**Project Report**

**Title:** Ethical Phishing Simulation (Single-Page Demo Web App)

**1. Introduction**

This project is a web-based ethical phishing simulation tool designed to raise awareness about phishing attacks and train users in identifying suspicious messages. It is built as a single self-contained HTML file with inline CSS and JavaScript, requiring no backend server.

The simulation is purely educational and ensures no real credentials or data are collected. All user inputs and results are stored locally in the browser’s localStorage, making it safe for training and classroom use.

**2. Objectives**

* To provide a hands-on phishing awareness tool for individuals and organizations.
* To allow trainers to simulate phishing campaigns in a safe environment.
* To help users practice safe reporting of phishing attempts.
* To teach recognition of red flags in phishing emails and fake login pages.

**3. Features**

**a) Campaign Creation**

* Users can create mini phishing campaigns.
* Three predefined templates:
  + Password Expiry Alert
  + Package Delivery Notice
  + Shared Document Request
* Target groups can be labeled (e.g., “Interns Batch A”).
* A preview of the fake email is generated dynamically.

**b) Simulated Login Page**

* Mimics a fake company login portal.
* Encourages entering dummy values only (e.g., user@example.com).
* Captures submissions as “training events” with password length only (never actual content).
* Provides teaching points: domain spoofing, urgency, spelling issues, and unsafe data requests.

**c) Report Phish**

* Users can report suspicious emails instead of submitting credentials.
* Options for reasons (unexpected request, suspicious link, grammar errors, etc.).
* Reports are logged locally.

**d) Results Dashboard**

* Displays summary statistics (submissions vs. reports).
* Event table shows time, type of action, template used, and details.
* Options to export results as JSON or reset all data.

**4. Technical Design**

* **Frontend Technology:** HTML, CSS, and vanilla JavaScript (no frameworks).
* **Data Storage:** Browser localStorage (safe, no server communication).
* **User Interface:**
  + Responsive layout with cards and grids.
  + Buttons styled with gradients and thematic colors.
  + Navigation uses hash-based routing (#home, #campaign, etc.).
* **Security Considerations:**
  + Explicit disclaimers to avoid using real credentials.
  + Passwords are never stored—only password length is recorded.
  + All operations run locally inside the user’s browser.

**5. Use Cases**

* Corporate training workshops on phishing awareness.
* Classroom demonstrations for cybersecurity courses.
* Self-learning tool for individuals who want to recognize phishing tactics.
* Awareness campaigns in organizations without external dependencies.

**6. Limitations**

* Only provides basic templates (not customizable beyond text).
* Works entirely offline; no analytics or centralized tracking.
* Limited to demo-level reporting (not enterprise grade).
* UI is functional but minimal, with room for polish.

**7. Future Improvements**

* Add more phishing templates (bank fraud, job scams, etc.).
* Introduce custom email template builder.
* Provide graphs/charts of results for better visualization.
* Add multi-user tracking with anonymized IDs.
* Optional server integration (for controlled enterprise environments).

**8. Hardware Interfaces**

This project runs fully in a web browser and does not require any specialized hardware. The hardware interfaces are therefore minimal and generic:

* **Client Device**:
  + Any modern device capable of running a standards-compliant web browser (desktop, laptop, tablet, or mobile phone).
* **Processor and Memory**:
  + CPU: Standard x86/x64/ARM processors (sufficient to run a browser).
  + RAM: At least 1 GB free memory recommended for smooth browser operation.
* **Storage**:
  + Uses the client’s local storage (localStorage API) for event logging; requires only a few kilobytes.
* **Input/Output Devices**:
  + Keyboard and mouse/touchscreen for interaction.
  + Display for rendering the HTML/CSS interface.

**9. Software Interfaces**

The phishing simulation is implemented entirely as a single-page HTML/JavaScript application. It relies on the following software components and APIs:

* **Operating System**: Any OS that supports a modern browser (Windows, macOS, Linux, Android, iOS).
* **Web Browser** (required):
  + Must support HTML5, CSS3, JavaScript ES6+, and Web Storage API.
  + Examples: Chrome, Firefox, Edge, Safari.
* **APIs / Libraries Used**:
  + **DOM API**: For manipulating page structure and updating results dynamically.
  + **Web Storage API (localStorage)**: For persisting campaign and event data locally.
  + **Blob & URL APIs**: For exporting results as JSON files.
* **No server dependency**: All logic executes client-side, ensuring no backend interface is required.

**10. Modules in the Code**

The code can be logically divided into several modules (even though it’s in one file):

1. **UI Module (HTML + CSS)**
   * Defines structure (header, navigation, sections).
   * Provides styles (cards, grids, buttons, forms).
   * Ensures responsive design and user interaction flow.
2. **Campaign Module**
   * Functions: startCampaign()
   * Purpose: Create and preview phishing campaign templates.
   * Stores metadata (template, target group, timestamp).
3. **Simulation Module**
   * Functions: submitPhish(), fillDummy()
   * Purpose: Fake login page for training.
   * Captures dummy input (email/password) and logs submission events.
4. **Reporting Module**
   * Function: submitReport()
   * Purpose: Lets users report suspicious messages.
   * Logs reporting events for awareness tracking.
5. **Results & Persistence Module**
   * Functions: persist(), render(), exportJSON(), resetAll()
   * Purpose: Save state to localStorage, render event tables and summary statistics, allow exporting, and reset data.
6. **Utility Module**
   * Function: nowISO()
   * Provides current timestamp formatting for event logs.

**11. Conclusion**

The Ethical Phishing Simulation project successfully demonstrates how phishing awareness training can be delivered **safely and interactively** without risking user data. It achieves its goal of raising awareness by simulating phishing emails and fake login pages while reinforcing the importance of reporting.

This single-file, self-contained design makes it **lightweight, portable, and safe**, suitable for use in classrooms, corporate workshops, or personal cybersecurity learning.