**1. INTRODUCTION**

The Cyber Security Toolkit is a lightweight web-based application designed to provide essential tools for enhancing awareness, practice, and learning in the domain of cybersecurity. With the increasing number of threats and attacks on digital platforms, it has become crucial for individuals and organizations to use simple yet effective security tools. This project integrates functionalities such as user authentication (registration, login, and logout), a main toolkit dashboard, and security-related modules, packaged in an easy-to-use interface. The project has been developed using HTML, CSS, and JavaScript and is deployed on GitHub and Netlify for accessibility.

**1.1 Background**

Cybersecurity is an essential field as digital systems continue to face threats such as malware, phishing, and weak authentication mechanisms. Traditional standalone tools often require technical expertise or paid subscriptions.  
The Cyber Security Toolkit was developed to provide a simple, web-based application for authentication and user security awareness, while remaining lightweight and easily deployable on platforms such as GitHub and Netlify.

**1.2 Objectives**

* Build a user-friendly web toolkit for authentication.
* Enable registration, login, and logout features securely.
* Style the UI with a unified CSS theme to enhance user experience.
* Deploy the system on GitHub and Netlify for global accessibility.
* Provide scope for testing through Selenium automation.

**1.3 Purpose, Scope & Applicability**

**1.3.1 Purpose**

The purpose of this project is to design a lightweight cybersecurity learning tool with user authentication, hosted online for easy access, and serving as a base for further extension.

**1.3.2 Scope**

* Browser-based system (no heavy backend).
* Covers authentication lifecycle (Register → Login → Dashboard → Logout).
* Uses LocalStorage instead of JSON/database.
* Educational and demonstrational use, not enterprise-level.

**1.3.3 Applicability**

* Students learning the basics of web security concepts.
* Beginners practicing UI testing using Selenium.
* Showcasing deployment workflow with GitHub + Netlify.

**1.4 Achievement**

* Successfully created a working authentication system (login, register, logout).
* Unified style.css applied across all pages.
* Removed unnecessary JSON dependency, making the project fully frontend-based.
* Successfully deployed to GitHub and Netlify.
* Prepared project for Selenium-based testing.

**1.5 Organization of Reports**

This report is structured as:

* Introduction
* Technologies used
* Requirements and analysis
* System design
* Implementation & testing
* Results and discussions
* Conclusion and future work

**2. Survey of Technologies**

**2.1 Awareness**

Cybersecurity awareness is the foundation of digital protection. With the rising cases of phishing, data breaches, and identity theft, individuals and organizations need simple and accessible tools to practice safe digital behavior. Most existing cybersecurity solutions are either too advanced for beginners or limited to professional environments. The Cyber Security Toolkit bridges this gap by providing a beginner-friendly platform that demonstrates fundamental authentication and security concepts. This enhances awareness by showing how login, registration, and logout mechanisms work securely and how users can interact with a safe application.

**2.2 Existing System**

The existing systems for cybersecurity awareness and tools include:

* **Commercial Security Suites (e.g., Norton, McAfee, Kaspersky):** Provide professional-grade security but are costly and not designed for educational or project-based demonstrations.
* **Open-Source Tools (e.g., OWASP Juice Shop, Wireshark):** Effective for professional learning but often too complex for beginners.
* **Basic Authentication Templates:** Several small web applications provide login/logout functionality but lack dedicated customization and integration for cybersecurity training purposes.

**Limitations of existing systems:**

* Lack of beginner-focused interfaces.
* Over-reliance on backend and API integrations (making them harder for newcomers to deploy).
* Limited scope for student-level demonstration projects.

**2.3 Proposed System**

The proposed system — Cyber Security Toolkit — addresses these limitations by offering a lightweight, web-based platform with:

* A user-friendly interface (HTML, CSS, JS).
* Authentication system (registration, login, logout).
* Dashboard (main page) for toolkit interactions.
* Deployment using GitHub + Netlify to make it globally accessible without backend servers.
* Future scope for integrating advanced security tools like password encryption, input validation, or phishing simulators.

This system is ideal for academic projects, awareness workshops, and prototype development without requiring heavy backend systems.

**2.4 Requirement Analysis**

Requirement analysis is crucial for understanding the needs of the system before implementation. Since this is a frontend-only project, the requirements are minimal but well-structured.

**2.4.1 Hardware Requirements**

* Processor: Minimum Intel i3 (or equivalent)
* RAM: 4GB or higher
* Storage: 200MB free disk space
* Display: 1024x768 resolution (or higher)
* Internet: Required for GitHub access and Netlify deployment

(The system is lightweight and can run on any modern computer or laptop.)

**2.4.2 Software Requirements**

* Operating System: Windows / Linux / macOS
* Frontend Tools:
  + HTML5 (for structure)
  + CSS3 (for styling)
  + JavaScript (for interactivity)
* Version Control & Hosting: Git, GitHub
* Deployment: Netlify
* Testing Framework: Selenium (for automated browser testing)
* Editor: VS Code or any text editor supporting web development

**2.5 Justification of Selection of Technology**

The choice of technologies for this project is based on simplicity, accessibility, and deployment ease:

1. **HTML5 & CSS3:**
   * Standard, lightweight, and supported on all browsers.
   * Allows responsive and modern UI design.
2. **JavaScript (Vanilla JS):**
   * No need for complex backend or frameworks.
   * Provides interactivity (login validation, dynamic content updates).
3. **GitHub & Git Bash:**
   * Facilitates version control and collaboration.
   * Ensures code is safely stored and shareable.
4. **Netlify Deployment:**
   * Free, simple, and efficient platform for hosting static web apps.
   * Automatic build and deploy directly from GitHub.
5. **Selenium Testing:**
   * Automated testing ensures correctness of login, registration, and logout functionality.
   * Helps simulate user interaction without manual repetitive testing.

This combination ensures that the Cyber Security Toolkit remains lightweight, easily deployable, and accessible for learning purposes, while still allowing scope for expansion into advanced features later.

**3. REQUIREMENTS AND ANALYSIS**

**3.1 Problem Definition**

**Existing System**

* Many authentication demos require backends (PHP, Node.js, APIs).
* Deployment often requires additional setup.

**Proposed System**

* A static, frontend-only authentication system using LocalStorage.
* Lightweight and easy to deploy on Netlify.

**3.2 Requirement Specification**

* **Functional Requirements**: Register new users, authenticate login, logout, validate password rules.
* **Non-functional Requirements**: Responsive design, lightweight performance, maintainability.

**3.3 Planning and Scheduling (Gantt Overview)**

* Week 1: UI design (HTML, CSS).
* Week 2: JavaScript integration.
* Week 3: Testing with Selenium.
* Week 4: Deployment via GitHub & Netlify.
* Week 5: Documentation.

**3.4 Hardware and Software**

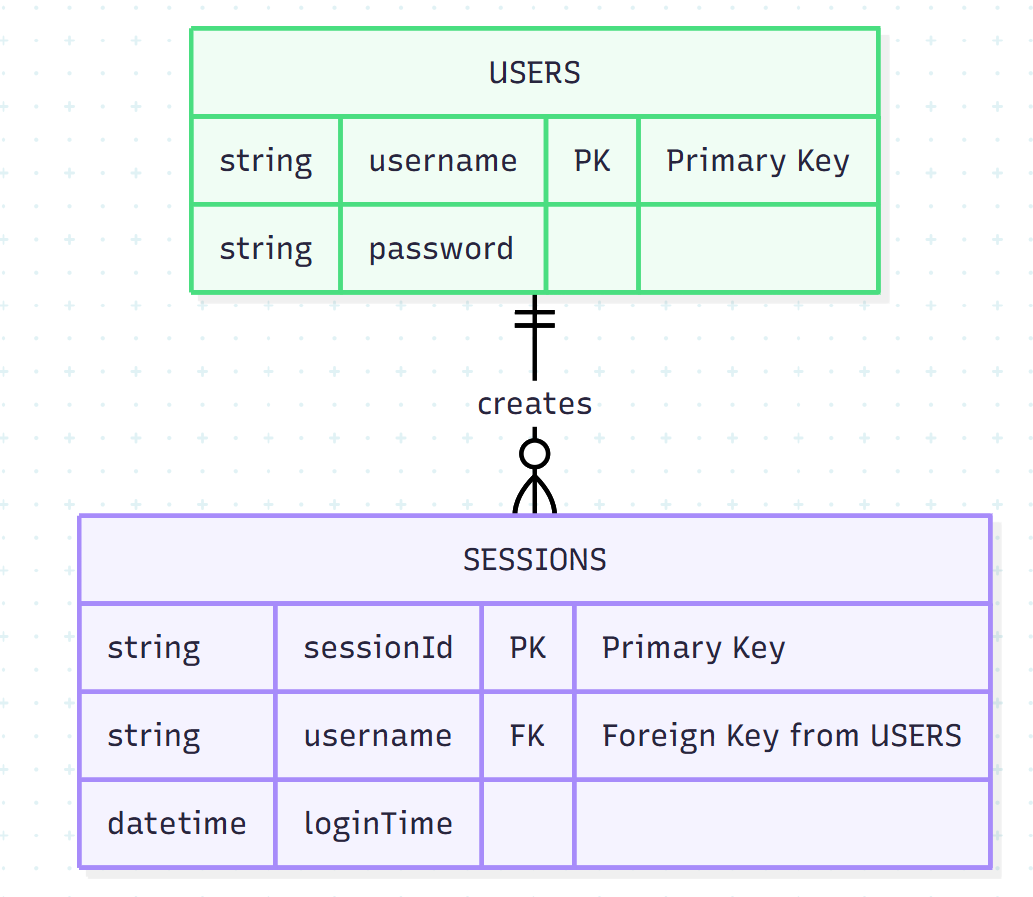
* **Hardware**: 4GB RAM, standard laptop/desktop.
* **Software**: VS Code, Git Bash, GitHub, Netlify, Selenium framework.

**3.5 Preliminary Product**

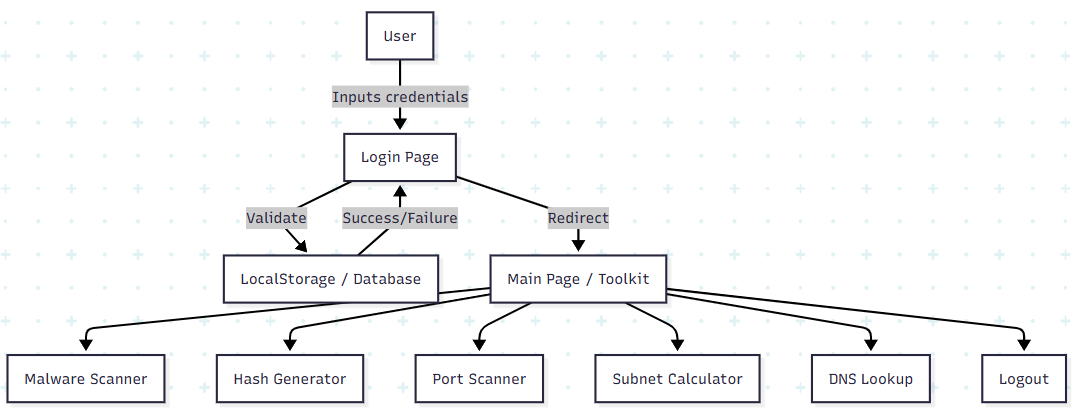
* Authentication system with login, register, logout.
* CSS styled UI.
* Deployed online and testable.

**3.6 Conceptual Models**

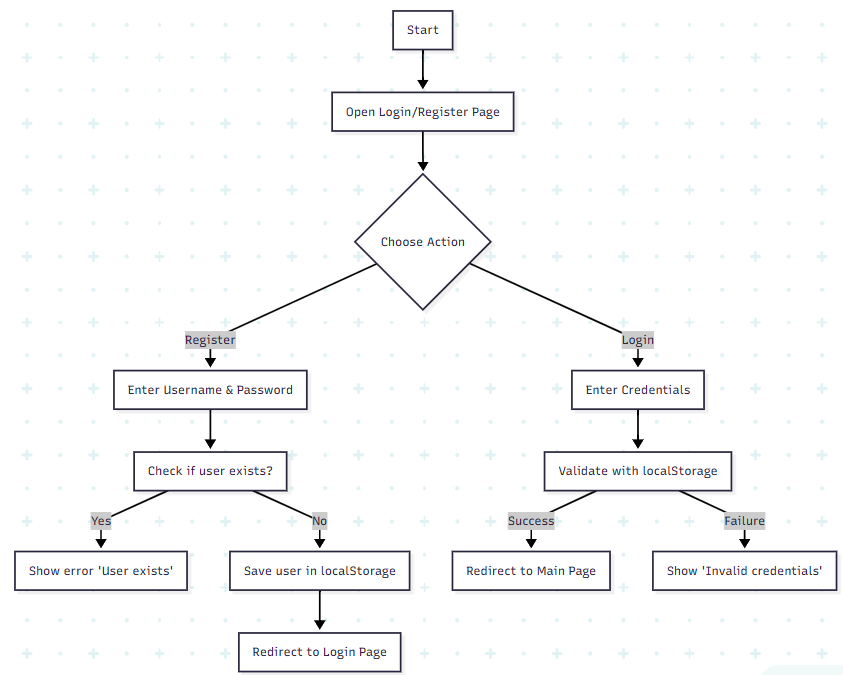
**3.6.1 ER Diagram**

****

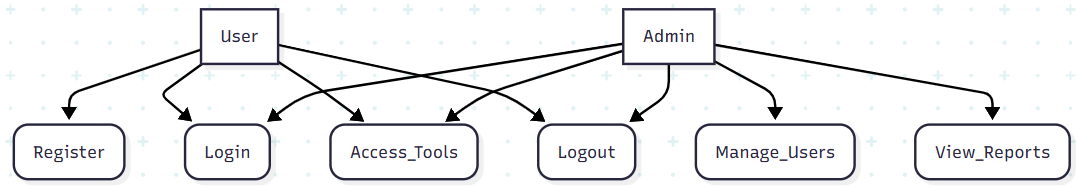
**3.6.2. Data Flow Diagram (DFD Level 0)**



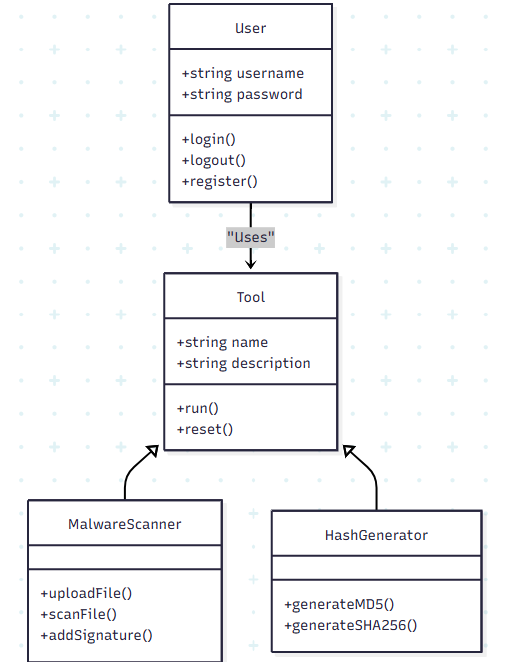
**3.6.3. Flowchart: Login & Registration**



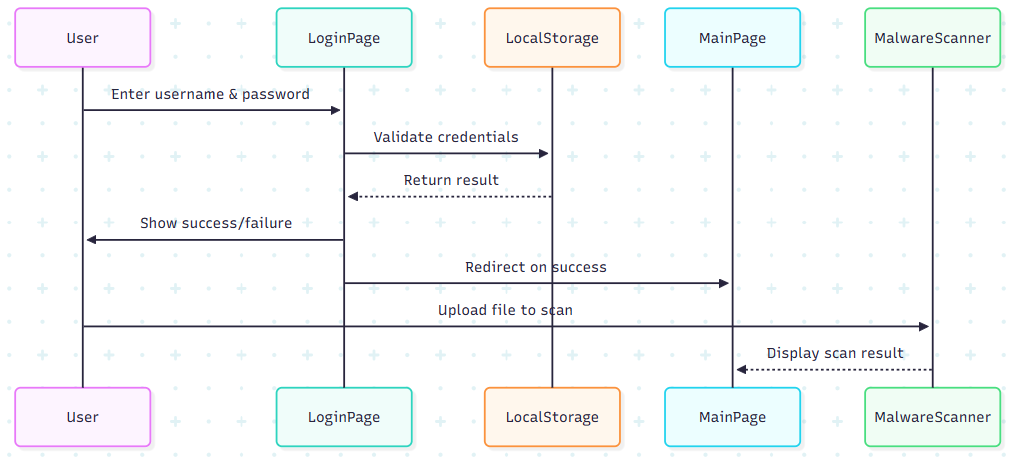
**3.6.4. Use Case Diagram**

****

**3.6.5. Class Diagram**

****

**3.6.6. Sequence Diagram**

****

**CHAPTER 4: SYSTEM DESIGN**

System design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. For the Cyber Security Toolkit, the system design focuses on simplicity, modularity, and security practices since the project is built as a learning-oriented and awareness-driven platform.

**4.1 Basic Module**

The system is divided into fundamental modules to ensure modularity and ease of understanding:

1. **Default Page (Index.html)**
   * Entry point of the application.
   * Provides the user with access to different tools within the toolkit.
   * Displays project title and navigation options.
2. **Registration Page (register.html)**
   * Allows new users to register with a unique username and password.
   * Data is stored locally using browser localStorage.
   * Prevents duplicate registration.
3. **Login Page (login.html)**
   * Provides secure login for registered users.
   * Validates credentials against stored data.
   * Redirects authenticated users to the main page.
4. **Main Page (index.html + index.js)**
   * Core of the Cyber Security Toolkit.
   * Displays authenticated user details.
   * Provides toolkit functionalities (e.g., password strength check, input validations).
   * Logout option is available.
5. **Logout Page (logout.html)**
   * Ends the session by clearing user data from localStorage.
   * Redirects back to the login page.

These modules are interlinked to maintain smooth navigation and proper session management.

**4.2 Data Design**

Data design ensures how information is stored, accessed, and maintained in the system.

* **Data Source:** localStorage (client-side storage).
* **Entities:**
  + User: username, password.
  + Session: currently logged-in user.
* **Data Flow:**
  + Registration stores username: password in localStorage.
  + Login retrieves and verifies this data.
  + Logout removes the session variable currentUser.
* **Schema Representation:**Since no external DBMS is used, schema is represented in JSON-like format:

{

"users": {

"username1": "password1",

"username2": "password2"

},

"currentUser": "username1"

}

**4.3 Procedural Design**

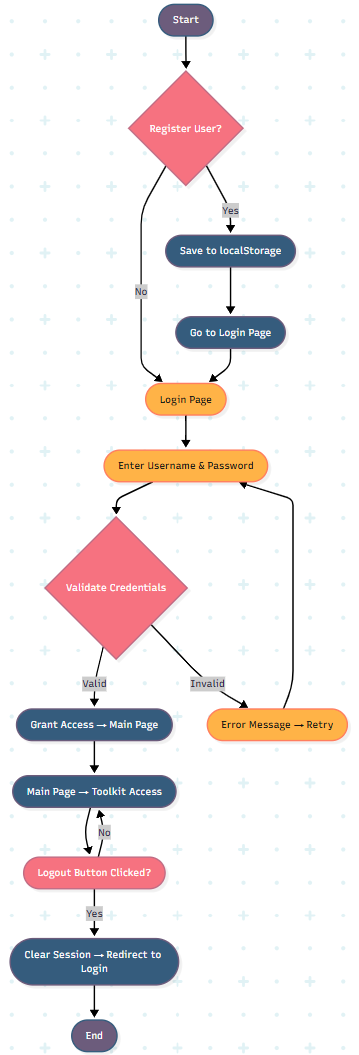
Procedural design focuses on how the system’s operations are carried out step by step. It defines the flow of control, the sequence of operations, and how different modules interact to achieve the system’s functionality. For this project, procedural design ensures that:

* User actions such as registration, login, and logout follow a defined sequence.
* Tools on the main page (malware scanner, password checker, hash generator, subnet calculator, DNS lookup) execute in an organized, predictable manner.
* Input validation and error handling are integrated into each process to maintain system reliability.
* Session management and local storage operations are clearly mapped to user actions.

By clearly defining the procedure, the system becomes easier to implement, test, and maintain, and ensures consistent results for all users.

**4.3.1 Logic Diagram**

The logic flow of the system can be represented as:

****

**4.3.2 Data Structures**

The project uses JavaScript objects and arrays as primary data structures.

1. **User Data (Object Map):**

let users = {

"john": "1234",

"alice": "abcd"

};

1. **Session Data:**

let currentUser = "john";

localStorage.setItem("currentUser", currentUser);

1. **Toolkit Functions Data:**  
   (Example: password strength evaluation using regex patterns)

function checkPasswordStrength(password) {

let strong = /^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[@$!%\*?&])/;

return strong.test(password);

}

**4.4 User Interface Design**

User Interface (UI) is designed with simplicity and accessibility in mind.

* **Login & Registration Pages:**Clean form layouts, styled with style.css, blue-themed container with input fields and buttons.
* **Main Page:**Displays project title on top (blue strip).  
  Logout button and "Signed in as ..." positioned at top-right corner.  
  Toolkit functions are arranged in cards or sections for easy navigation.
* **Logout Page:**Simple confirmation message with a styled "Login Again" button.

**Design Principles Applied:**

* Consistency in color scheme (blue + white).
* Responsive design for various screen sizes.
* Clear text with appropriate font size for readability.
* Proper use of hover effects to improve interactivity.

**4.5 Security Issues**

Although the system is a prototype for learning, certain security issues are acknowledged:

1. **Plaintext Passwords:**
   * Passwords are stored as plain text in localStorage.
   * In a production system, passwords must be hashed (e.g., SHA-256, bcrypt).
2. **Client-Side Storage Risks:**
   * localStorage can be accessed by anyone with access to the browser.
   * No real server-side authentication is present.
3. **Session Management:**
   * Basic session handling with currentUser.
   * Vulnerable to manipulation by changing localStorage manually.
4. **No HTTPS Enforcement (on local):**
   * Since deployed on Netlify, HTTPS is available, but local file usage may lack encryption.
5. **Potential Enhancements:**
   * Use JWT tokens for authentication if extended with backend.
   * Encrypt sensitive data before storing.
   * Add CAPTCHA on login/registration to prevent brute force.
   * Perform input sanitization to avoid malicious scripts.

**5. IMPLEMENTATION AND TESTING**

**5.1 Implementation Approaches**

* Entirely frontend-based system.
* Deployed via GitHub → Netlify.
* Modular JS functions for register/login/logout.

**5.2 Coding Details and Efficiency**

* JavaScript ensures minimal complexity (O(1) for storage retrieval).
* CSS external file for maintainability.

**5.3 Beta Testing (Selenium)**

* Planned to test:
  + Register page loads & stores credentials.
  + Login authentication flow.
  + Logout clears session.
* Test scripts automate these flows across Chrome.

**5.4 Coding**

* login.html: Form + validation.
* register.html: Registration + storage.
* index.js: Central script with logic.
* style.css: Global styles.

**6. Results and Discussions**

This chapter evaluates the outcomes of the project, the functionalities implemented, and discusses the observations, limitations, and insights.

**6.1 Summary of Results**

The Cyber Security Toolkit provides a collection of security-related utilities for learning and awareness purposes. The following functionalities were successfully implemented:

1. **User Authentication:**
   * Registration, login, and logout mechanisms using localStorage.
   * Unique usernames enforced to prevent duplicates.
   * Session handling ensures only authenticated users can access the toolkit.
2. **Toolkit Functionalities:**
   * **Password Strength Checker**: Validates password strength based on character types and length.
   * **MD5 Hash Cracker:** Accepts a wordlist to find matching plaintext for a given MD5 hash.
   * **SHA256 Hash Generator:** Converts user input text into a secure SHA256 hash.
   * **Subnet / CIDR Calculator:** Calculates network address, broadcast, usable hosts, and total addresses from IP and mask/CIDR.
   * **Port Scanner (Browser-based simulation):** Scans common ports with fetch probes.
   * **Network Sniffer (Simulated):** Generates random packet logs to illustrate packet capture.
   * **DNS Lookup:** Queries DNS records via Google DNS-over-HTTPS API.
3. **UI/UX Results:**
   * Clean, consistent, and responsive interface using a blue-themed design.
   * Clear layout with containers and cards for each tool, improving usability.
   * Logout and session info displayed at the top-right corner for better navigation.

**6.2 Discussion**

* **Effectiveness:**
  + The system provides a simple, interactive environment to understand basic cybersecurity concepts.
  + All tools are fully functional and accessible through a browser without additional installations.
* **Limitations:**
  + Passwords are stored in plaintext (localStorage) and are not encrypted.
  + No server-side backend, so functionalities are limited to client-side simulations.
  + Network tools (Port Scanner, Network Sniffer) are simulated due to browser restrictions and security policies.
* **Insights:**
  + Implementing a full cybersecurity toolkit with only frontend technologies highlighted the constraints and strengths of client-side systems.
  + User session management and modular design provided a clean structure and reusability of code.
* **Future Testing Considerations:**
  + Selenium automation testing can validate registration, login, logout, and tool functionalities.
  + Cross-browser testing ensures UI consistency across devices.

**7. Conclusion and Future Work**

**7.1 Conclusion**

The Cyber Security Toolkit project successfully achieves its objectives:

* Provides a functional platform for practicing and understanding basic cybersecurity tools.
* Demonstrates user authentication, modular interface design, and interactive tools.
* Offers a responsive and visually appealing UI with intuitive navigation.
* Deploying on GitHub and Netlify ensures accessibility and ease of testing.

The project serves as a foundation for educational purposes, helping users understand concepts like hashing, password strength, IP addressing, DNS queries, and network probing.

**7.2 Future Work**

While the current implementation focuses on frontend functionality, there are several potential improvements:

1. **Backend Integration:**
   * Implement a server-side backend using Node.js, Python Flask, or Django.
   * Store user credentials securely in a database with encryption.
2. **Enhanced Security:**
   * Use bcrypt or SHA256 hashing for passwords.
   * Implement proper session management with JWT tokens.
   * Add input sanitization and validation to prevent XSS attacks.
3. **Real Network Tools:**
   * Integrate real network scanning and packet sniffing using server-side scripts (Python’s Scapy or Nmap APIs).
   * Implement real-time results for port scanning and packet captures.
4. **Additional Modules:**
   * Vulnerability scanner, malware analysis simulator, or encryption/decryption utilities.
   * Email validation, phishing detection, and other educational security tools.
5. **Automated Testing:**
   * Expand Selenium test scripts to automate tool interactions and verify expected results.
   * Include unit tests for JavaScript functions (e.g., password check, hash generator).
6. **User Experience Enhancements:**
   * Dashboard customization, light/dark mode themes.
   * Enhanced tooltips and explanations for educational purposes.

**8. References**

The project references a combination of online tools, documentation, and educational resources for implementation:

1. **Mozilla Developer Network (MDN) Web Docs:**
   * HTML, CSS, JavaScript references: https://developer.mozilla.org/
2. **Pyodide Documentation:**
   * Running Python in the browser: https://pyodide.org/
3. **Google DNS-over-HTTPS API:**
   * DNS resolution using DoH: https://developers.google.com/speed/public-dns/docs/doh
4. **StackOverflow and Developer Community Resources:**
   * Implementation techniques for localStorage, session handling, and JS modules.
5. **GitHub and Netlify Documentation:**
   * Version control and deployment practices:  
     GitHub Docs  
     Netlify Docs
6. **Cybersecurity Tutorials and Online Learning Platforms:**
   * Cybersecurity concepts, hashing, port scanning, and IP addressing tutorials.
   * Example platforms: OWASP, Cybrary