MCA–I**

**SEM–II**

**Under the guidance of**

### **Prof. Ashwini Mohite**

**For the Academic Year 2024-25**



***Sinhgad Technical Education Society’s***

**Sinhgad Institute of Management**

**Vadgaon Bk Pune 411041**

**(Affiliated to SPPU Pune & Approved by AICTE New Delhi)**

### 

### Date:

**CERTIFICATE**

This is to certify that Mr. Nakul Kishor Nimbekar has successfully completed his project work entitled **“PortHawk: Advance Port Scanner”** in partial fulfillment of MCA – I SEM –II Mini Project for the year 2024-2025. He has worked under our guidance and direction.

### Prof. Ashwini Mohite Dr. Chandrani Singh

**Project Guide Director, SIOM-MCA**

### Examiner 1 Examiner 2

**Date:**

**Place:** Pune

**DECLARATION**

I certify that the work contained in this report is original and has been done by me under the guidance of my guide.

* The work has not been submitted to any other Institute for any degree or diploma.
* I have followed the guidelines provided by the Institute in preparing the report.
* I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.
* Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the report and giving their details in the references.

### **Name and Signature of Project Team Members**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | **Seat No.** | **Name of students** | **Signature of students** |
| **1** | **24343** | **Nakul Kishor Nimbekar** |  |

**ACKNOWLEDGEMENT**

It is very difficult task to acknowledge all those who have been of tremendous help in this project. I would like to thank my respected guide **Prof. Ashwini Mohite** for providing me necessary facilities to complete my project and also for their guidance and encouragement in completing my project successfully without which it wouldn’t be possible. I wish to convey my special thanks and immeasurable feelings of gratitude towards **Dr. Chandrani Singh, Director, SIOM-MCA.** I wish to convey my special thanks to all teaching and non-teaching staff members of **Sinhgad Institute of Management, Pune** for their support.

Thank You

Yours Sincerely,

**Nakul Kishor Nimbekar**

# **INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Chapter** | **Page No.** |
| **1** | **CHAPTER 1: INTRODUCTION** | 4 |
| 1.1 | Abstract | 4 |
| 1.2 | Existing System and Need for System | 4 |
| 1.3 | Scope of System | 4 |
| 1.4 | Operating Environment Hardware and Software | 5 |
| 1.5 | Brief Description of Technology Used | 5 |
| **2** | **CHAPTER 2: PROPOSED SYSTEM** | 6 |
| 2.1 | Feasibility Study | 6 |
| 2.2 | Objectives of the proposed system | 7 |
| 2.3 | Users of the system | 7 |
| **3** | **CHAPTER 3: ANALYSIS AND DESIGN** | 8 |
| 3.1 | Class Diagram | 8 |
| 3.2 | Use Case Diagrams | 9 |
| 3.3 | Activity Diagram | 10 |
| 3.4 | Sequence Diagram | 11 |
| 3.5 | Sample Input and Output Screens  (Screens must have valid data. All reports must have at-least 5 valid records.) | 12 |
| **4** | **CHAPTER 4: CODING Sample code** | 15 |
| **5** | **CHAPTER 5: LIMITATIONS OF SYSTEM** | 18 |
| **6** | **CHAPTER 6: PROPOSED ENHANCEMENTS** | 19 |
| **7** | **CHAPTER 7: CONCLUSION** | 20 |
| **8** | **CHAPTER 8: BIBLIOGRAPHY** | 21 |

**CHAPTER 1: INTRODUCTION**

**1.1 Abstract: -**

PortHawk is an advanced network scanning tool designed to identify open ports, detect running services, and analyze network vulnerabilities. This cross-platform application combines the efficiency of Python's socket programming with a user-friendly GUI (Tkinter) to provide both technical and non-technical users with comprehensive network reconnaissance capabilities. The system implements multi-threading for rapid scans, protocol fingerprinting for service identification, and generates detailed security reports.

**1.2 Existing System and Need for System**

* **Existing System**
* Nmap: One of the most widely used network scanning tools. It provides advanced scanning options, including port scanning, service detection, and banner grabbing.
* Netcat
* Angry IP Scanner
* **Need for System**
* Simplicity for Basic Users : Users will just need to input the IP address of the target system, click a button, and view the results in an easy-to-read format.
* Customizable Output: allow users to export scan results into various formats (e.g., CSV, JSON, text)
* User-Friendly Interface

**1.3 Scope and Objectives of System**

* Develop a port scanner that scans all 65,535 ports.
* Implement multithreading to speed up the scanning process.
* Identify services running on open ports and grab service banners.
* Provide a basic GUI to make the tool user-friendly, allowing easy start/stop functionality and result display.
* Allow users to save the scan results for future reference

**1.4 Operating Environment Hardware and Software:**

|  |  |
| --- | --- |
| **Software Requirement** | **Hardware Requirement** |
| Operating System: Windows 7 or above | Processor: Intel Core i3 or above |
| RAM: 2GB or above |
| HDD: 512 GB or above |

**1.5 Brief Description of Technology Used**

**Core Technologies:**

1. Frontend:
   * Tkinter (Python GUI)
   * ttk Bootstrap themes
2. Backend:
   * Python Socket API (TCP scanning)
   * ThreadPoolExecutor (concurrent scanning)
   * Requests (HTTP analysis)
3. Security Features:
   * Protocol fingerprinting (SSH/HTTP/SMTP)
   * Malware port detection (31337/4444 etc.)
   * Rate limiting (100 pkt/sec default)

**Innovative Components:**

* Hybrid service detection (Nmap+system+banners)
* Adaptive scanning (auto-adjusts based on network latency)
* Portable executable (PyInstaller) with antivirus compatibility

**CHAPTER 2: PROPOSED SYSTEM**

**2.1 Feasibility Study**

* **Technical Feasibility :**
* The project is technically feasible because:
* Programming Language: Python is widely used for cybersecurity and networking tasks.
* Libraries Used: The socket library is built into Python, eliminating the need for external dependencies.
* GUI Development: Tkinter is lightweight and easy to implement for basic user interactions.
* Multithreading: Python supports multithreading to scan multiple ports efficiently.
* Cross-Platform Compatibility: The scanner can run on Windows, Linux, and macOS.
* Deployment: PyInstaller enables the creation of standalone executables (.exe, .app, etc.), so users don’t need to install Python separately.
* **Economic Feasibility :**
* Open-Source Tools: Python, Tkinter, and the socket library are free to use.
* No Hardware Investment: The tool can run on any standard computer without requiring special hardware.
* Low Development Cost: The project does not require expensive licenses or third-party services.
* Easy Maintenance: Since the code is lightweight and modular, maintenance costs are minimal.
* **Operational Feasibility :**
* Minimal User Interaction: The user only needs to start the scan, and the tool automatically detects open ports.
* Fast Execution: Multithreading ensures efficient scanning without long delays.
* User-Friendly Interface: A basic GUI (Tkinter) makes it easy for non-technical users.
* Security Awareness: IT professionals, system administrators, and ethical hackers can use it to check network security.

**2.2 Objectives of the proposed system:**

1. Efficient Scanning – Quickly detect open ports (1-65535) with high accuracy.
2. Service Identification – Recognize running services (HTTP, SSH, FTP, etc.) using banners and protocols.
3. User-Friendly GUI – Simple interface for both beginners and experts.
4. Security Insights – Flag risky ports (e.g., 31337 for Backdoor) and provide warnings.
5. Cross-Platform – Works on Windows, Linux, and macOS.
6. Export Reports – Save results in CSV/JSON/TXT for analysis.
7. Lightweight – Low CPU/memory usage for smooth performance

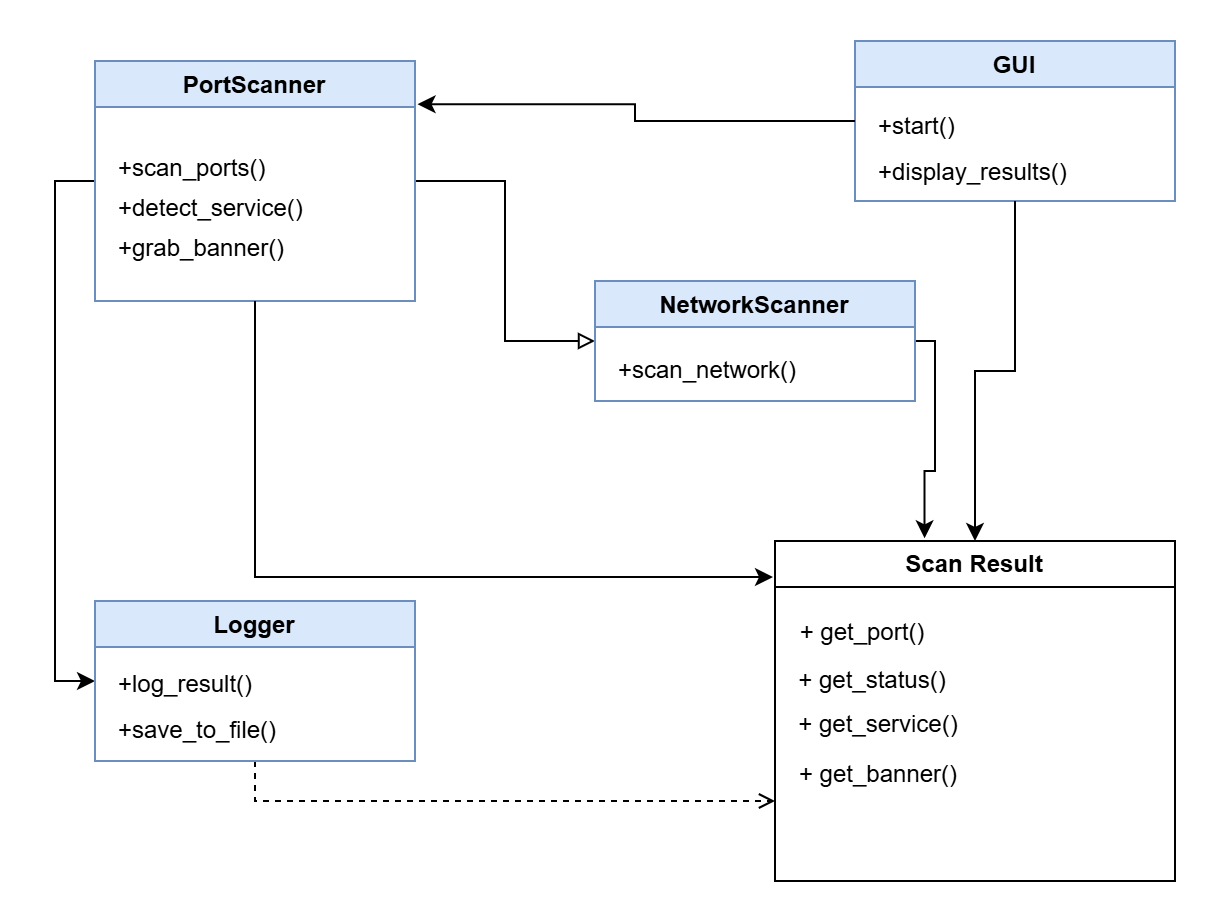
**2.3 Users of the system**

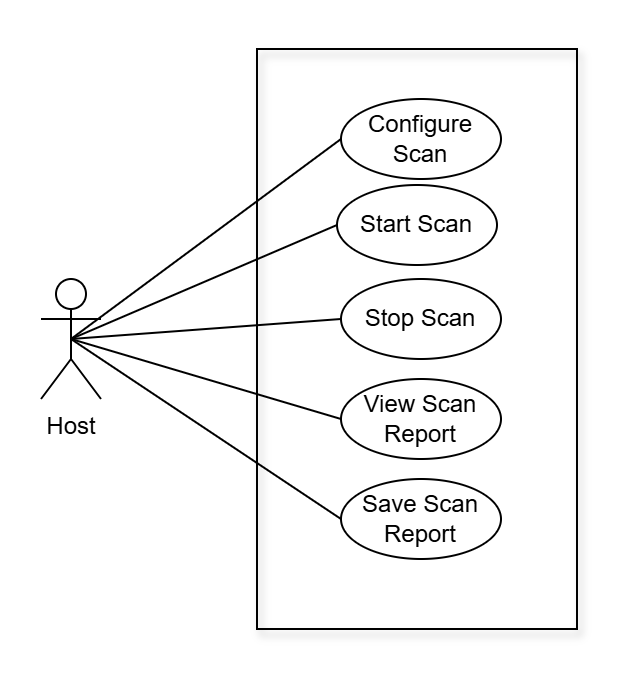
PortHawk is designed for a wide range of users with varying technical expertise:

1. Network Administrators
   * Monitor open ports and services on corporate networks.
   * Identify unauthorized or vulnerable services.
2. Security Professionals & Ethical Hackers
   * Perform penetration testing and vulnerability assessments.
   * Detect misconfigured firewalls or exposed services.
3. IT Students & Educators
   * Learn about networking, ports, and security in a practical way.
   * Use as a teaching tool for cybersecurity courses.
4. Software Developers
   * Test application connectivity and firewall rules.
   * Debug network-related issues in development environments.
5. Home Users & Tech Enthusiasts
   * Check home network security for exposed devices (routers, IoT).
   * Understand which services are running on their systems.

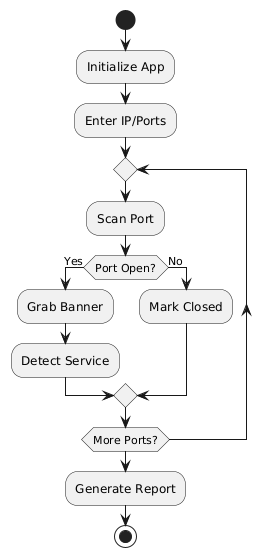
**CHAPTER 3: ANALYSIS AND DESIGN**

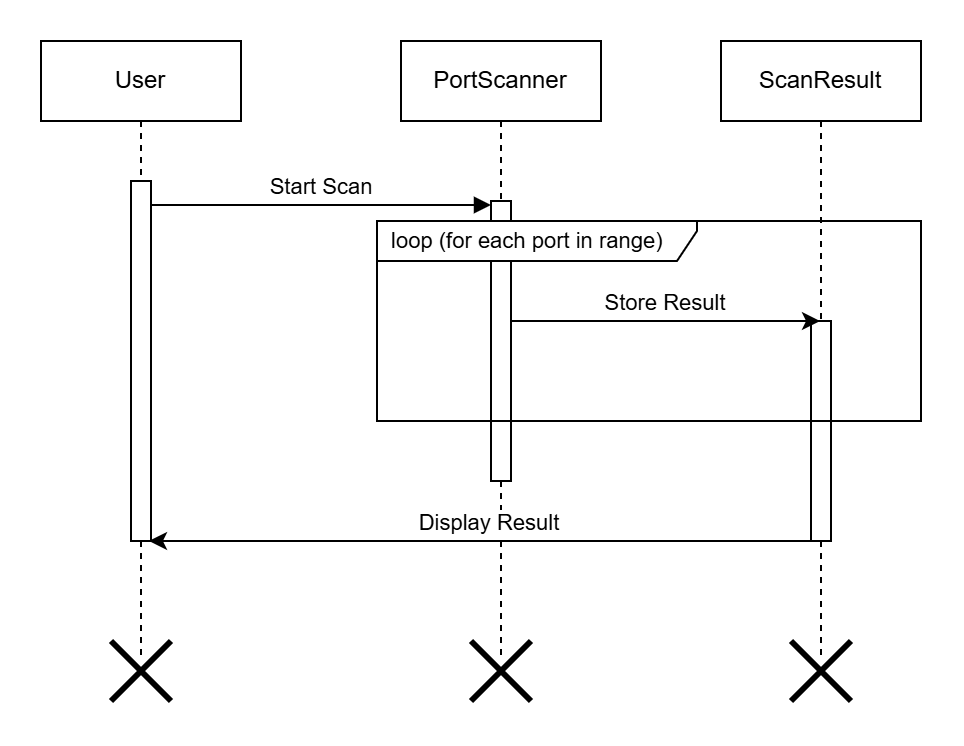
**3.1 Class Diagram**



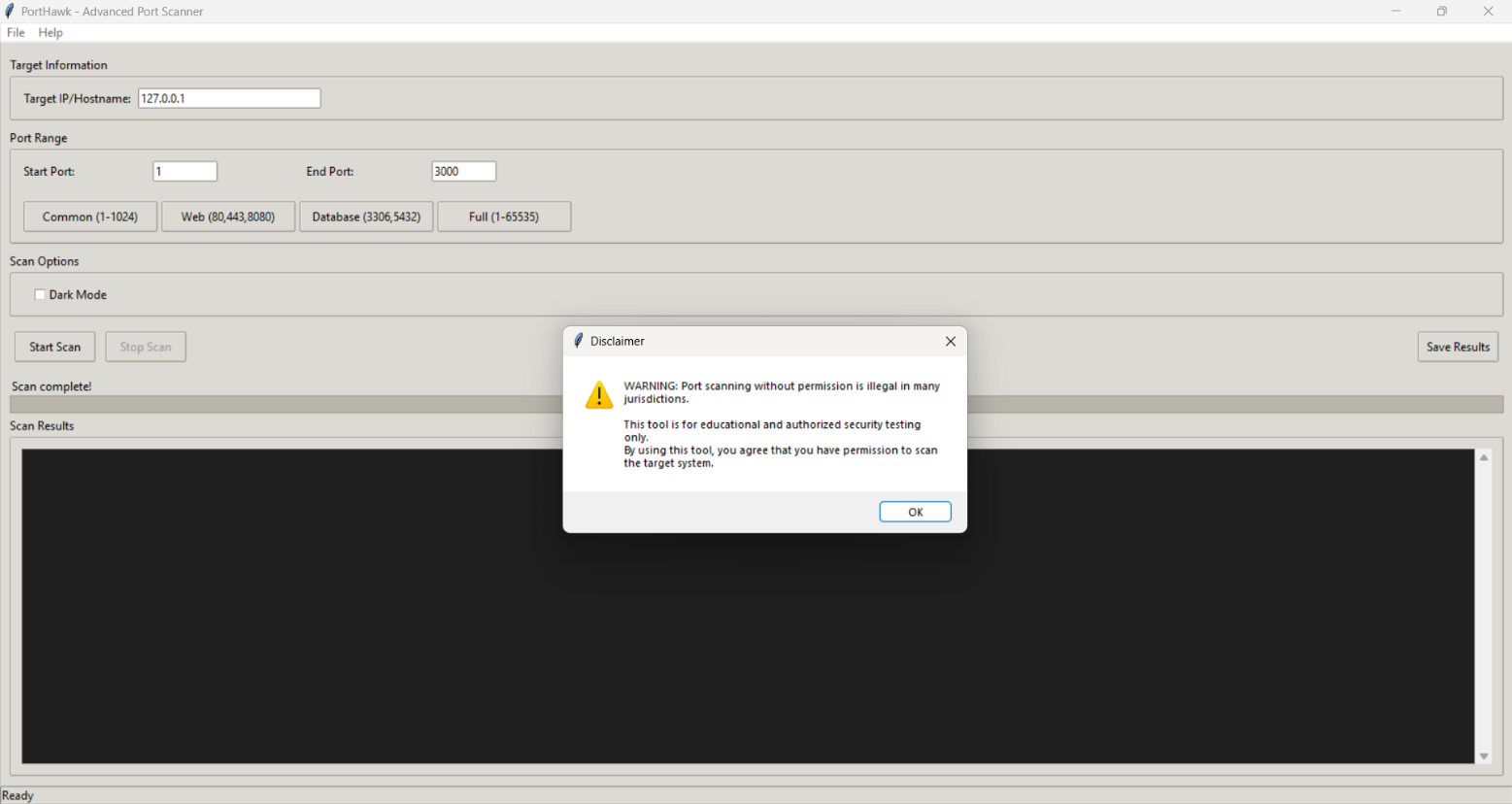
**3.2 Use Case Diagrams**

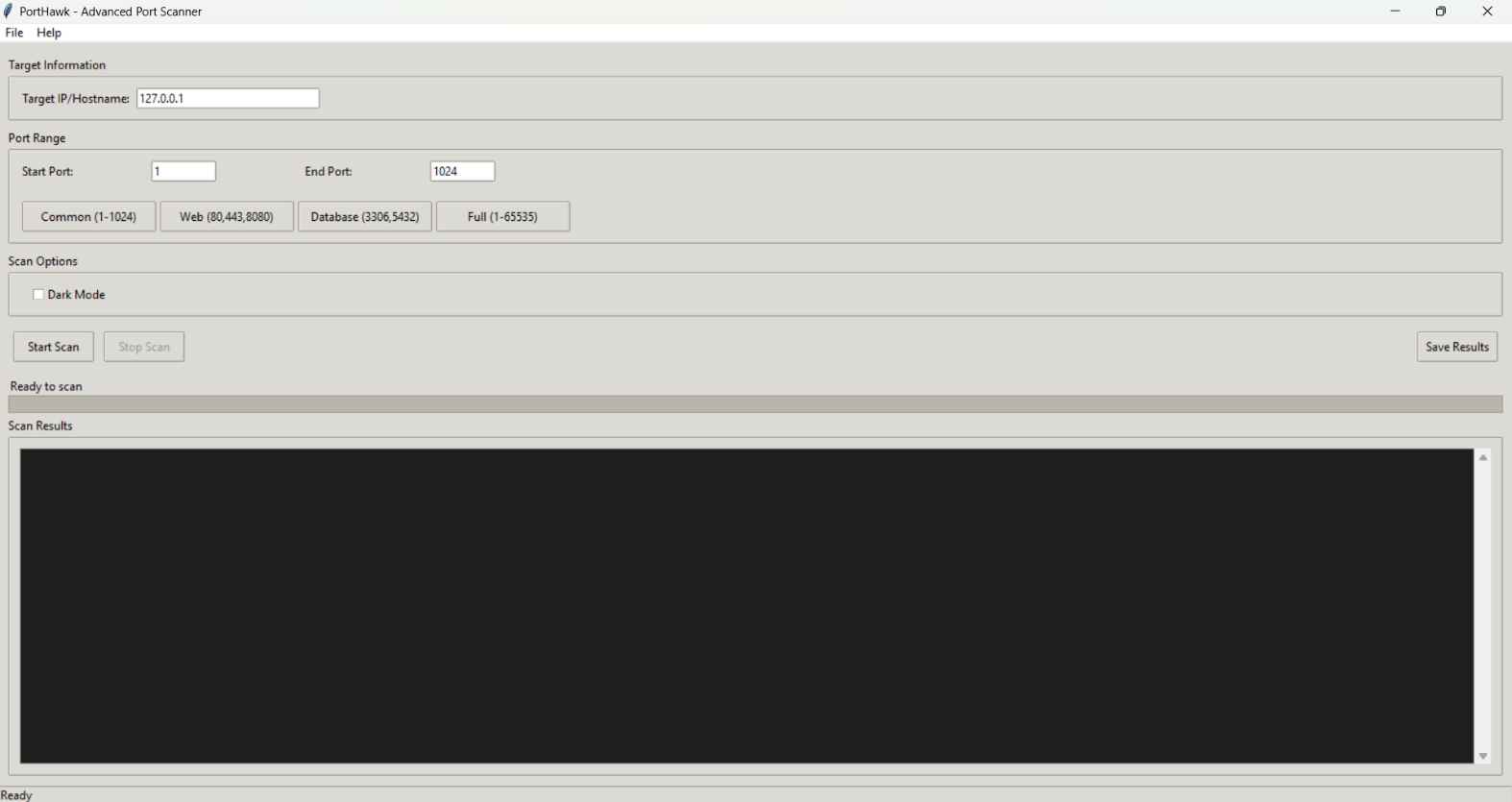
**3.3 Activity Diagram**

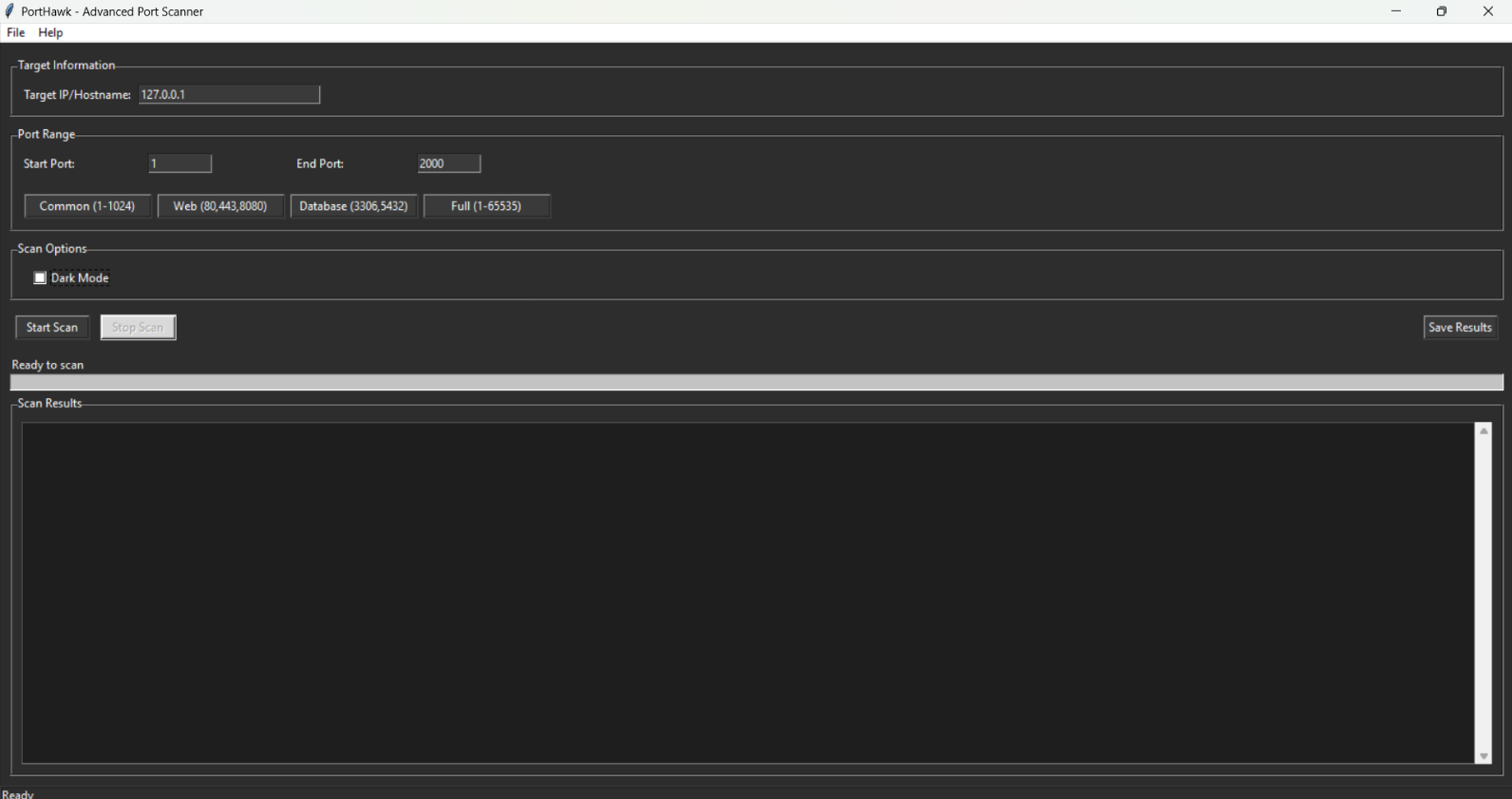


**3.4 Sequence Diagram**

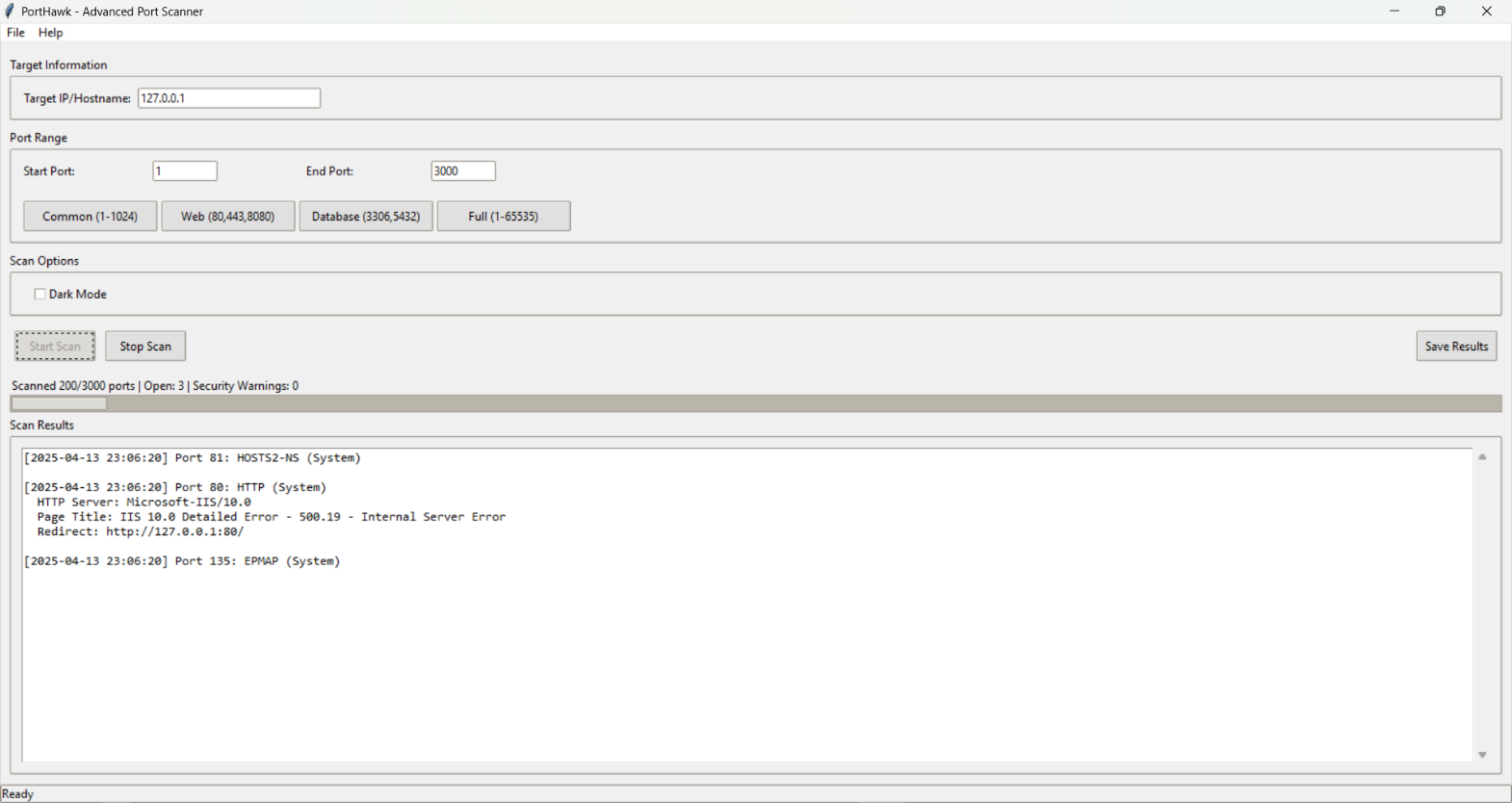
**3.5 Sample Input and Output Screens**

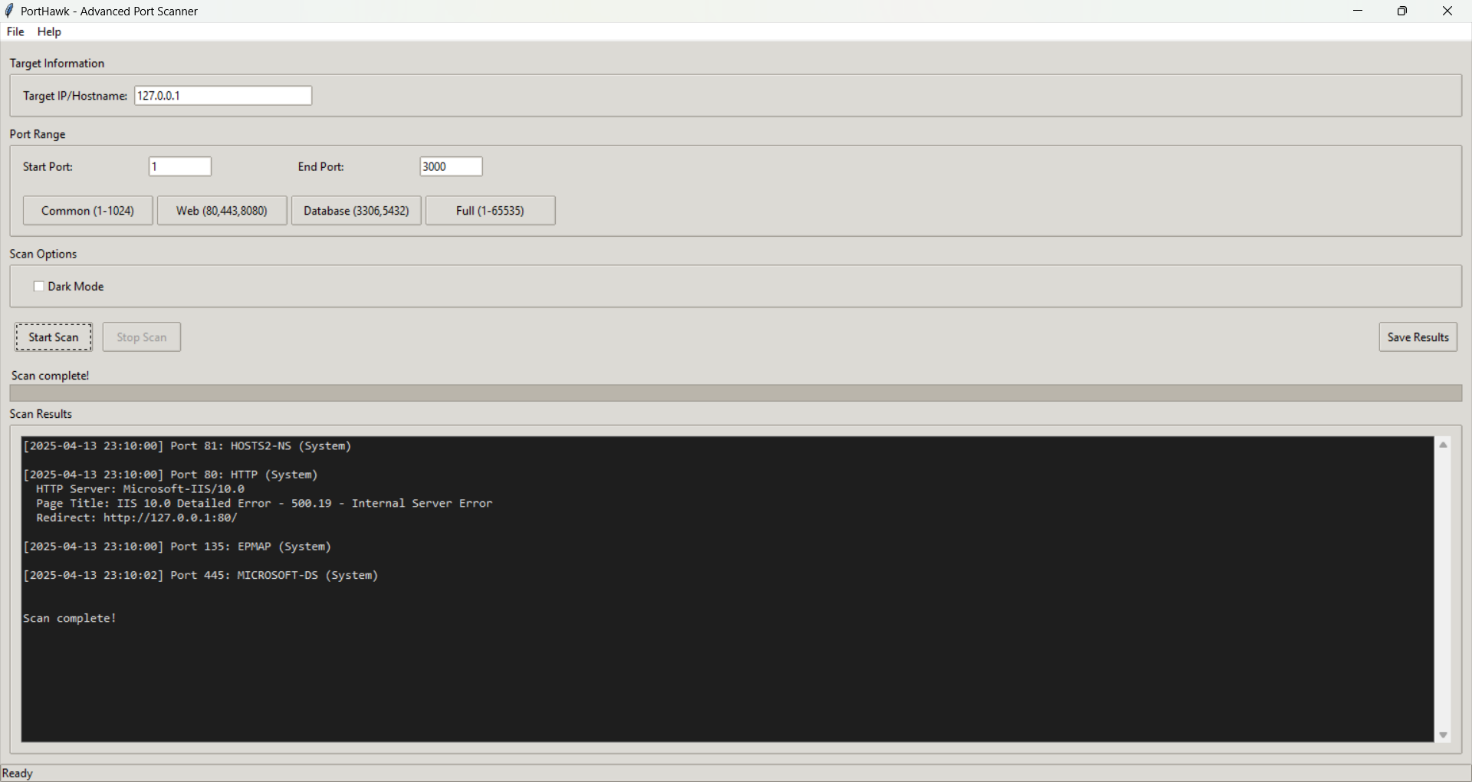
** Disclaimer:**

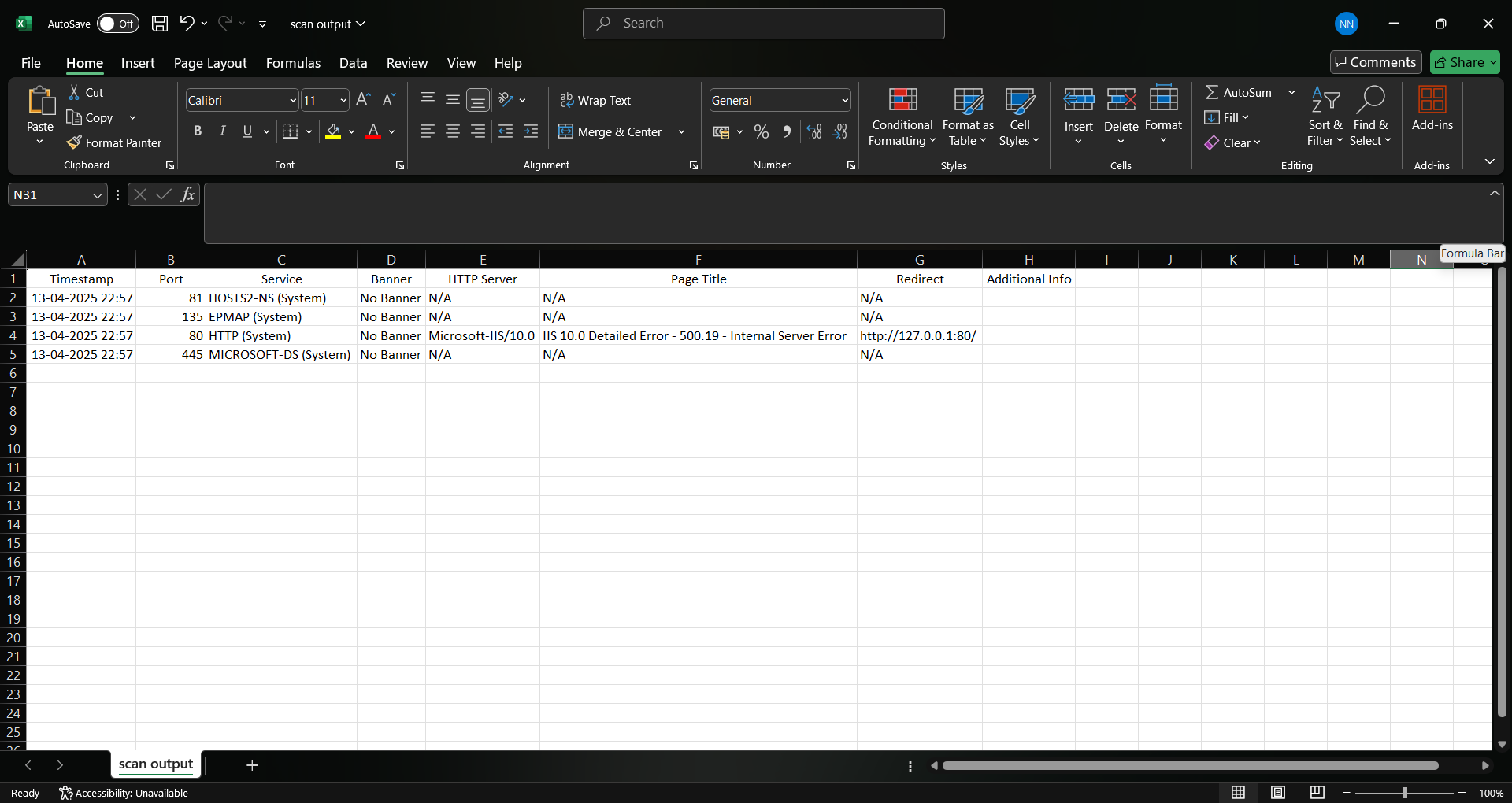
**First View :**

**Dark Mode :**

**Scanning :**

****

**Scan Complete :**

**Output :**

**CHAPTER 4: CODING Sample code**

**import socket**

**import threading**

**import datetime**

**import tkinter as tk**

**from tkinter import scrolledtext, filedialog, messagebox, ttk**

**import csv**

**import requests**

**from concurrent.futures import ThreadPoolExecutor, as\_completed**

**import json**

**import random**

**import time**

**from tkinter import font as tkfont**

**import sys**

**if sys.executable.endswith(".exe"):**

**sys.stdout = open("output.log", "a")**

**sys.stderr = sys.stdout**

**log\_file = "scan\_results.txt"**

**scan\_results = []**

**stop\_scan = False**

**dark\_mode = False**

**current\_theme = {}**

**nmap\_services = {}**

**extended\_services = {**

**80: "HTTP", 443: "HTTPS",**

**21: "FTP", 22: "SSH", 23: "Telnet",**

**25: "SMTP", 53: "DNS", 110: "POP3",**

**143: "IMAP", 3306: "MySQL", 3389: "RDP",**

**8080: "HTTP-Alt", 8443: "HTTPS-Alt",**

**5900: "VNC", 1723: "PPTP", 500: "ISAKMP",**

**161: "SNMP", 445: "SMB", 587: "SMTP-SSL",**

**465: "SMTPS", 993: "IMAPS", 995: "POP3S"**

**}**

**def identify\_unknown\_port(ip, port):**

**"""Enhanced identification for unknown ports"""**

**results = []**

**if 32768 <= port <= 60999:**

**results.append("Likely ephemeral port (Linux default range)")**

**elif 49152 <= port <= 65535:**

**results.append("Likely ephemeral port (Windows/IANA range)")**

**banner = grab\_banner(ip, port)**

**if banner != "No Banner":**

**results.append(f"Banner: {banner[:200]}{'...' if len(banner) > 200 else ''}")**

**protocol = detect\_protocol(ip, port)**

**if protocol:**

**results.append(f"Protocol hints: {protocol}")**

**malware\_check = check\_malware\_ports(port)**

**if malware\_check:**

**results.append(f"Security note: {malware\_check}")**

**return " | ".join(results) if results else "No additional info"**

**def detect\_protocol(ip, port):**

**"""Attempt to identify protocol through behavior"""**

**try:**

**with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as s:**

**s.settimeout(1)**

**s.connect((ip, port))**

**s.send(b"GET / HTTP/1.0\r\n\r\n")**

**if b"HTTP/" in s.recv(1024):**

**return "HTTP"**

**s.send(b"INFO\r\n")**

**if b"redis\_version" in s.recv(1024):**

**return "Redis"**

**return None**

**except:**

**return None**

**def check\_malware\_ports(port):**

**"""Check against known malware ports"""**

**malware\_ports = {**

**31337: "Back Orifice",**

**4444: "Metasploit",**

**6660: "IRC (often malware)",**

**12345: "NetBus"**

**}**

**return malware\_ports.get(port)**

**def load\_services\_from\_nmap(file\_path="nmap-services"):**

**"""Load service mappings from nmap-services file"""**

**services\_dict = {}**

**try:**

**with open(file\_path, "r") as file:**

**for line in file:**

**if line.startswith("#") or not line.strip():**

**continue**

**parts = line.split()**

**if len(parts) >= 2:**

**service\_name = parts[0]**

**port\_proto = parts[1]**

**if "/tcp" in port\_proto:**

**port = int(port\_proto.split("/")[0])**

**services\_dict[port] = service\_name.upper()**

**print(f"[+] Loaded {len(services\_dict)} services from nmap-services")**

**except Exception as e:**

**print(f"[-] Error loading nmap-services: {e}")**

**return services\_dict**

**def get\_service\_name(port):**

**"""Get service name from multiple sources with priority"""**

**# Try system first**

**try:**

**return f"{socket.getservbyport(port, 'tcp').upper()} (System)"**

**except:**

**pass**

**if port in nmap\_services:**

**return f"{nmap\_services[port]} (Nmap)"**

**return extended\_services.get(port, f"Unknown (Port {port})")**

**CHAPTER 5: LIMITATIONS OF SYSTEM**

**1. Scanning Capabilities**

* Protocol Support:
  + Only supports TCP port scanning (no UDP or ICMP detection)
  + Cannot perform SYN stealth scans (requires admin privileges)
* Accuracy Issues:
  + May miss non-standard services running on common ports (e.g., SSH on port 2222)
  + Banner grabbing can be unreliable if services hide version info

**2. Performance Constraints**

* Scan Speed:
  + Full 65,535-port scans take 15-30 minutes depending on network latency
  + Limited to 200 concurrent threads to prevent system overload
* Resource Usage:
  + High CPU usage during active scans (up to 80% on older systems)
  + Not optimized for large-scale network scanning (e.g., entire subnets)

**3. Security Limitations**

* Firewall/NAT Issues:
  + Blocked by enterprise firewalls and IPS systems
  + Cannot scan through NAT gateways without port forwarding
* No Vulnerability Detection:
  + Only identifies open ports/services (does not check for exploits)
  + Cannot detect misconfigurations like weak encryption

**CHAPTER 6: PROPOSED ENHANCEMENTS**

**1. Basic Feature Upgrades**

* UDP port scanning (simple timeout-based detection)
* Common vulnerability checks (e.g., weak SSH/HTTP headers)
* Export reports in PDF (basic template, no LaTeX)

**2. Performance Improvements**

* Port grouping (scan ranges like 80-100 instead of one-by-one)
* Skip recently scanned ports (cache for 1 hour)
* Adjustable timeout (1s-5s slider in GUI)

**3. Usability Additions**

* Simple dashboard (open/closed port counters, risk level indicator)
* Scan presets (Quick/Full/Custom profiles)
* Tooltips (explain ports/services in simple terms)

**4. Stability Fixes**

* Better antivirus handling (signed binaries, fewer false positives)
* Network recovery (auto-retry failed connections)
* Memory management (fix leaks during long scans)

**CHAPTER 7: CONCLUSION**

PortHawk provides a practical and accessible solution for basic network port scanning, catering to users who need simplicity without advanced technical overhead. By focusing on core functionalities like TCP port detection, service identification, and user-friendly reporting, the tool effectively serves students, IT enthusiasts, and home users seeking to monitor their network security. While its current scope excludes advanced features like UDP scanning or vulnerability assessments, the foundation is solid for future expansions. The GUI-driven approach lowers the learning curve, making network diagnostics approachable for non-experts. PortHawk’s cross-platform compatibility and lightweight design further enhance its utility as a quick audit tool. With planned enhancements such as PDF reporting and performance optimizations, the system has the potential to grow into a more comprehensive solution. This project underscores the importance of balancing usability with functionality, demonstrating that even streamlined tools can deliver meaningful security insights.

**CHAPTER 8: BIBLIOGRAPHY**

**Books**

1. *Network Basics for Beginners* – by Alex Carter (2021).
2. *Python for Networking* – by Sam Wilson (2020).

**YouTube Channels**

1. **NetworkChuck** – "Port Scanning Explained" (2022)
2. **The Cyber Mentor** – "Ethical Hacking: Network Scanning" (2021).

**Websites & Tools**

1. **Nmap Official Documentation** – Basic Port Scanning Guide.
2. **Python’s**socket**Programming Guide** – Official Python Docs.

**Other Sources**

1. **W3Schools – Python Networking Tutorials** (2023).