



Kristu Jayanti College

AUTONOMOUS

Bengaluru

Reaccredited 'A++' Grade by NAAC | Affiliated to Bengaluru North University

IV SEMESTER PROJECT REPORT ON

“CyberSafe Tools”

Submitted in partial fulfillment of the requirements for the award of the degree of

MASTER OF SCIENCE- DATA SCIENCE

Work carried out at

Shamgar Software Solutions

During the Academic Year 2024 - 2025

Submitted By

Aashly K Azeez

23MDTS01

Under the guidance of

Internal Guide

Dr. Ayshwarya B

Assistant Professor

Department of Computer Science (PG)

External Guide

Dr. Uttam Mande

Solutions Architect

Shamgar Software Solutions

DEPARTMENT OF COMPUTER SCIENCE (PG)

KRISTU JAYANTI COLLEGE (Autonomous)

K. Narayanapura, Kothanur P.O., Bengaluru – 560077



Kristu Jayanti College

AUTONOMOUS

Bengaluru

Reaccredited 'A++' Grade by NAAC | Affiliated to Bengaluru North University

Certificate

This is to certify that the project entitled “**CyberSafe Tools**” has been satisfactorily completed by **AASHLY K AZEEZ, 23MDTS01** in partial fulfillment of the requirements of **IV Semester M.Sc Data Science** Programme prescribed by Kristu Jayanti College (Autonomous) Bangalore (Affiliated to Bangalore North University) during the academic year 2024-25.

Internal Guide

Head of the Department

Valued by Examiners

1: _____

Centre: Kristu Jayanti College

2: _____

Date:



Cert num:- 23069



SHAMGAR
SOFTWARE SOLUTIONS

CERTIFICATE OF INTERNSHIP

On

CYBERSECURITY ANALYST INTERN

This is to certify that

Aashly K Azeez

with Internship ID: INT23069 from Kristu Jayanti College

has successfully completed an internship at Shamgar
Software Solutions from

from 03 Feb 2025 to 03 Jun 2025

Uttam Mande

Course Director
Uttam Mande



Divya Matta

Project Director
Divya Matta



<https://shamgarsoft.com>

mail:shamgarsoft@gmail.com



ACKNOWLEDGEMENT

First and foremost, I express my gratitude to the Almighty God for bestowing me with the fortitude, endurance, and discernment necessary, from the inception to the completion of this project effectively.

Undoubtedly, this project would not have reached the finish line, without his blessing and guidance.

Then, I want to offer my sincere gratitude and appreciation to everyone who has helped me through the journey of finishing this project. I am deeply grateful for every ounce of their effort.

For their unwavering support and for creating a supportive learning atmosphere for us, I would like to express my profound gratitude to our distinguished Principal Rev. **Dr. Fr. Augustine George** and Vice Principal Rev. **Dr. Fr. Lijo P Thomas**. Our academic experience has been significantly shaped by their remarkable leadership and dedication to education.

For the many valuable insights and expertise provided by the Head, Department of Computer Science (PG) **Dr. Kumar R**, I remain thankful for his guidance and encouragement that have been the guiding light paving way for the shaping and scope of this project.

I would like to thank **Dr. Murugantham A** program Coordinator for his constant support and encouragement.

My project mentor, **Dr. Ayshwarya B**, deserves a special note of thanks. I am incredibly appreciative of his direction, patience, and unwavering support throughout the endeavor. The accomplishment of this project has been made possible by his knowledge, prompt remarks, and unique ideas.

The tireless support and motivation from my dearest friends and classmates, I express my heartfelt gratitude. Again, to all those who have travelled with me, directly or indirectly, this project would have breathed life. Hence, I express gratitude and appreciation, for their support has been invaluable, and I am truly grateful.

SYNOPSIS

Aim

The primary aim of the **CyberSafe Tools** project is to promote cybersecurity awareness by demonstrating how common digital threats function and how individuals can protect themselves from such attacks. This project is designed to educate users—especially students and beginners—about the risks associated with weak passwords, phishing attempts, and keylogging, using simple and interactive simulations.

Instead of just reading about cyber threats, users gain practical understanding through hands-on experiences that reflect real-world scenarios in a safe and controlled environment. The project encourages the adoption of safe online habits and aims to equip users with the knowledge and skills needed to recognize risks, safeguard their data, and stay secure in the digital world.

.

Introduction

The **CyberSafe Tools** project represents a practical step toward improving cybersecurity awareness through interactive education. Developed as a toolkit for learning and demonstration purposes, it includes essential modules that simulate real-world cyber threats in a controlled and ethical environment. The primary focus is to help users understand how attackers exploit weak passwords, phishing techniques, and keylogging tools, while also educating them on preventive measures.

By combining theoretical knowledge with hands-on interaction, this project aims to simplify the learning process for users—especially students and beginners—and make cybersecurity concepts more approachable. Ultimately, **CyberSafe Tools** serves as a user-friendly platform to promote safe digital practices and help individuals become more informed and responsible in the online world.

.

Proposed System

The proposed **CyberSafe Tools** system is a web-based cybersecurity education platform designed to provide users with an interactive and practical learning experience. It integrates several essential cybersecurity tools into a single, cohesive application aimed at enhancing awareness and promoting safe online behavior. The system includes modules such as a port scanner to identify open network ports, a password strength checker to evaluate the robustness of user-entered passwords, a hash generator to demonstrate how data can be encrypted using cryptographic algorithms like MD5, SHA1, and SHA256, a vulnerability scanner to detect missing HTTP security headers on websites, and a password generator to help users create strong, secure passwords. Each module is designed to be simple, intuitive, and informative, making it suitable for students, developers, and anyone new to the field of cybersecurity. To support learning retention and practical usage, CyberSafe Tools also allows users to download detailed reports of their interactions with each tool, including all prior entries. This feature not only aids in documentation but also reinforces understanding through review. By merging theoretical concepts with real-world applications, CyberSafe Tools bridges the gap between classroom learning and practical implementation, thereby fostering better cybersecurity practices and digital literacy among users.

Modules

The CyberSafe Tools project is built around a collection of interactive, cybersecurity-focused modules, each of which simulates a specific aspect of cyber threats and defenses. These modules are designed to offer hands-on learning experiences in a safe, controlled environment, helping users—particularly students and beginners—gain valuable insights into common vulnerabilities and how to mitigate them. Each module encapsulates a real-world security concern and presents it in a user-friendly, educational format. The goal is to bridge the gap between theoretical cybersecurity concepts and practical implementation.

Below is a detailed overview of each module:

Password Strength Checker

This module enables users to input a password and instantly receive feedback on its strength. The system evaluates the password based on multiple parameters including:

- Length
- Inclusion of uppercase and lowercase letters
- Use of digits and special characters
- Pattern repetition or predictability
- The feedback includes a rating (e.g., Weak, Moderate, Strong) along with actionable suggestions to improve password quality. By visually demonstrating how weak passwords can be exploited, this module helps users build stronger and safer password habits.

Port Scanner

Network ports are often targeted by attackers seeking unauthorized access to a system. This module allows users to specify a host (IP address or domain) and a port range to scan. The scanner uses the TCP protocol to:

- Identify which ports are open
- Detect potential vulnerabilities in exposed services
- The tool provides educational value by helping users understand the importance of firewall configurations and minimizing unnecessary open ports. It encourages network hygiene and helps illustrate the attacker's perspective when probing networks.

Hash Generator

Hashing is a foundational concept in data integrity and secure password storage. This module allows users to convert plaintext input into cryptographic hash values using popular algorithms such as:

- MD5
- SHA1
- SHA256
- By showing the irreversible transformation from input text to fixed-size hashes, users gain a practical understanding of how hashes work, where they're used (e.g., password storage, digital signatures), and the differences between various algorithms.

Vulnerability Scanner

This module provides a basic website security audit by analyzing HTTP response headers for common security misconfigurations. It checks for the presence or absence of important headers such as:

- X-Frame-Options
- Content-Security-Policy
- Strict-Transport-Security
- X-XSS-Protection
- These headers are crucial for defending against web-based attacks like clickjacking, cross-site scripting (XSS), and protocol downgrade attacks. The scanner educates users on the role of HTTP headers in web security and highlights potential improvements for safer web deployment.

Password Generator

To support users in creating strong, unpredictable credentials, this module generates random passwords based on configurable parameters such as:

- Desired length
- Inclusion of uppercase, lowercase, numbers, and special characters
- This tool helps reinforce best practices by providing examples of complex passwords that resist brute-force and dictionary attacks. It is especially useful for users who struggle to create secure passwords manually.

Report Download Feature

Every tool in the application includes an option to download the results in a readable format (e.g., PDF or text file). This feature allows users to:

- Keep a personal record of test inputs and results
- Use the reports for academic or demonstration purposes
- Review security feedback for deeper learning
- The download functionality adds value by supporting documentation, offline study, and progress tracking. It also makes the toolkit more useful in classroom or workshop settings.

CONTENTS

S.No	Topic	Page No
1.	Introduction	1
1.1	About the Organization	2
1.2	Problem Definition	3
2	Project Description	4
3	System Study	6
3.1	Existing System	6
3.2	Proposed System	7
3.3	DFD, ER Diagram	9,10
3.4	UML Diagrams (Use Case, Activity, Class Diagram,	11,12
3.5	Feasibility Study	13
4	System Configuration	16
4.1	Hardware Requirements	16
4.2	Software Requirements	17
5	Details of Software	23
5.1	Overview of Frontend	25
5.2	Overview of Backend	27
5.3	About the Platform	31
6	System Design	33
6.1	Architectural Design	37
6.2	Input / Output Design	40
7	Testing	44
8	Implementation	47
9	Conclusion and Future Enhancement	50
10	Bibliography	55
11	APPENDICES A-Table Structure	61
12	APPENDICES B-Screenshots	
13	APPENDICES C-Sample Report of test cases	
14	Supporting Information	

INTRODUCTION

1. INTRODUCTION

Cybersecurity awareness has become increasingly critical in today's digital age, where individuals are continuously exposed to threats such as weak passwords, unpatched vulnerabilities, phishing attempts, and malicious software. With the rise in cybercrimes targeting both personal users and large organizations, it is essential to build digital literacy and equip users with tools to understand and combat these risks effectively. In response to this growing need, the **CyberSafe Tools** project was developed as part of the Master of Science in Data Science academic curriculum. The primary objective of the project is to promote cybersecurity knowledge through interactive learning, real-time analysis, and hands-on demonstrations. Unlike traditional theory-based approaches, this toolkit leverages practical engagement to teach users about cyber threats and defense mechanisms.

The CyberSafe Tools application integrates multiple cybersecurity modules into a single, user-friendly platform built using **Flask** and **HTML/CSS/JavaScript**. These tools are designed to simulate real-world cyber threat scenarios in a controlled environment, helping users—especially students and beginners—gain insight into how attacks work and how to respond effectively.

The **Password Strength Checker** module enables users to test the robustness of their passwords based on length, character variety, and complexity, while offering actionable feedback. The **Port Scanner** helps users understand how open ports can expose systems to threats, by scanning specified ranges on a target host. The **Hash Generator** demonstrates how text is converted into cryptographic hashes such as MD5, SHA1, and SHA256, reinforcing the importance of data integrity and password storage practices.

Additionally, the **Vulnerability Scanner** examines websites for missing security headers, raising awareness about HTTP-based vulnerabilities. The **Password Generator** provides users with strong, randomly generated passwords to encourage secure credential practices. Each tool includes a **report download feature**, allowing users to review and document their entries and results.

Through these modules, the CyberSafe Tools project aims to bridge the gap between theoretical knowledge and practical cybersecurity application. It empowers users with both awareness and experience, promoting safer digital behavior and better preparedness in facing cybersecurity threats.

1.1 ABOUT THE ORGANIZATION

Shamgar Software Solutions, a forward-thinking IT services and consulting company, is dedicated to empowering businesses and individuals through innovative technological solutions. Renowned for its commitment to delivering high-quality software products and services, Shamgar operates with a vision to bridge the gap between digital potential and practical application, especially in the fields of cybersecurity, automation, and enterprise solutions.

Founded with the mission of providing reliable and customized software services, Shamgar Software Solutions has steadily built a reputation for excellence by working closely with clients across sectors, including education, finance, healthcare, and government. The company is driven by a strong belief in continuous learning, client-focused development, and the strategic use of modern technologies to solve real-world problems.

One of Shamgar's notable initiatives is its collaboration with emerging professionals and student developers, particularly those from postgraduate programs. By involving students in live projects, the company offers valuable industry exposure while also fostering a culture of mentorship, innovation, and hands-on learning. This initiative enables aspiring developers to work on real-life applications like phishing detection systems, file encryption tools, and web-based platforms under professional guidance. Shamgar Software Solutions emphasizes secure software practices and has made significant strides in developing tools that align with contemporary cybersecurity standards. The organization believes in developing modular, scalable, and user-friendly applications that not only meet client requirements but also push the boundaries of what's possible in everyday digital operations.

With a growing portfolio of successful projects and a team passionate about technology and integrity, Shamgar Software Solutions continues to pave the way for impactful software development. Through projects such as the Email Phishing Detection Tool and the File Encryption-Decryption Utility, the organization exemplifies how practical, security-focused software can contribute meaningfully to both industry needs and academic advancement.

1.2 PROBLEM DEFINITION

The rapid growth of cyber threats—including phishing attacks, weak password exploitation, open port vulnerabilities, and insecure data handling—has exposed a serious gap in cybersecurity awareness, especially among students, non-technical users, and beginners entering the digital and IT space. While sophisticated tools and professional solutions are available for enterprises, there remains a lack of accessible platforms that allow individuals to explore and understand the workings of these threats through practical, hands-on interaction.

The **CyberSafe Tools** project was conceptualized to address this gap by providing an educational, simulation-based toolkit that presents core cybersecurity concepts in an interactive and beginner-friendly format. Existing awareness programs typically rely on passive learning methods like articles, presentations, or awareness campaigns, which often lack interactivity and fail to give users a real sense of how vulnerabilities can be identified or exploited. This shortfall reduces the effectiveness of cybersecurity training, particularly in academic or entry-level environments.

CyberSafe Tools bridges this divide by integrating multiple modules—such as a **Password Strength Checker**, **Port Scanner**, **Hash Generator**, **Password Generator**, and **Vulnerability Scanner**—into one unified platform. Each module is designed to simulate real-world cybersecurity practices in a safe, controlled environment, allowing users to understand how threats manifest and how they can be mitigated. Additionally, the application supports **report downloading**, enabling users to keep records of their actions and learnings for review or academic documentation.

The primary problem tackled by this project is the lack of engaging, hands-on cybersecurity education platforms that are accessible to all. By offering interactive tools and simulations built with a simple user interface, CyberSafe Tools empowers users to experiment, learn, and strengthen their digital safety skills without needing a deep technical background.

In essence, the project seeks to create a meaningful educational experience that enhances awareness, encourages safer digital habits, and contributes to grassroots-level cyber resilience through a practical and approachable toolkit.

.

.

PROJECT DESCRIPTION

2. PROJECT DESCRIPTION

In the ever-evolving digital landscape, cyber threats have become more sophisticated and frequent, impacting users across all levels of technical expertise. From password leaks and network vulnerabilities to insecure websites and poor credential practices, the modern user faces a growing number of risks online. Despite the availability of technical resources, there remains a significant gap in cybersecurity awareness, particularly among students and non-technical individuals. Recognizing this challenge, the **CyberSafe Tools** project was developed as a hands-on educational platform under the Master of Science in Data Science program. The project aims to bridge the gap between theoretical cybersecurity concepts and practical, real-world applications.

CyberSafe Tools is a lightweight, modular web application developed using Python and the Flask framework. It combines ease of use with technical depth, offering users the ability to explore fundamental cybersecurity tools interactively. The project emphasizes practical learning by simulating how attackers may exploit vulnerabilities and how users can protect themselves by adopting safe digital practices. Unlike static presentations or text-heavy tutorials, this platform encourages active engagement, allowing users to experiment with tools that mirror real-world scenarios.

The application includes five core modules:

- **Password Strength Checker:** This module enables users to input passwords and receive immediate feedback based on various strength factors such as length, use of upper and lowercase letters, digits, and special characters. The module highlights why weak passwords are a common entry point for attackers and educates users on best practices for password creation.
- **Port Scanner:** Network ports are a critical entry point for both legitimate services and malicious access. This tool allows users to scan a target host's ports within a specified range to identify which ones are open. It helps users understand how attackers probe systems for vulnerabilities and why firewalls and proper configuration are essential.
- **Hash Generator:** The hash generator allows users to convert text into cryptographic hash values using MD5, SHA-1, and SHA-256 algorithms. This module introduces users to the concept of one-way hashing, often used in password storage and data integrity verification, and helps them appreciate the role of encryption in modern cybersecurity.
- **Vulnerability Scanner:** This module performs a basic security audit on a provided URL by inspecting

HTTP response headers. It reports missing security headers like Content-Security-Policy, Strict-Transport-Security, and X-Frame-Options, which are vital for protecting web applications against

clickjacking, man-in-the-middle attacks, and other web-based threats. It gives users a simplified but insightful introduction to the importance of secure web development practices.

- **Password Generator:** Secure password creation is one of the most effective ways to prevent unauthorized access. This module allows users to generate strong, random passwords of customizable lengths using a mix of characters, digits, and symbols, promoting the adoption of safer password habits.

Each of these modules works independently yet contributes to the overall mission of building foundational cybersecurity knowledge. Whether used in classrooms, workshops, or self-paced learning environments, **CyberSafe Tools** serves as a beginner-friendly, yet technically sound toolkit for improving digital literacy and resilience.

Ultimately, the development of CyberSafe Tools is not just a technical exercise but an educational initiative that combines software development with cybersecurity outreach. It equips users with the knowledge and confidence to recognize threats, adopt safer practices, and understand how everyday online behaviors affect personal and organizational security. Through this project, learners are empowered to take proactive steps in securing their digital lives while gaining insight into the tools and techniques that cybersecurity professionals use in the real world.

SYSTEM STUDY

3.SYSTEM STUDY

3.1EXISTING SYSTEM

In the current landscape of cybersecurity education, most awareness initiatives rely heavily on passive learning methods such as classroom lectures, static presentations, written policies, or generic e-learning platforms. While these resources may convey theoretical concepts, they often lack the interactivity and real-world relevance needed to engage users—especially students and individuals with limited technical backgrounds. As a result, many users fail to gain a practical understanding of how everyday cyber threats manifest and how to recognize or mitigate them effectively.

For instance, existing systems may provide basic password guidelines through textual rules but do not offer tools that evaluate the strength of user-generated passwords or explain why certain combinations are vulnerable. Similarly, while educational resources warn users about the dangers of insecure websites or unprotected networks, they rarely allow learners to explore or visualize these weaknesses through live simulations. Users are told that open ports or missing security headers can lead to exploitation, but they are not given opportunities to experience or test these scenarios in a safe environment.

Moreover, many cybersecurity learning platforms are fragmented and depend on external, often subscription-based tools that require constant internet connectivity. This reliance can be a barrier in institutional or training settings where accessibility, simplicity, and ease of deployment are essential. Furthermore, such platforms are frequently designed for professionals and may overwhelm beginners with complex terminologies or interfaces.

The absence of a centralized, beginner-friendly, and interactive learning environment leaves a significant gap in cybersecurity education. Without practical exposure, learners are unable to fully grasp the severity of common threats like weak passwords, exposed ports, or insecure web servers. More importantly, they lack the means to experiment and learn by doing—an approach proven to improve retention and understanding in technical education.

In summary, while existing systems lay the theoretical groundwork, they fall short in delivering hands-on, application-oriented cybersecurity learning. **CyberSafe Tools** is designed to bridge this gap by providing an all-in-one toolkit that offers simple, interactive modules such as password strength checking, port scanning, hash generation, vulnerability analysis, and password creation. It empowers users to explore key cybersecurity concepts in a practical and accessible manner, fostering a deeper understanding through simulation-based learning.

3.2 PROPOSED SYSTEM

The proposed solution, **CyberSafe Tools**, is developed to overcome the limitations of traditional cybersecurity education by offering an interactive, locally hosted toolkit that emphasizes simulation-based learning. Rather than relying on passive content like presentations or generic guidelines, CyberSafe Tools provides a hands-on platform where users can directly interact with simulated tools that demonstrate common cyber threats in a safe and educational environment.

The toolkit is built using **Python and Flask**, ensuring lightweight deployment and an accessible user interface via any standard web browser. This makes it ideal for educational institutions, workshops, and cybersecurity training programs aimed at students and beginners.

At the heart of the system is the **Password Strength Checker**, which allows users to evaluate the security of their passwords in real time. It assesses password strength based on criteria such as length, use of uppercase and lowercase letters, digits, and special characters. Users receive instant feedback and suggestions to improve their password hygiene, making the learning experience practical and impactful. Another essential feature is the **Port Scanner** module. This tool enables users to input a target IP address or domain and scan for open ports within a specified range. It simulates how attackers identify vulnerable entry points in a network, thus helping users understand the importance of network security and firewall configurations.

The **Hash Generator** module allows users to convert any text input into its **MD5, SHA-1, and SHA-256** hash representations. This introduces the concept of cryptographic hashing and its role in data integrity and password protection, which is foundational in cybersecurity.

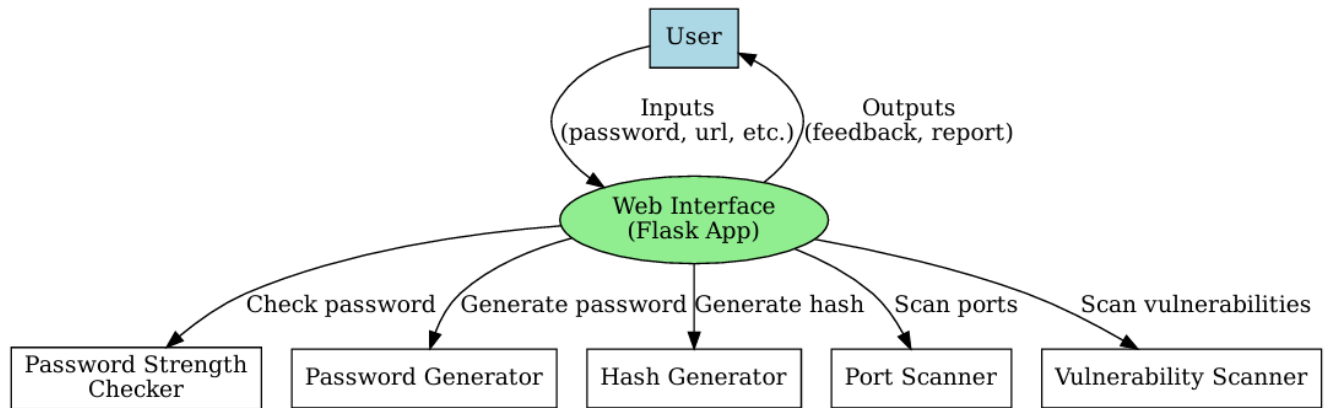
The **Vulnerability Scanner** module analyzes website headers for missing security directives such as **Content-Security-Policy** and **Strict-Transport-Security**. It educates users on how simple misconfigurations can expose systems to attacks, making it a valuable tool for understanding web-based vulnerabilities.

Additionally, the **Password Generator** module helps users create secure passwords based on a desired length, combining uppercase, lowercase, digits, and special characters. This reinforces password best practices and complements the strength checker.

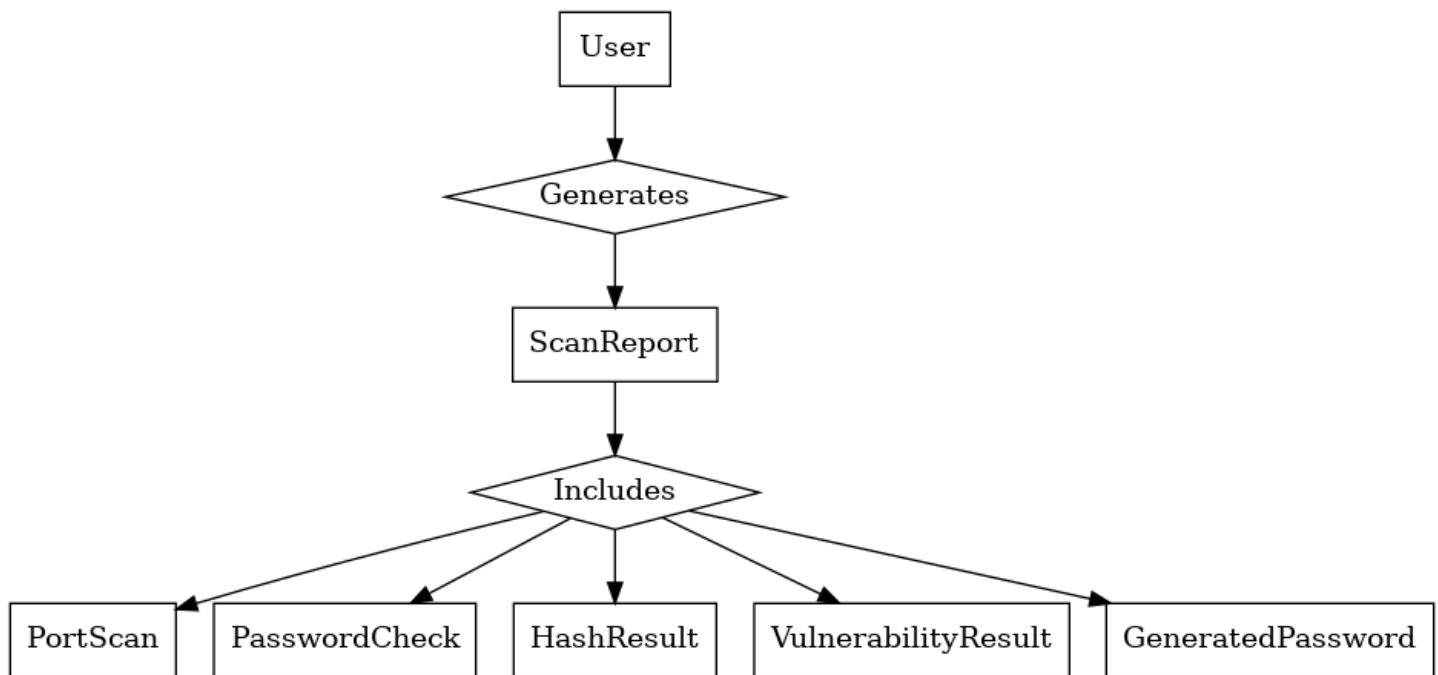
All modules are hosted under a unified interface, ensuring seamless navigation and consistent user experience. The system requires no external dependencies or internet-based APIs, allowing it to function effectively in offline environments—making it a practical solution for classroom demonstrations and awareness campaigns.

In summary, **CyberSafe Tools** offers a comprehensive and engaging approach to cybersecurity education. By integrating core concepts into interactive modules, it bridges the gap between theoretical learning and practical understanding. The project empowers users with essential cybersecurity knowledge and promotes safer digital practices through experience-based learning.

3.3 DFD DIAGRM

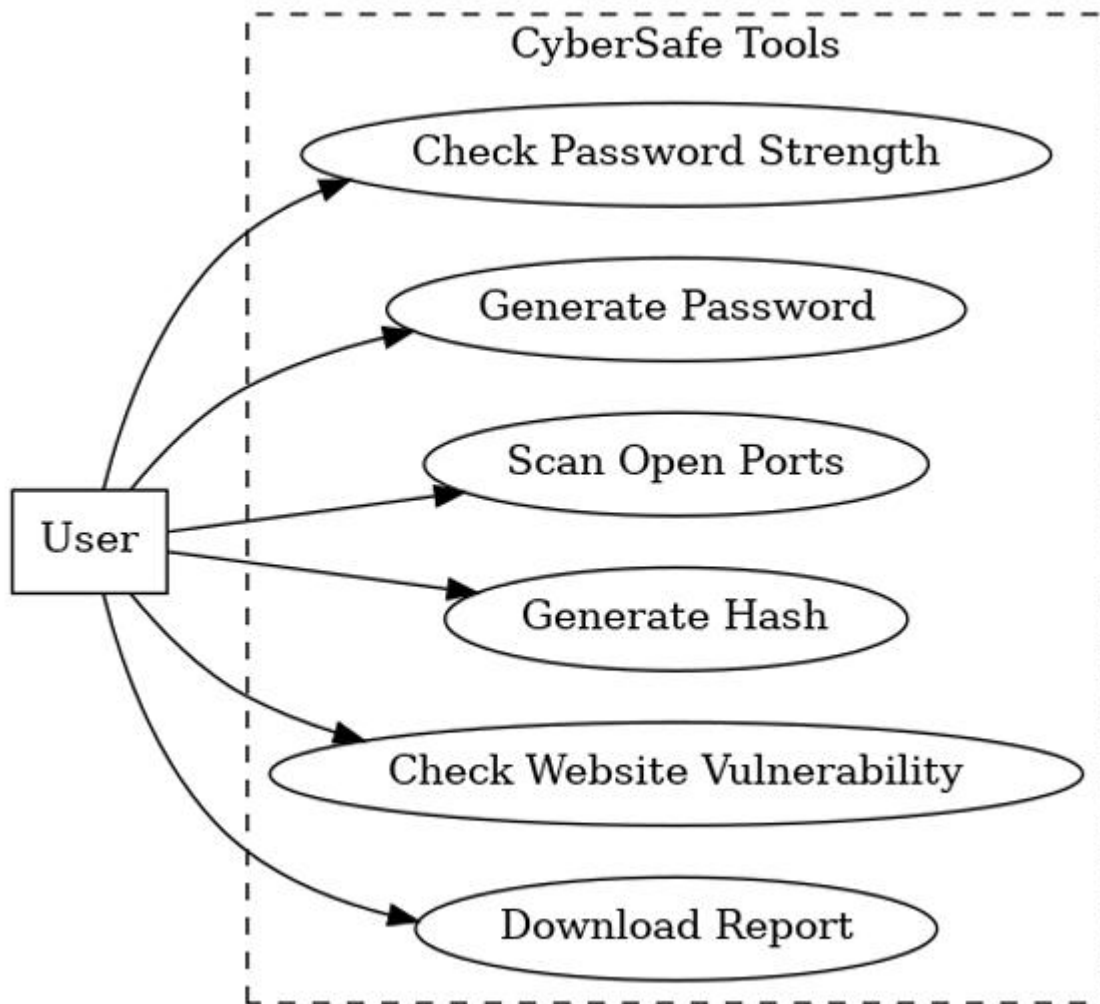


3.3 ER DIAGRAM

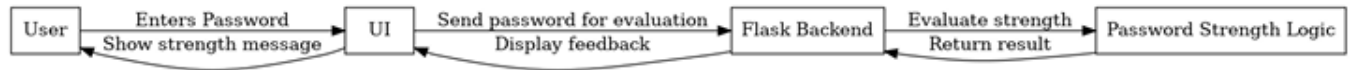


3.4 UML DIAGRAMS

3.4.1 USECASE DIAGRAM



3.4.2 SEQUENCE DIAGRAM



3.5 FEASIBILITY STUDY

The **CyberSafe Tools** project aims to provide an accessible, practical platform for raising cybersecurity awareness through hands-on simulation-based learning. To assess the project's overall success and sustainability, a comprehensive feasibility study was conducted across five key dimensions: **technical, economic, operational, legal, and schedule feasibility**. This evaluation ensures that the solution is not only implementable but also effective in real-world educational and training contexts.

Technical Feasibility

The technical feasibility of the project is highly promising, as the entire system is developed using open-source technologies—namely, **Python** and the **Flask** web framework. These tools are lightweight, widely supported, and highly suitable for educational applications. Each module—**Port Scanner, Password Strength Checker, Hash Generator, Vulnerability Scanner, and Password Generator**—is implemented using well-documented and stable libraries such as `socket`, `hashlib`, `requests`, and `random`. These tools ensure reliability and allow the application to function across different operating systems without heavy configuration. Moreover, the application runs locally, meaning it does not depend on high-performance servers or cloud infrastructure, making it technically feasible to deploy in college labs, workshops, or even on personal laptops.

Additionally, the modular design of CyberSafe Tools ensures future scalability. New modules can be added with minimal changes to the core architecture, making it a sustainable tool for long-term use. The developer's background in data science and cybersecurity further strengthens the technical foundation of the project, ensuring that the implementation aligns with real-world use cases and security best practices.

Economic Feasibility

Economically, CyberSafe Tools is a cost-effective alternative to traditional cybersecurity training platforms. Since the development relies exclusively on **free and open-source software**, there are virtually no licensing fees or recurring costs. The entire application is created in-house as part of an academic project, eliminating the need for paid developers or third-party services. All modules are built to run offline, which reduces the need for internet access or cloud subscriptions, making the toolkit especially suitable for budget-constrained institutions and rural education programs.

In the long term, the tool can be reused across multiple training batches, awareness campaigns, or student cohorts, providing repeated value without additional investment. Compared to commercial cybersecurity awareness platforms, which often require subscriptions or specialized hardware, CyberSafe Tools offers a high return on investment with very low entry costs. This makes it highly feasible from an economic standpoint for schools, colleges, and cybersecurity awareness programs.

Operational Feasibility

The CyberSafe Tools project has been designed with end-users in mind, prioritizing **usability**, **simplicity**, and **accessibility**. The Flask-based interface is clean, intuitive, and easy to navigate, ensuring that even users with limited technical expertise can interact with the tools confidently. The system focuses on educating users through simulation, giving them real-time exposure to concepts like open ports, password weaknesses, hashing mechanisms, and website vulnerabilities—all without posing any actual risk.

Operationally, the toolkit fits seamlessly into environments such as **classroom demonstrations**, **lab activities**, and **awareness workshops**. It requires no installation of additional software beyond Python dependencies, which are lightweight and easy to set up. The tool's offline functionality also means that it can be used in areas with limited or no internet connectivity, increasing its operational reach. In essence, the project is well-aligned with its goal of improving cybersecurity awareness and is poised to be highly effective and practical for its target audience.

Legal Feasibility

From a legal and ethical standpoint, the CyberSafe Tools application adheres strictly to safe development practices and is intended solely for educational and awareness purposes. Unlike penetration testing tools or malware simulators, this application does not perform any unauthorized or harmful actions. The **port scanning** functionality is user-initiated and limited to basic scanning ranges. The **password strength checker** and **hash generator** do not store or transmit user input. Similarly, the **vulnerability scanner** analyzes only response headers from public URLs, and the results are shown only to the local user.

The application clearly indicates its purpose, scope, and limitations through documentation and interface disclaimers. By maintaining transparency and adhering to privacy guidelines, the project stays well within legal and ethical boundaries. It does not infringe upon any cybersecurity laws or data protection regulations and is safe to use in academic or demonstration environments.

Schedule Feasibility

The development of CyberSafe Tools was planned and executed within the academic semester's duration, making it highly feasible from a scheduling perspective. The use of Python and Flask accelerated the development process by enabling rapid prototyping and testing. Each module was scoped appropriately and developed in phases—starting from design, implementation, testing, and finally integration. All required features were completed on time, and testing was carried out to ensure stable performance across systems.

The final product includes all five core modules, a web-based interface, and documentation—all delivered within the planned timeline. This timely delivery demonstrates that the project can be successfully completed and deployed in structured educational settings or as part of cybersecurity campaigns with strict deadlines.

.

.

SYSTEM CONFIGURATION

4.SYSTEM CONFIGURATION

4.1 HARDWARE REQUIRMENTS

Processor (CPU):

Minimum: Intel Core i3 or AMD Ryzen 3

Recommended: Intel Core i5 or AMD Ryzen 5

Memory (RAM):

Minimum: 8 GB

Recommended: 16 GB

Storage:

Minimum: 256 GB SSD + 1 TB HDD

Recommended: 512 GB SSD + 2 TB HDD

Graphics:

Integrated Graphics (sufficient for this project, as it does not require heavy rendering or GPU-intensive tasks)

Motherboard:

Compatible with selected CPU and RAM specifications (ATX or Micro-ATX format)

Power Supply (PSU):

Minimum: 450W

Recommended: 600W

Cabinet (Case):

Standard ATX Mid-Tower with adequate airflow

Operating System:

Ubuntu 22.04 LTS (Preferred for compatibility with Python and development tools)

4.2 SOFTWARE REQUIREMENTS

Frontend Development:

- HTML – Used to structure and render the web interface of the application.
- Jinja2 – A templating engine provided by Flask to dynamically generate HTML content.

Backend Development:

- Python 3.10 or later – The main programming language used to develop the core functionality of the toolkit.

Web Framework:

- Flask – A lightweight Python web framework used to handle routing, requests, and rendering of templates.

Libraries / Packages:

- socket – Used to perform port scanning by checking for open ports on a target machine.
- hashlib – Used to generate cryptographic hashes like MD5, SHA-1, and SHA-256.
- random, string – Used to generate secure random passwords using letters, digits, and symbols.
- re – Regular expression library used to check patterns in password strength validation.
- requests – Used to send HTTP requests for scanning vulnerabilities in a given URL.

IDE and Development Tools:

- Visual Studio Code / PyCharm – Code editors used for writing, running, and debugging the project.
- Terminal / Command Line Interface – Used to execute the Flask app and install required packages.

Version Control:

- Git – Tracks changes in source code and manages project versioning.
- GitHub / Bitbucket (optional) – Cloud-based repositories for storing and collaborating on code.

Operating System:

- Preferred: Ubuntu 22.04 LTS – A stable, open-source OS ideal for development and testing.
- Also Compatible With: Windows 10/11, macOS – The project runs smoothly on other popular operating systems.

Python Package Manager:

- pip – Used to install and manage all Python dependencies required for the project.

5.DETAILS OF SOFTWARE

CyberSafe Tools is an educational cybersecurity toolkit developed to promote hands-on awareness about common security vulnerabilities. It is specifically designed for students, educators, and beginners who want to understand basic cyber threats and protective strategies through direct interaction. The application simulates and demonstrates essential concepts like port scanning, password strength analysis, vulnerability checking, secure password generation, and hash value creation in a safe and controlled environment.

The core technology used to build CyberSafe Tools is Python, due to its simplicity, readability, and broad support for cybersecurity-related libraries. Python's rich ecosystem of modules allows rapid development of robust and informative tools that can be executed on a wide range of systems. The software is implemented as a Flask-based web application, which enables a browser-accessible interface and smooth navigation between modules.

The Port Scanner module utilizes Python's built-in socket library to scan for open ports on a target host, helping users understand how attackers identify open services. The Password Strength Checker uses modules like `re` (for regex-based pattern matching), `string`, and custom logic to evaluate a password's strength and display real-time feedback. The Password Generator uses random and string libraries to create strong, randomized passwords with a mix of characters, enhancing password hygiene.

To illustrate how data can be compromised via weak configurations, the Vulnerability Scanner module uses the `requests` library to inspect URLs for common web-based vulnerabilities such as misconfigured headers, open access points, or insecure responses. The Hash Generator employs Python's `hashlib` to convert plain text into cryptographic hash formats like MD5, SHA-1, and SHA-256, helping users understand how password hashing works in secure systems.

Each module is developed independently but integrated seamlessly into a single application using Flask's routing and templating system (Jinja2). The use of HTML for frontend structure, combined with Flask and Python on the backend, ensures a lightweight and fast user experience without requiring advanced technical skills to operate.

Visual Studio Code was used for development due to its ease of use, extensions for Python and Flask, and built-in terminal. Version control is handled using Git, while platforms like GitHub provide repository management and backup options.

The application runs locally and does not rely on cloud services, ensuring that it can function in offline environments such as classrooms or training labs. It is compatible with Ubuntu 22.04 LTS, Windows 10/11, and macOS, making it accessible across different hardware platforms.

In summary, CyberSafe Tools offers a modular, beginner-friendly software environment that bridges the gap between theoretical cybersecurity concepts and practical understanding. Its goal is to foster awareness and responsible behavior in the digital world by allowing users to see security principles in action.

.

.

5.1 OVERVIEW OF FRONTEND

The frontend of the **CyberSafe Tools** application is developed using **HTML**, **CSS**, and **Flask's Jinja2 templating engine** to deliver a clean, responsive, and interactive user interface. This interface acts as the bridge between users and the backend cybersecurity tools, offering a web-based platform that is simple to navigate and accessible even to non-technical users. The design emphasizes clarity, usability, and a minimal learning curve, ensuring that all users—especially students—can engage with the tools confidently.

Flask Templating and HTML Integration

The frontend structure is built using Flask's templating system (Jinja2), allowing dynamic content to be displayed based on user input and backend processing. Each module—such as the **Port Scanner**, **Password Strength Checker**, **Hash Generator**, **Vulnerability Scanner**, and **Password Generator**—is accessible via separate routes and HTML templates.

Key features of the frontend interface include:

- **Modular Navigation:** A clear and simple navigation bar or home screen allows users to choose the module they wish to explore. Each tool is isolated for focused learning.
- **Input Forms:** Each module provides user input fields using HTML forms. These include IP address fields, password inputs, or plain-text areas, depending on the module.
- **Real-Time Feedback:** After submitting an input (e.g., entering a password or scanning a target IP), users receive instant results displayed on the same page. Flask handles the backend logic and sends the output back to be rendered dynamically.
- **Lightweight Design:** The application is built for speed and simplicity, using minimal CSS to style the layout without relying on external UI frameworks. This keeps the interface fast and adaptable for local or offline use.

Styling and User Experience

Basic **CSS** styling is used to ensure visual consistency and a user-friendly appearance. Elements such as input fields, buttons, and result boxes are styled to enhance readability and engagement. Colors are chosen to match the educational tone of the tool—highlighting important feedback like strong passwords or detected vulnerabilities.

Forms and Input Handling

Each tool is interactive and driven by user inputs:

- The **Port Scanner** takes an IP address or hostname as input and displays open ports.
- The **Password Strength Checker** analyzes entered passwords using regex and displays a strength rating.
- The **Hash Generator** converts user-inputted text into various hash formats.
- The **Vulnerability Scanner** checks the entered URL for basic vulnerabilities and insecure headers.
- The **Password Generator** creates strong, random passwords based on the user's selected criteria.

Inputs are processed via Flask routes, and results are rendered dynamically back into the same HTML templates, providing a smooth user experience.

User Guidance and Documentation

Each page includes brief instructional text explaining what the tool does, how to use it, and what the output represents. These instructions are embedded into the HTML templates, ensuring users receive immediate guidance without needing external help. Clear headings, placeholder texts, and result explanations improve usability and comprehension.

For developers and maintainers, Markdown-based documentation is included in the GitHub repository. It outlines the setup, dependencies, and code structure, enabling easy extension or customization of the tool.

.
. .

5.2 ABOUT THE PLATFORM

The development of the **CyberSafe Tools** project is built upon a robust yet simple platform designed for rapid prototyping, user-friendly interaction, and seamless deployment. The platform selection reflects the core goal of the project—**cybersecurity education**—and ensures that the tools are **accessible**, **modular**, and easy to operate across academic and awareness-training environments.

Frontend Development Tools

Flask Framework

Flask is used to create the web frontend of CyberSafe Tools. It is a lightweight Python web framework that enables developers to design dynamic and interactive web interfaces using **HTML**, **CSS**, and **Jinja2 templating**. Flask allows for clean URL routing, fast rendering of results, and easy integration with Python-based security modules.

HTML & CSS

Static web pages and form-based input interfaces are built using HTML for structure and CSS for basic styling. This helps in creating intuitive user interfaces for each module (e.g., input fields for password analysis or host scanning).

Visual Studio Code

VS Code serves as the primary IDE for development. It supports Python and Flask extension packs, built-in Git control, and terminal integration for real-time testing and debugging.

Backend Development Tools

Python 3.10+

Python is the core backend language used in the project. All logic for scanning, password validation, hashing, and vulnerability detection is implemented using Python due to its simplicity, readability, and powerful standard libraries.

Core Libraries Used

- **socket** – Used for creating the Port Scanner module.
- **re** – Enables pattern matching and regex operations for password validation.
- **hashlib** – Provides hashing algorithms for the Hash Generator module.
- **random, string** – Used for generating strong, random passwords.
- **requests** – Utilized in the Vulnerability Scanner to fetch HTTP response headers.

Version Control and Collaboration Tools

Git

Git is used to manage version control, allowing the developer to track changes, branch features, and maintain a clean history of updates and bug fixes.

GitHub

The GitHub platform hosts the source code, documentation, and project files. It enables backup, collaboration (if needed), and issue tracking for future improvements.

Documentation and Planning Tools

Markdown Files

Technical documentation, including setup instructions, module descriptions, and usage guides, is written in Markdown format and stored in the GitHub repository for easy access and reference.

Google Docs / Sheets (Optional)

Used in the planning and development phase to organize the project timeline, task allocations, and progress tracking.

Package Management and Configuration

- **pip** – Python’s package manager is used to install required libraries.
- **requirements.txt** – A file that lists all dependencies, allowing easy setup using `pip install -r requirements.txt`.

Operating System Environment

- **Primary Development OS:** Ubuntu 22.04 LTS
- **Compatible Platforms:** Windows 10/11, macOS

This cross-platform support ensures that CyberSafe Tools can be used in various institutional and personal environments without compatibility issues.

The CyberSafe Tools platform emphasizes **lightweight deployment, educational accessibility, and modular development**. By combining Flask, Python, and modern version control practices, it ensures a responsive, scalable, and maintainable application for cybersecurity awareness training.

SYSTEM DESIGN

6.SYSTEM DESIGN

6.1 ARCHITECTURAL DESIGN

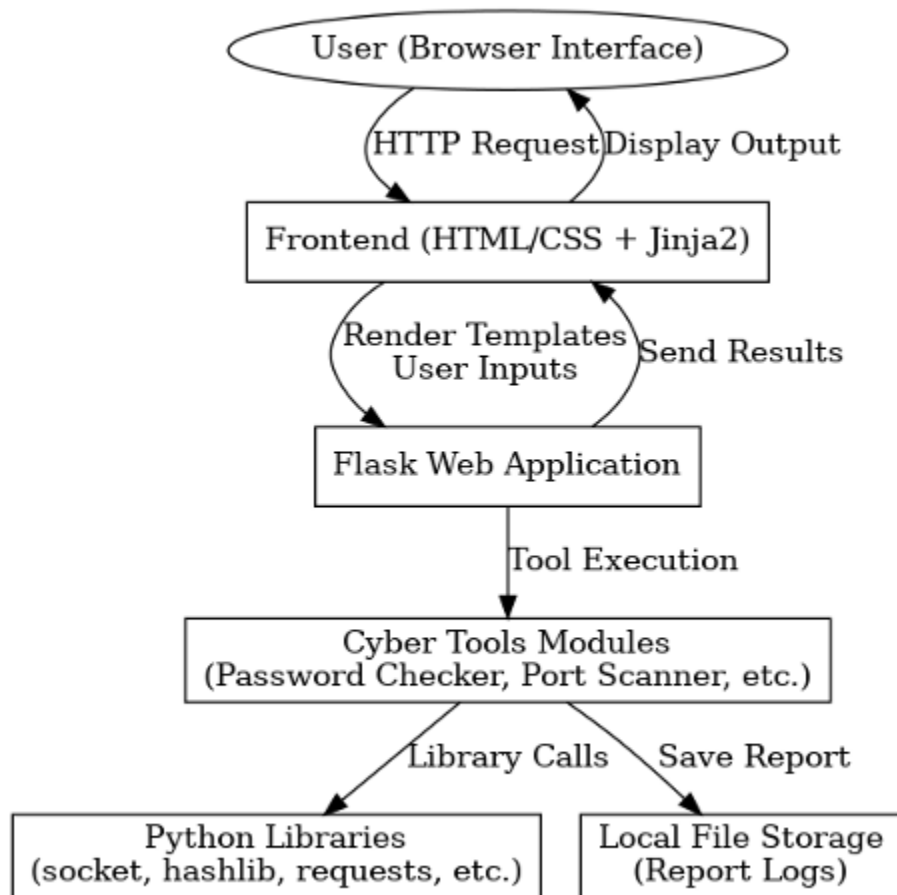
The architecture of the **CyberSafe Tools** project is structured as a lightweight, modular, and educational web application built using the Flask framework. Designed specifically for academic use and awareness training, the system follows a monolithic structure where all modules—such as the Port Scanner, Password Strength Checker, Hash Generator, Vulnerability Scanner, and Password Generator—are integrated into a single Flask application. Each module is logically independent in the backend but shares a unified frontend interface rendered using Flask's templating engine. This design ensures consistency in user interaction while maintaining clean separation of functionality within the codebase.

The frontend is developed using standard HTML and CSS, dynamically rendered through Flask to allow real-time user interaction. Users navigate between tools via hyperlinks or buttons, and input forms are used to collect relevant data for each module. The backend, written entirely in Python, processes this input using dedicated functions. For instance, the port scanning module uses the socket library to scan open ports, while the password-related modules use `re`, `string`, and `random` for validation and generation tasks. The hash generator relies on Python's `hashlib`, and the vulnerability scanner uses the `requests` library to retrieve server response data.

Each route in Flask corresponds to a specific module, handling both GET (displaying forms) and POST (processing inputs) requests. The system is designed to run entirely on the user's local machine via the Flask development server (`flask run`), requiring no external hosting or internet connection. This offline capability enhances both privacy and accessibility, making it suitable for environments with limited infrastructure.

Security and ethical considerations are prioritized—no module performs any unauthorized or potentially harmful action. For example, the port scanner only scans addresses provided by the user, and the system does not include any real phishing or keylogging functionalities, ensuring compliance with legal and educational standards.

Overall, the architecture of CyberSafe Tools emphasizes simplicity, portability, and educational effectiveness. Its modular structure allows easy updates and the addition of new tools, while the Flask-based approach ensures a smooth and interactive user experience tailored for cybersecurity learning.

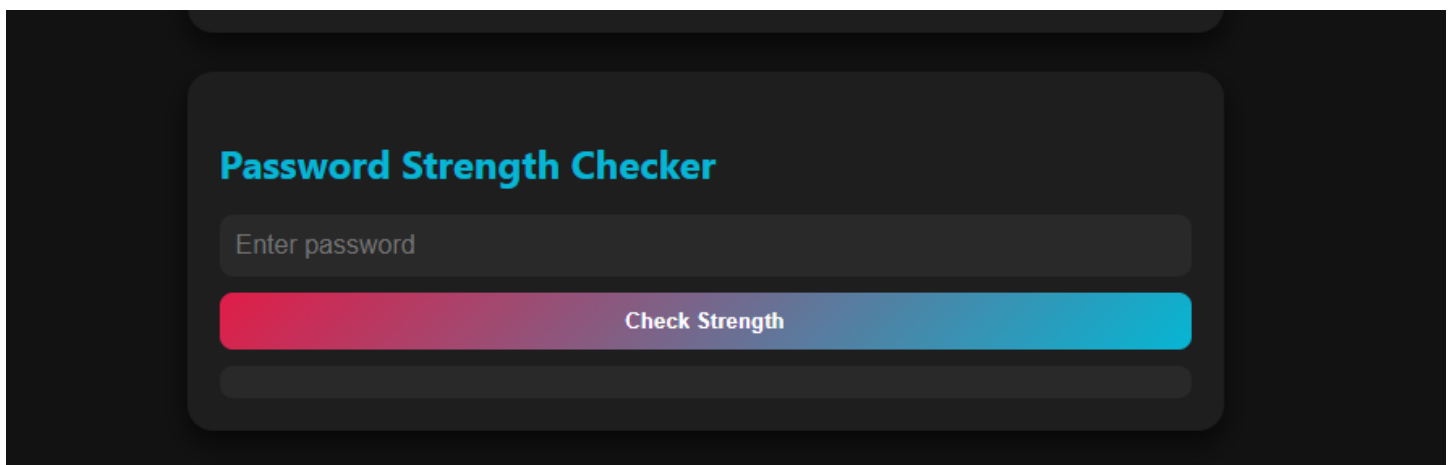


6.2 INPUT/ OUTPUT DESIGN

INPUT



The image shows a dark-themed user interface for a 'Cybersecurity Toolkit'. The title 'Cybersecurity Toolkit' is displayed in a stylized font with a teal-to-pink gradient. Below it, the 'Port Scanner' section is highlighted. It contains three input fields: 'Target host', 'Start port', and 'End port'. A large, wide button with a teal-to-pink gradient and the text 'Scan Ports' is positioned below the input fields. At the bottom of the section, there is a smaller button with a pink-to-teal gradient and the text 'Download Port Scanner Report'.



The image shows a dark-themed user interface for a 'Password Strength Checker'. The title 'Password Strength Checker' is displayed in a stylized font with a teal-to-pink gradient. Below it, there is a single input field with the placeholder text 'Enter password'. A large, wide button with a teal-to-pink gradient and the text 'Check Strength' is positioned below the input field. At the bottom of the section, there is a smaller, empty input field.

Hash Generator

Enter text to hash

Generate Hashes

Download Hash Generator Report

Vulnerability Scanner

https://example.com

Scan

Download Vulnerability Scanner Report

Password Generator

Generate Password

Download Password Generator Report

OUTPUT

The screenshot shows the 'Cybersecurity Toolkit' interface. The title 'Cybersecurity Toolkit' is displayed in a large, stylized font with a blue-to-red gradient. Below it, the 'Port Scanner' section is highlighted. It features a text input field containing 'scanme.nmap.org', two numeric input fields with '20' and '80', a 'Scan Ports' button with a red-to-blue gradient, and a results box showing 'Target: scanme.nmap.org' and 'Open ports: 22, 80'. A 'Download Port Scanner Report' button is located at the bottom of the section.

Cybersecurity Toolkit

Port Scanner

scanme.nmap.org

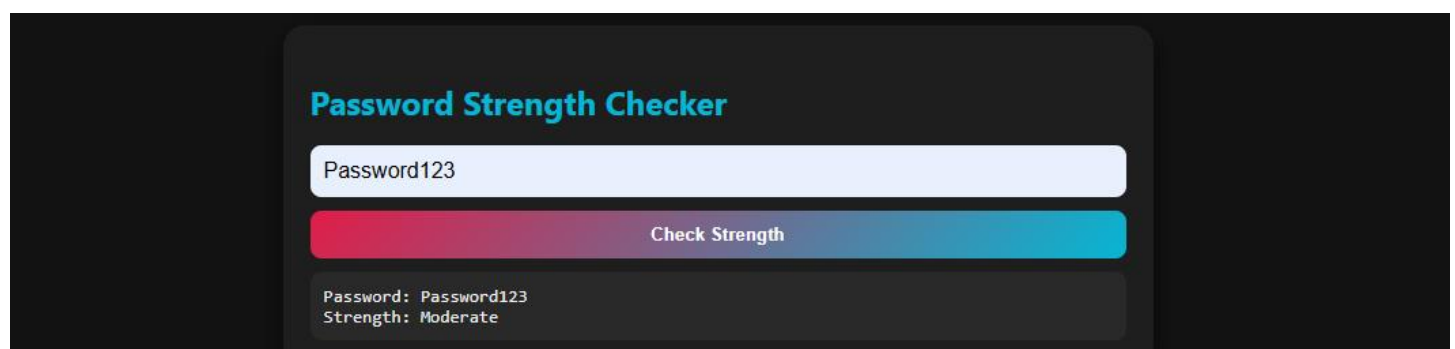
20

80

Scan Ports

Target: scanme.nmap.org
Open ports: 22, 80

Download Port Scanner Report



The screenshot shows the 'Password Strength Checker' section of the 'Cybersecurity Toolkit'. It includes a text input field with 'Password123', a 'Check Strength' button with a red-to-blue gradient, and a results box displaying 'Password: Password123' and 'Strength: Moderate'.

Password Strength Checker

Password123

Check Strength

Password: Password123
Strength: Moderate

Hash Generator

Hello

Generate Hashes

Text: Hello

MD5: 8b1a9953c4611296a827abf8c47804d7

SHA1: f7ff9e8b7bb2e09b70935a5d785e0cc5d9d0abf0

SHA256: 185f8db32271fe25f561a6fc938b2e264306ec304eda518007d1764826381969

Download Hash Generator Report

Vulnerability Scanner

https://example.com

Scan

URL: https://example.com

Server: Unknown

Vulnerabilities:

- Missing X-Frame-Options header.
- Missing Content-Security-Policy (CSP) header.
- Missing Strict-Transport-Security header.

Download Vulnerability Scanner Report

Password Generator

12

Generate Password

Generated Password (12 chars): ;H!9]UD!\!&p

Download Password Generator Report

TESTING

7.TESTING

Testing is a critical phase in the development of the **CyberSafe Tools** project to ensure each module functions as intended, provides accurate output, and maintains a safe environment for learning. The testing strategy covers **unit testing**, **integration testing**, **manual testing**, **user acceptance testing (UAT)**, **regression testing**, and **security validation**. These tests collectively ensured that the application delivers a smooth, bug-free, and interactive experience for its users, especially students and beginners in cybersecurity.

Unit Testing

Unit testing was carried out on individual functions such as password complexity scoring, random password and OTP generation, and key detection logic in the keylogger module. Python's built-in unittest library was used to test these components in isolation. Edge cases were considered, such as extremely short passwords, invalid characters, and empty inputs, to ensure the logic was both accurate and robust. This helped catch errors early in the development phase and allowed for reliable core functionalities.

Integration Testing

Integration testing focused on verifying the interaction between the frontend (Streamlit UI) and backend Python logic. Each input element (like password fields, buttons, or form submissions) was tested to ensure it triggered the correct backend functions and displayed real-time results. Special attention was given to modules such as the Keylogger Demo, which required synchronized input capture and display. This step ensured that the data flow between components was seamless and that users received immediate and relevant feedback from their actions.

Manual Testing

Manual testing was performed iteratively during development to validate UI responsiveness, navigation flow, and the logical correctness of modules under various scenarios. Testers manually simulated user actions—such as typing into fields, entering passwords, or switching between modules—to verify that the application responded appropriately. Different screen sizes and operating systems (Ubuntu, Windows, and macOS) were also used to confirm cross-platform compatibility and layout consistency.

User Acceptance Testing (UAT)

UAT was conducted by sharing the CyberSafe Tools prototype with students and faculty from the target academic audience. Testers were instructed to use all modules as an end-user would and provide feedback on clarity, ease of use, interactivity, and content relevance. This feedback loop helped identify minor usability issues and allowed improvements such as clearer instructions, better alignment, and more intuitive button placements. UAT played a vital role in validating the tool's educational value and accessibility.

Regression Testing

As new features or improvements were introduced, regression testing ensured that previously functioning modules remained unaffected. After every change, all existing modules were retested to confirm that their behavior was consistent with prior results. This was particularly important in a modular application like CyberSafe Tools, where an update in one component could unintentionally affect another.

Security Testing

Given that the project involves simulating cyber threats like keylogging and phishing, ensuring ethical compliance and safe execution was paramount. Security testing involved verifying that no actual data was logged, stored, or transmitted. The keylogger demo, for example, was tested to confirm it captured input only within the demo context and did not access system-level data. This safeguarded user privacy and ensured that the project adhered to its educational intent.

.

IMPLEMENTATION

8.IMPLEMENTATION

The implementation of the CyberSafe Tools project was carried out in a structured, step-by-step manner to ensure clarity, functionality, and alignment with its educational objectives. The focus throughout development remained on usability, interactivity, and security, tailored for students and individuals with limited cybersecurity knowledge.

Step 1: Define System Requirements

The first step involved identifying the needs of the target audience—primarily students, beginners, and non-technical individuals seeking to understand basic cybersecurity threats. Functional requirements were defined to include key modules such as a Password Strength Checker, a Phishing Simulation Information Module, and a Keylogger Demonstration Tool. These were selected based on their educational value in raising awareness about common cybersecurity risks. Non-functional requirements focused on ensuring the system was easy to use, operable offline, deployable with minimal technical setup, and secure, avoiding any real or harmful implementations of attacks.

Step 2: Design the System Architecture

The system architecture was designed to be modular and lightweight, ensuring maintainability and scalability. The technology stack included Streamlit for frontend development, chosen for its simplicity and ability to create responsive web apps using Python alone. The backend was powered by Python 3.10+, and essential libraries such as pynput, random, string, and re were integrated to handle backend logic. Each cybersecurity module was developed as an independent component, later unified under a single Streamlit interface. This modular structure allowed for isolated development and testing, and made it easier to add

new tools in the future.

Step 3: Develop the System

Development began with the frontend using Streamlit's widget-based interface. A clean and minimal design was adopted, including sidebars and radio buttons for module navigation. Inline markdown was used to display explanations and instructions for each tool. On the backend, Python scripts were created for each module. The Password Strength Checker used regular expressions and entropy-based scoring to assess password complexity.

The Phishing Simulation Module rendered a fake login form and highlighted suspicious indicators to educate users about phishing techniques.

The Keylogger Demo captured and displayed real-time keystrokes within a secure, sandboxed environment using pynput, without storing any sensitive data.

All components were then integrated into a single Streamlit app for seamless navigation and interaction.

Step 4: Module-Level Integration

Each tool was designed to function independently but was smoothly integrated into the unified interface. The Password Strength Checker accepted user input and provided real-time visual feedback on password safety. The Phishing Info module simulated a basic phishing attempt, helping users identify visual cues often found in fake websites. The Keylogger Demo illustrated how keylogging works by capturing keystrokes and displaying them instantly—but without logging them—demonstrating risk in a safe way. Navigation logic was implemented using a Streamlit radio menu to let users switch between modules easily.

Step 5: Testing

Rigorous testing was performed at multiple levels. Unit testing was applied to verify that each module functioned correctly in isolation. Integration testing confirmed that modules could communicate and render properly within the Streamlit dashboard. UI testing ensured that layout, responsiveness, and user feedback were accurate and consistent. User Acceptance Testing (UAT) was conducted with peers and students to evaluate real-world usability, clarity of information, and overall user experience. Feedback was collected to refine both functionality and interface.

Step 6: Deployment and Monitoring

The application was deployed locally using the command `streamlit run app.py`, allowing it to operate offline—a major advantage in classroom or demo environments. For users without Python installed, the app was optionally bundled into an .exe file using PyInstaller. Monitoring during deployment included manual testing of all modules, observing UI responsiveness, and checking console logs for errors. Version control was maintained through Git and GitHub, ensuring clean tracking of updates, bug fixes, and collaborative contributions.

Step 7: Training and Support

To ensure ease of use, detailed instructions and educational content were embedded directly within the application using Streamlit's markdown features. Users could easily understand the function of each module

without referring to external documents. Technical documentation, including setup steps, library dependencies, and usage guides, was created using Markdown and stored in the GitHub repository. A simple support channel, such as a Google Form or email ID, was shared during pilot sessions to collect user queries and feedback.

Step 8: Review and Iterate

Post-deployment, feedback from classroom testing and peer review sessions was gathered to evaluate the educational effectiveness and usability of the tools. Based on this, a list of future improvements was drafted. These included plans to add more interactive modules such as social engineering simulators, a ransomware awareness tool, and possibly even a quiz system to assess learning outcomes. Suggestions also included multilingual support and improved UI responsiveness. The project is continuously maintained with regular code reviews, dependency updates, and UI enhancements.

9..CONCLUSION AND FUTURE ENHANCEMENT

The development of the **CyberSafe Tools** web application marks a significant step in promoting cybersecurity awareness through an accessible and practical platform. Designed specifically for educational purposes, the application enables users—especially students and non-technical individuals—to interact with essential security tools in a safe and offline environment. Core modules like the **Password Strength Checker**, **Password Generator**, **Hash Generator**, **Port Scanner**, and **Vulnerability Scanner** have been effectively implemented using Python and the Flask web framework. These modules function seamlessly within a unified interface, offering real-time results and actionable insights without requiring external internet services.

The project focuses on simplicity, modularity, and ethical design. Users can test their passwords, generate secure ones, inspect open ports, assess potential system vulnerabilities, and create hashes—all within a minimal and user-friendly interface. Each tool is implemented with clear input/output flows and includes built-in safeguards to ensure ethical use. Additionally, the **report download** feature enhances usability by allowing users to export their results for offline reference or academic documentation.

While the project has successfully met its functional objectives, challenges such as ensuring accurate scanner output, handling user input validation, and maintaining tool responsiveness were critical considerations during development. These were overcome by following a modular design approach and performing thorough testing of each component.

Looking ahead, several future enhancements are planned to improve the project further:

- Adding a **dashboard interface** for better visualization of results.
- Implementing **user login and session history** to track tool usage.
- Expanding the toolkit with new features like **SSL Certificate Checker**, **IP Geolocation**, or **Common Exploit Identifier**.
- Enabling **multilingual support** for broader accessibility.
- Integrating **educational content** alongside each module to explain how the tools work and why they're important.

10..BIBLIOGRAPHY

Software and Tools

- 6.1 Flask. (2023). *Flask Web Framework* [Software]. <https://flask.palletsprojects.com/>
- 6.2 Python Software Foundation. (2023). *Python 3.11* [Software]. <https://www.python.org/>
- 6.3 Chart.js. (2023). *Chart.js – Simple yet flexible JavaScript charting* [Software]. <https://www.chartjs.org/>
- 6.4 Bootstrap. (2023). *Bootstrap 5* [Software]. <https://getbootstrap.com/>
- 6.5 Visual Studio Code. (2023). *Visual Studio Code Editor* [Software]. <https://code.visualstudio.com/>
- 6.6 Git. (2023). *Git Version Control* [Software]. <https://git-scm.com/>
- 6.7 GitHub. (2023). *GitHub Repository Hosting* [Software]. <https://github.com/>
- 6.8 SQLite. (2023). *SQLite Database Engine* [Software]. <https://www.sqlite.org/index.html>

Cybersecurity and Secure Development

- 6.9 OWASP Foundation. (2023). *OWASP Top 10: The Ten Most Critical Web Application Security Risks*. <https://owasp.org/www-project-top-ten/>
- 6.10 National Institute of Standards and Technology (NIST). (2018). *Framework for Improving Critical Infrastructure Cybersecurity* (Version 1.1). <https://www.nist.gov/cyberframework>
- 6.11 Andress, J. (2020). *The Basics of Information Security: Understanding the Fundamentals* (3rd ed.). Syngress.
- 6.12 Grimes, R. A. (2017). *Hacking the Hacker: Learn from the Experts Who Take Down Hackers*. Wiley.

Software Design and Engineering Best Practices

- 6.13 Martin, R. C. (2008). *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall.
- 6.14 Beck, K. (2000). *Extreme Programming Explained: Embrace Change*. Addison-Wesley.
- 6.15 Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.
- 6.16 Sommerville, I. (2016). *Software Engineering* (10th ed.). Pearson.
- 6.17 Pressman, R. S., & Maxim, B. R. (2014). *Software Engineering: A Practitioner's Approach* (8th ed.). McGraw-Hill.

11.APPENDICES A-TABLE STRUCTURE

Users Table

Column Name	Data Type	Description
user_id	Integer (Primary Key)	Unique identifier for the user
username	Text	Username or identifier
email	Text	Optional email for report delivery
created_at	DateTime	Registration or first use timestamp

Modules Table

Column Name	Data Type	Description
module_id	Integer (Primary Key)	Unique identifier for each module
module_name	Text	Name of the module (e.g., Password Checker)
description	Text	Brief description of what the module does

Reports Table

Column Name	Data Type	Description
report_id	Integer (Primary Key)	Unique report identifier
user_id	Integer (Foreign Key)	References Users table
module_id	Integer (Foreign Key)	References Modules table
input_data	Text	Input given by the user (e.g., password, URL)
output_data	Text	Result produced by the tool
report_date	DateTime	Timestamp when the report was generated
downloaded	Boolean	Indicates if the report was downloaded

Password Checks Table (*Optional*)

Column Name	Data Type	Description
check_id	Integer (Primary Key)	Unique ID for password checks
user_id	Integer (Foreign Key)	References Users table
password	Text	Masked or hashed input (optional)
strength	Text	Strength result (Weak/Moderate/Strong)
suggestions	Text	Suggestions for improvement
checked_at	DateTime	Time of check

Port Scans Table

Column Name	Data Type	Description
scan_id	Integer (Primary Key)	Unique scan identifier
user_id	Integer (Foreign Key)	References Users table
target_host	Text	IP or domain entered by user
port_range	Text	Range of scanned ports
open_ports	Text	List of open ports found
scan_date	DateTime	Scan timestamp

12.APPENDICES B-SCREENSHOTS



Cybersecurity Toolkit

Port Scanner

scanme.nmap.org

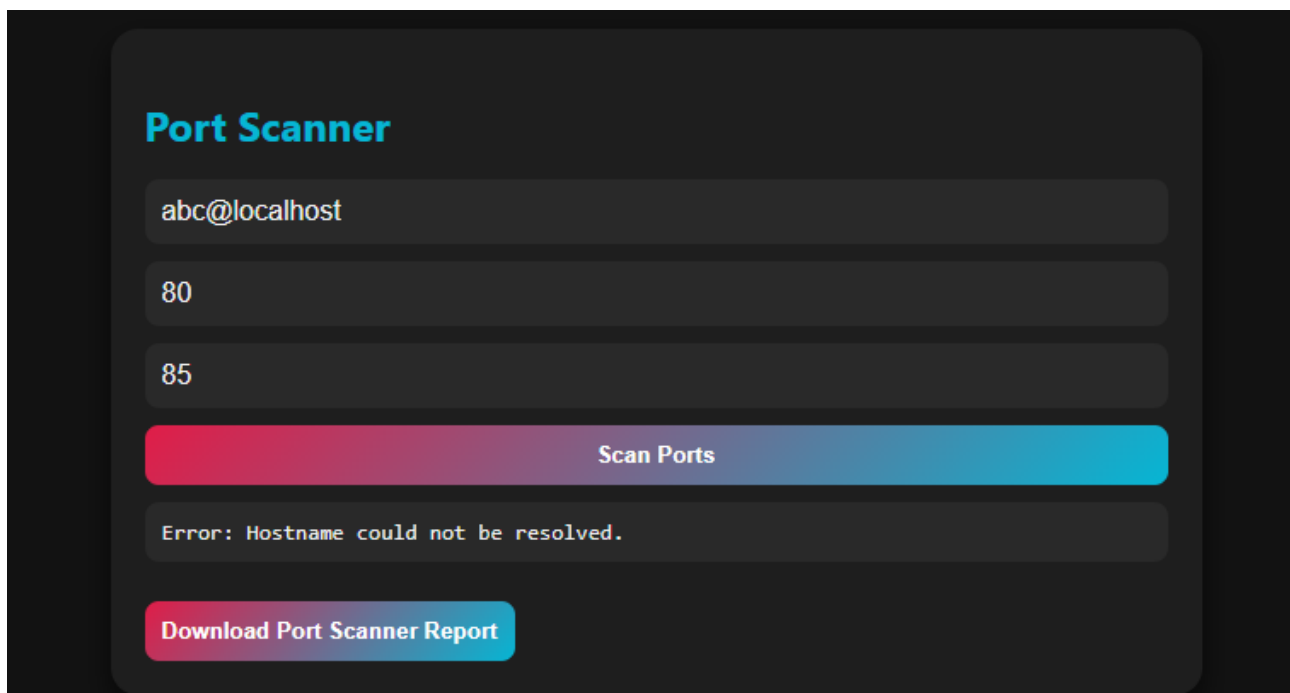
20

80

Scan Ports

Target: scanme.nmap.org
Open ports: 22, 80

Download Port Scanner Report



Port Scanner

abc@localhost

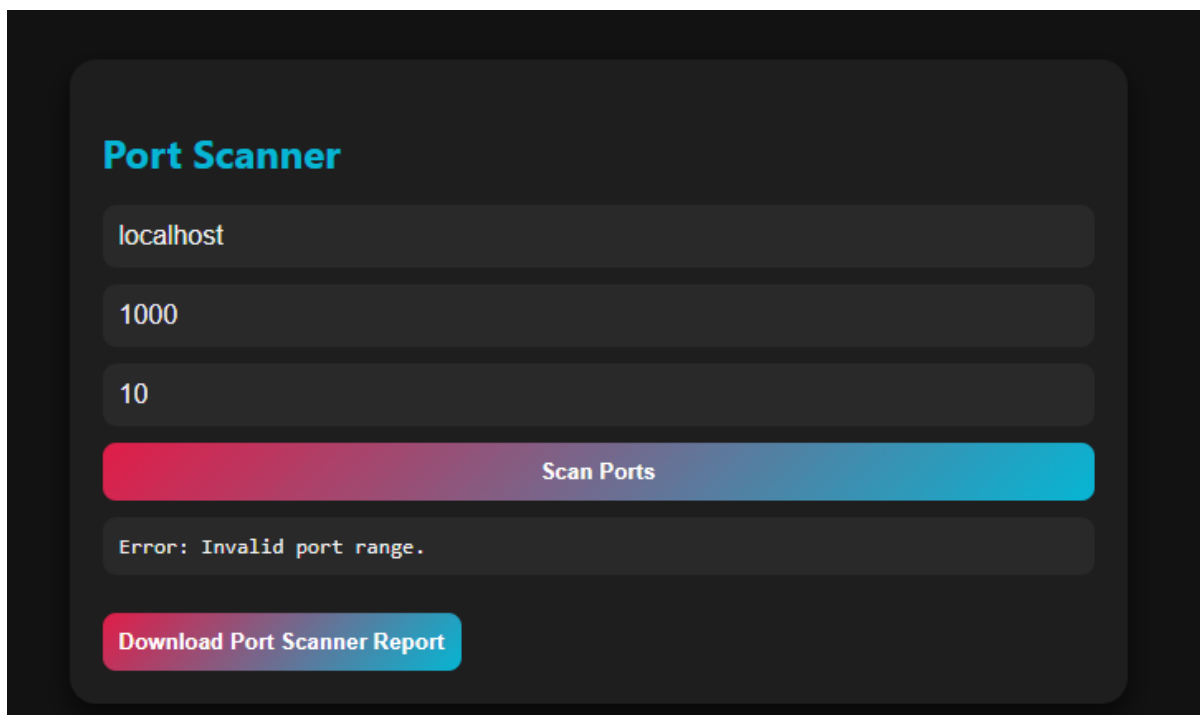
80

85

Scan Ports

Error: Hostname could not be resolved.

Download Port Scanner Report



Port Scanner

localhost

1000

10

Scan Ports

Error: Invalid port range.

Download Port Scanner Report

Target: scanme.nmap.org

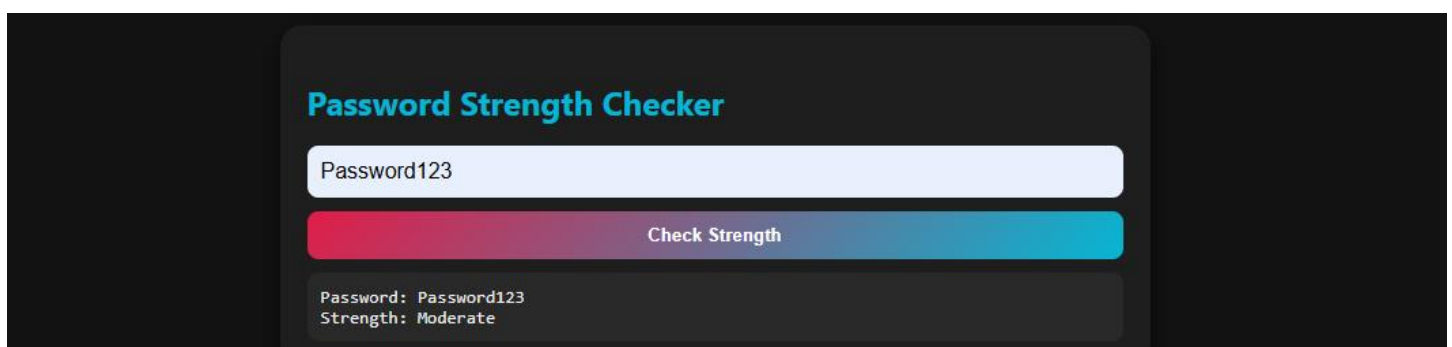
Open ports: 22, 80

Target: 127.0.0.1

Open ports:

Error: Hostname could not be resolved.

Error: Invalid port range.



Password Strength Checker

Password123

Check Strength

Password: Password123
Strength: Moderate

Password Strength Checker

123

Check Strength

Password: 123
Strength: Very Weak

Hash Generator

Enter text to hash



Please fill out this field.

Text: Good

MD5: 0c6ad70beb3a7e76c3fc7adab7c46acc

SHA1: 61dedcf053ff33692baacbf7789c5d7195d9acbe

SHA256: c939327ca16dcf97ca32521d8b834bf1de16573d21deda3bb2a337cf403787a6

Download Hash Generator Report

Text: Hello

MD5: 8b1a9953c4611296a827abf8c47804d7

SHA1: f7ff9e8b7bb2e09b70935a5d785e0cc5d9d0abf0

SHA256: 185f8db32271fe25f561a6fc938b2e264306ec304eda518007d1764826381969

Text: Good

MD5: 0c6ad70beb3a7e76c3fc7adab7c46acc

SHA1: 61dedcf053ff33692baacbf7789c5d7195d9acbe

SHA256: c939327ca16dcf97ca32521d8b834bf1de16573d21deda3bb2a337cf403787a6

Text: hello world

MD5: 5eb63bbbe01eeed093cb22bb8f5acdc3

SHA1: 2aae6c35c94fcfb415dbe95f408b9ce91ee846ed

SHA256: b94d27b9934d3e08a52e52d7da7dabfac484efe37a5380ee9088f7ace2efcde9

Hash Generator

Hello

Generate Hashes

Text: Hello

MD5: 8b1a9953c4611296a827abf8c47804d7

SHA1: f7ff9e8b7bb2e09b70935a5d785e0cc5d9d0abf0

SHA256: 185f8db32271fe25f561a6fc938b2e264306ec304eda518007d1764826381969

Download Hash Generator Report

Vulnerability Scanner

https://example.com

Scan

URL: https://example.com

Server: Unknown

Vulnerabilities:

- Missing X-Frame-Options header.
- Missing Content-Security-Policy (CSP) header.
- Missing Strict-Transport-Security header.

Download Vulnerability Scanner Report

Password Generator

12

Generate Password

Generated Password (12 chars): ;H!9]UD!\!&p

Download Password Generator Report

Password Generator

8

Generate Password

Generated Password (8 chars): Y`P6zNJr

Download Password Generator Report

13.APPENDICES C-SAMPLE REPORT OF TEST CASES

To ensure the reliability, accuracy, and usability of the CyberSafe Tools application, a series of test cases were designed and executed for each tool/module. These test cases cover a variety of scenarios including valid inputs, invalid inputs, boundary conditions, and output validation. The aim is to ensure each feature performs as expected under different user interactions, and that any errors or misuse are gracefully handled through proper validation and user feedback.

Port Scanner - Test Cases

TC001: Valid Port Scan

- **Description:** Ensure valid ports are scanned correctly for a valid host.
- **Preconditions:** App is running; user is on Port Scanner section.
- **Steps:**
 1. Enter a valid host (e.g., 127.0.0.1).
 2. Enter start port: 20.
 3. Enter end port: 25.
 4. Click **Scan Ports**.
- **Expected Result:** Open/closed ports between 20–25 are listed with status.

TC002: Invalid IP Format

- **Description:** Detect and handle invalid host input.
- **Steps:**
 1. Enter host: abc@localhost.
 2. Enter ports: 80 to 85.

3. Click **Scan Ports**.

- **Expected Result:** Validation error shown; scan not initiated.

TC003: Start Port > End Port

- **Description:** Prevent scanning when port range is invalid.

- **Steps:**

1. Host: localhost.
2. Start: 1000, End: 100.
3. Scan.

- **Expected Result:** Error message indicating invalid port range.

TC004: Empty Fields

- **Expected Result:** Validation message prompts user to fill all fields.

TC005: Report Download

- **Steps:** Click **Download Port Scanner Report** after scanning.
- **Expected Result:** PDF or text report is downloaded.

Password Strength Checker - Test Cases

TC006: Strong Password Check

- **Steps:**

1. Enter: P@ssW0rd123!.
2. Click **Check Strength**.

- **Expected Result:** Displays message like “Strong password”.

TC007: Weak Password Check

- **Steps:**

1. Enter: 123.
2. Check strength.

- **Expected Result:** Message: “Weak password”.

TC008: Empty Field Submission

- **Expected Result:** Validation error prompting input.

TC009: Special Characters Only

- **Input:** !@#\$%
- **Expected Result:** “Weak password” message shown.

Hash Generator - Test Cases

TC010: Generate Valid Hashes

- **Steps:**
 1. Input: hello world.
 2. Click **Generate Hashes**.
- **Expected Result:** Hashes like MD5, SHA-256 displayed.

TC011: Empty Input

- **Expected Result:** Error prompting for input text.

TC012: Long Input Text

- **Steps:** Paste long string (10,000+ chars).
- **Expected Result:** Hashes are generated without UI crash.

TC013: Report Download

- **Steps:** Click **Download Hash Generator Report**.
- **Expected Result:** Download contains original text and hash results.

Vulnerability Scanner - Test Cases**TC014: Valid URL Scan**

- **Input:** https://example.com.
- **Expected Result:** Vulnerabilities listed if detected.

TC015: Invalid URL Format

- **Input:** example_com.
- **Expected Result:** Show error "Invalid URL".

TC016: Empty URL Field

- **Expected Result:** Alert for required field.

TC017: Report Download

- **Steps:** After scan, click **Download Vulnerability Scanner Report**.
- **Expected Result:** Report is downloaded in PDF/CSV format.

Password Generator - Test Cases**TC018: Generate Password Successfully**

- **Steps:**
 1. Click **Generate Password**.
- **Expected Result:** Strong password shown in text field.

TC019: Generated Password Criteria

- **Expected Result:** Password must include uppercase, lowercase, number, symbol.

TC020: Report Download

- **Steps:** After generation, download report.
- **Expected Result:** PDF contains generated password with timestamp.

Sample Test Case Report Template

Test Case ID	Test Scenario	Preconditions	Test Steps	Expected Result	Actual Result
TC001	Port Scanner - Successful Scan	User is on the Port Scanner page	1. Enter valid target host 2. Enter valid port range 3. Click "Scan Ports"	Open ports within the specified range are listed correctly	Successful
TC002	Port Scanner - Invalid Host	User is on the Port Scanner page	1. Enter invalid or unreachable host 2. Enter port range 3. Click "Scan Ports"	Appropriate error message is displayed	Error Message
TC003	Password Strength Checker - Valid	User is on the Password Strength Checker page	1. Enter a strong password 2. Click "Check Strength"	Strength level (e.g., Strong) is displayed	Successful
TC004	Password Strength Checker - Weak	User is on the Password Strength Checker page	1. Enter a weak password (e.g., "123") 2. Click "Check Strength"	Warning or "Weak" message displayed	Successful
TC005	Hash Generator - Generate Hashes	User is on the Hash Generator page	1. Enter valid text 2. Click "Generate Hashes"	Hash values (MD5, SHA1, SHA256, etc.) are displayed	Successful
TC006	Hash Generator - Empty Input	User is on the Hash Generator page	1. Leave input blank 2. Click "Generate Hashes"	Error or warning for empty input	Error Message
TC007	Vulnerability Scanner - Valid URL	User is on the Vulnerability Scanner page	1. Enter a valid URL 2. Click "Scan"	Scanner returns list of potential vulnerabilities	Successful
TC008	Vulnerability Scanner - Invalid URL	User is on the Vulnerability Scanner page	1. Enter malformed or empty URL 2. Click "Scan"	Appropriate error message is displayed	Error Message
TC009	Password Generator - Generate Password	User is on the Password Generator page	1. Click "Generate Password"	A secure, random password is generated and displayed	Successful
TC010	Report Download - Port Scanner	User has performed a port scan	1. Click "Download Port Scanner Report"	Report is downloaded successfully in expected format	Successful
TC011	Report Download - Hash Generator	User has generated hashes	1. Click "Download Hash Generator Report"	Report is downloaded successfully	Successful
TC012	Report Download - Vulnerability Scanner	User has scanned a website	1. Click "Download Vulnerability Scanner Report"	Report is downloaded successfully	Successful
TC013	Report Download - Password Generator	User has generated a password	1. Click "Download Password Generator Report"	Report is downloaded successfully	Successful

14.SUPPORTING INFORMATION

The implementation of the **CyberSafe Tools** project represents a significant step forward in enhancing cybersecurity awareness and protection mechanisms for students, educators, and institutions. Designed to simulate real-world security tasks and provide actionable insights, this toolkit integrates several powerful functionalities such as Port Scanning, Password Strength and Generation, Hashing, and Vulnerability Scanning. Below are the key supporting elements that contributed to the effectiveness and relevance of this solution:

Integration with Existing Environments

The CyberSafe Tools application is built to integrate smoothly with institutional or organizational digital environments. Whether accessed through local hosting or internal networks, the toolkit functions independently without disrupting existing infrastructure. It can complement current security awareness systems, academic platforms, or IT management workflows, offering essential insights without requiring invasive changes.

Modular & Customizable Features

A key strength of the toolkit is its **modular architecture**, where each security feature (e.g., port scanner, hash generator, vulnerability scanner) operates as an independent yet cohesive unit. This design supports easy expansion and customization — future modules such as phishing simulators or forensic analysis tools can be added without overhauling the existing system. This ensures the toolkit stays relevant and adapts to evolving cybersecurity education needs.

User Training & Support

To ensure effective use, CyberSafe Tools prioritizes user-friendly design and ease of use. Each module features intuitive input forms, helpful error messages, and result interpretations. Educators and students can quickly grasp basic cybersecurity concepts while interacting with real-time data. Additionally, documentation and built-in help can guide users, while support can be scaled up for institutional deployments.

Security & Compliance

Security is a core principle of the project. While the application is designed primarily for educational use, it includes essential **security best practices**: input validation, encrypted local storage for generated reports, access logging (if enabled), and regular security checks. These measures help prevent misuse or exploitation, while compliance with common web security standards (such as OWASP Top 10) enhances trust and reliability.

Scalability & Flexibility

The toolkit is lightweight and can be scaled from individual machines to lab-wide or campus-wide use. Institutions can host it on internal servers or the cloud, depending on their infrastructure. Its modular backend and decoupled frontend allow seamless updates or UI enhancements without downtime. This flexibility ensures long-term viability as needs grow or change.

Stakeholder Engagement

During development, active feedback from cybersecurity students, faculty members, and IT professionals shaped the direction and features of the toolkit. This collaboration ensured that the application remains both educationally valuable and technically relevant. Their insights guided the prioritization of features like detailed port scan reports, password policy feedback, and actionable vulnerability summaries.

Future-Proofing

CyberSafe Tools is built with sustainability in mind. Leveraging modern web frameworks (e.g., Flask for backend, responsive frontend designs), the project supports future enhancements such as AI-based threat detection, secure file analysis, or integration with institutional SIEM tools. Its design encourages continuous improvement, ensuring the toolkit remains an adaptable asset in the cybersecurity landscape.

.