***PassShield - Password Strength Analyzer***

Project Overview

PassShield is a comprehensive password analysis tool that helps users create and maintain secure passwords. It evaluates password strength in real-time, checks against known data breaches, and provides smart suggestions for improvement.

Features

* Real-time password strength analysis
* Visual strength meter
* Smart improvement suggestions
* Data breach checking via HaveIBeenPwned API
* Secure password generator
* Responsive design for all devices

Tech Stack

* **Frontend**: HTML5, CSS3, JavaScript (ES6+)
* **Libraries**:
  + [zxcvbn.js](https://github.com/dropbox/zxcvbn) for password strength analysis
  + [Bootstrap 4](https://getbootstrap.com/) for responsive layout
* **API**: [HaveIBeenPwned](https://haveibeenpwned.com/API/v3" \t "_blank) for breach checking
* **Tools**: Figma for wireframing

Implementation

File Structure

Copy

passshield/

├── index.html # Main HTML file

├── style.css # Custom styles

├── main.js # Core functionality

├── README.md # Project documentation

└── assets/ # Additional assets

├── screenshots/ # App screenshots

└── icons/ # Favicons and other icons

***Complete Code Implementation***

***index.html :***

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>PassShield - Password Strength Analyzer</title>

<link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">

<link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;500;700&display=swap" rel="stylesheet">

<link rel="stylesheet" href="style.css">

<link rel="icon" href="assets/icons/shield-lock.svg" type="image/svg+xml">

</head>

<body>

<div class="container">

<header class="text-center my-5">

<h1 class="display-4">PassShield</h1>

<p class="lead">Your comprehensive password security toolkit</p>

</header>

<div class="card shadow-lg">

<div class="card-body">

<div class="form-group">

<label for="passwordInput">Enter your password:</label>

<div class="input-group">

<input type="password" id="passwordInput" class="form-control" placeholder="Type or generate a password">

<div class="input-group-append">

<button class="btn btn-outline-secondary" id="toggleVisibility" title="Show/Hide password">

<i class="bi bi-eye"></i> 👁️

</button>

</div>

</div>

</div>

<div class="password-strength-section">

<div class="d-flex justify-content-between mb-1">

<small>Password Strength:</small>

<small id="strengthText">Very Weak</small>

</div>

<div id="passwordStrength" class="strength-meter">

<div class="strength-bar" id="strengthBar"></div>

</div>

</div>

<div id="suggestions" class="suggestions-box mt-3"></div>

<div class="d-flex flex-wrap justify-content-between mt-4">

<button class="btn btn-primary" id="checkBreach">

<i class="bi bi-shield-lock"></i> Check for Breaches

</button>

<button class="btn btn-success" id="generatePassword">

<i class="bi bi-key"></i> Generate Strong Password

</button>

</div>

<div id="breachInfo" class="breach-info mt-3"></div>

</div>

</div>

<footer class="text-center mt-5 text-muted">

<p>PassShield &copy; 2023 | Uses zxcvbn.js and HaveIBeenPwned API</p>

</footer>

</div>

<!-- Scripts -->

<script src="https://cdnjs.cloudflare.com/ajax/libs/zxcvbn/4.4.2/zxcvbn.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/js-sha1/0.6.0/sha1.min.js"></script>

<script src="main.js"></script>

</body>

</html>

***style.css :***

:root {

--weak: #ff4d4d;

--medium: #ffcc00;

--strong: #33cc33;

--very-strong: #009933;

--suggestion-bg: #f8f9fa;

--breach-warning: #fff3cd;

}

body {

font-family: 'Roboto', sans-serif;

background-color: #f5f7fa;

color: #333;

line-height: 1.6;

}

.container {

max-width: 800px;

}

.card {

border-radius: 10px;

border: none;

}

.password-strength-section {

margin: 1.5rem 0;

}

.strength-meter {

height: 8px;

width: 100%;

background-color: #e9ecef;

border-radius: 4px;

overflow: hidden;

}

.strength-bar {

height: 100%;

width: 0;

transition: width 0.3s ease, background-color 0.3s ease;

}

.suggestions-box {

padding: 1rem;

background-color: var(--suggestion-bg);

border-radius: 5px;

border-left: 4px solid #6c757d;

}

.suggestions-box ul {

margin-bottom: 0;

padding-left: 1.2rem;

}

.breach-info {

padding: 1rem;

border-radius: 5px;

}

.breach-found {

background-color: var(--breach-warning);

border-left: 4px solid #ffc107;

}

.breach-safe {

background-color: #d4edda;

border-left: 4px solid #28a745;

}

/\* Strength colors \*/

.weak {

background-color: var(--weak);

width: 25%;

}

.medium {

background-color: var(--medium);

width: 50%;

}

.strong {

background-color: var(--strong);

width: 75%;

}

.very-strong {

background-color: var(--very-strong);

width: 100%;

}

/\* Responsive adjustments \*/

@media (max-width: 576px) {

.card-body {

padding: 1.25rem;

}

.d-flex.justify-content-between button {

width: 100%;

margin-bottom: 0.5rem;

}

}

/\* Animation for generated password \*/

@keyframes highlight {

0% { background-color: rgba(40, 167, 69, 0.2); }

100% { background-color: transparent; }

}

.password-generated {

animation: highlight 1.5s ease-out;

}

***main.js:***

document.addEventListener('DOMContentLoaded', function() {

const passwordInput = document.getElementById('passwordInput');

const strengthBar = document.getElementById('strengthBar');

const strengthText = document.getElementById('strengthText');

const suggestionsDiv = document.getElementById('suggestions');

const checkBreachBtn = document.getElementById('checkBreach');

const generatePasswordBtn = document.getElementById('generatePassword');

const breachInfoDiv = document.getElementById('breachInfo');

const toggleVisibilityBtn = document.getElementById('toggleVisibility');

// Password strength analysis

passwordInput.addEventListener('input', function() {

const password = this.value;

const result = zxcvbn(password);

updateStrengthMeter(result.score);

updateSuggestions(result.feedback, password);

});

// Toggle password visibility

toggleVisibilityBtn.addEventListener('click', function() {

const type = passwordInput.getAttribute('type') === 'password' ? 'text' : 'password';

passwordInput.setAttribute('type', type);

this.innerHTML = type === 'password' ? '👁️' : '👁️‍🗨️';

});

// Check password against breaches

checkBreachBtn.addEventListener('click', async function() {

const password = passwordInput.value;

if (!password) {

showBreachInfo('Please enter a password to check', false);

return;

}

this.disabled = true;

this.innerHTML = '<span class="spinner-border spinner-border-sm" role="status" aria-hidden="true"></span> Checking...';

try {

const isBreached = await checkPasswordBreach(password);

if (isBreached) {

showBreachInfo('⚠️ This password has been exposed in data breaches! Do not use it.', true);

} else {

showBreachInfo('✅ This password has not been found in any known breaches.', false);

}

} catch (error) {

showBreachInfo('❌ Error checking breaches. Please try again later.', true);

console.error('Breach check error:', error);

} finally {

this.disabled = false;

this.innerHTML = '<i class="bi bi-shield-lock"></i> Check for Breaches';

}

});

// Generate strong password

generatePasswordBtn.addEventListener('click', function() {

const password = generateStrongPassword();

passwordInput.value = password;

passwordInput.dispatchEvent(new Event('input'));

passwordInput.classList.add('password-generated');

setTimeout(() => passwordInput.classList.remove('password-generated'), 1500);

});

// Helper functions

function updateStrengthMeter(score) {

// Reset classes

strengthBar.className = 'strength-bar';

// Update based on score (0-4)

const strengthClasses = ['weak', 'medium', 'strong', 'very-strong'];

const strengthLabels = ['Very Weak', 'Weak', 'Medium', 'Strong', 'Very Strong'];

if (score < strengthClasses.length) {

strengthBar.classList.add(strengthClasses[score]);

} else {

strengthBar.classList.add('very-strong');

}

strengthText.textContent = strengthLabels[score];

strengthText.style.color = getStrengthColor(score);

}

function getStrengthColor(score) {

const colors = ['#ff4d4d', '#ff4d4d', '#ffcc00', '#33cc33', '#009933'];

return colors[score];

}

function updateSuggestions(feedback, password) {

let suggestions = [];

// Add library suggestions

if (feedback.suggestions.length > 0) {

suggestions = suggestions.concat(feedback.suggestions);

}

// Add length suggestion if needed

if (password.length < 8) {

suggestions.push('Make your password longer (at least 12 characters recommended)');

}

// Add warning if password is too common

if (feedback.warning) {

suggestions.unshift(feedback.warning);

}

// Display suggestions

if (suggestions.length > 0) {

suggestionsDiv.innerHTML = `

<h6>Suggestions to improve your password:</h6>

<ul>

${suggestions.map(s => `<li>${s}</li>`).join('')}

</ul>

`;

} else if (password.length > 0) {

suggestionsDiv.innerHTML = '<p>Great password! No suggestions for improvement.</p>';

} else {

suggestionsDiv.innerHTML = '';

}

}

async function checkPasswordBreach(password) {

// Hash the password using SHA-1

const sha1Hash = sha1(password).toUpperCase();

const prefix = sha1Hash.substring(0, 5);

const suffix = sha1Hash.substring(5);

try {

const response = await fetch(`https://api.pwnedpasswords.com/range/${prefix}`);

if (!response.ok) throw new Error('API request failed');

const data = await response.text();

return data.includes(suffix);

} catch (error) {

console.error('Error checking password breach:', error);

throw error;

}

}

function showBreachInfo(message, isWarning) {

breachInfoDiv.textContent = message;

breachInfoDiv.className = 'breach-info';

breachInfoDiv.classList.add(isWarning ? 'breach-found' : 'breach-safe');

}

function generateStrongPassword() {

const lowercase = 'abcdefghijkmnpqrstuvwxyz';

const uppercase = 'ABCDEFGHJKLMNPQRSTUVWXYZ';

const numbers = '23456789';

const symbols = '!@#$%^&\*()\_+-=[]{}|;:,.<>?';

const allChars = lowercase + uppercase + numbers + symbols;

let password = '';

// Ensure at least one character from each set

password += getRandomChar(lowercase);

password += getRandomChar(uppercase);

password += getRandomChar(numbers);

password += getRandomChar(symbols);

// Fill the rest randomly

for (let i = 0; i < 12; i++) {

password += getRandomChar(allChars);

}

// Shuffle the password

return shuffleString(password);

}

function getRandomChar(charSet) {

return charSet[Math.floor(Math.random() \* charSet.length)];

}

function shuffleString(str) {

const array = str.split('');

for (let i = array.length - 1; i > 0; i--) {

const j = Math.floor(Math.random() \* (i + 1));

[array[i], array[j]] = [array[j], array[i]];

}

return array.join('');

}

});

How to Use

1. **Password Strength Check**:
   * Type a password in the input field
   * The strength meter will update in real-time
   * Suggestions for improvement will appear below
2. **Breach Check**:
   * Click "Check for Breaches" to see if your password has been exposed in known data breaches
3. **Password Generator**:
   * Click "Generate Strong Password" to create a secure random password

Deployment

To deploy this project:

1. Clone the repository
2. Open index.html in a web browser
3. Alternatively, host the files on any static web hosting service (GitHub Pages, Netlify, Vercel, etc.)

API Usage Notes

The application uses the HaveIBeenPwned API v3 with k-anonymity model for checking password breaches. No actual passwords are sent to the API - only the first 5 characters of the SHA-1 hash are transmitted.

License

This project is open-source and available under the MIT License.

Future Enhancements

* Add password history/remember feature (local storage only)
* Implement password sharing via secure links
* Add multi-language support
* Create browser extension version

This complete implementation provides a robust password analysis tool with all the features outlined in your plan. The code is well-organized, commented, and ready for deployment to a GitHub repository. The project includes:

* Clean, responsive UI with Bootstrap
* Comprehensive password strength analysis
* Data breach checking
* Secure password generation
* Helpful suggestions for improvement
* Proper error handling