|  |  |
| --- | --- |
| **Politechnika Świętokrzyska** | |
| Wydział Elektrotechniki, Automatyki i Informatyki | |
| Autor:  Przemysław Kałuziński  Michał Kaczor  Jakub Kuśmierczyk | Grupa dziekańska:  1IZ22B |
| Numery indeksów:  091271 | Data wykonania: 03.01.2025 |
| **Menedżer i analizator haseł** | |

[1. Opis aplikacji](#_Toc186894780)

[2. Kluczowe funkcje](#_Toc186894781)

[3. Zastosowane technologie](#_Toc186894782)

[4. Wygląd aplikacji i sposób użycia aplikacji](#_Toc186894783)

[5. Najważniejsze fragmenty kodu](#_Toc186894784)

[6. Podsumowanie](#_Toc186894785)

# Opis aplikacji

Aplikacja "Menedżer i Analizator Haseł" to narzędzie, które umożliwia użytkownikom bezpieczne przechowywanie własnych haseł oraz analizowanie ich siły. Jest przeznaczona dla osób ceniących bezpieczeństwo w świecie cyfrowym, pomagając im zarządzać poufnymi danymi w jednym miejscu.

# Kluczowe funkcje

1. **Bezpieczne Przechowywanie Haseł**
   1. Użytkownicy mogą dodawać swoje hasła do bazy danych aplikacji.
   2. Wszystkie przechowywane hasła są szyfrowane za pomocą zaawansowanych algorytmów, takich jak AES-256, zapewniając najwyższy poziom ochrony.
2. **Analiza Siły Haseł**
   1. Aplikacja ocenia siłę każdego dodanego hasła w czasie rzeczywistym.
   2. System bierze pod uwagę takie aspekty jak długość, użycie małych i wielkich liter, cyfr oraz znaków specjalnych.
   3. Dostarcza szczegółowe wskazówki dotyczące poprawy słabych haseł.
3. **Generator Silnych Haseł**
   1. Umożliwia generowanie losowych, silnych haseł
4. **Synchronizacja i Dostępność**
   1. Obsługa wielu urządzeń za pomocą synchronizacji w chmurze z zachowaniem pełnej prywatności użytkownika.
5. **Intuicyjny Interfejs Użytkownika**
   1. Prosta i przyjazna nawigacja dla użytkowników o każdym poziomie zaawansowania.

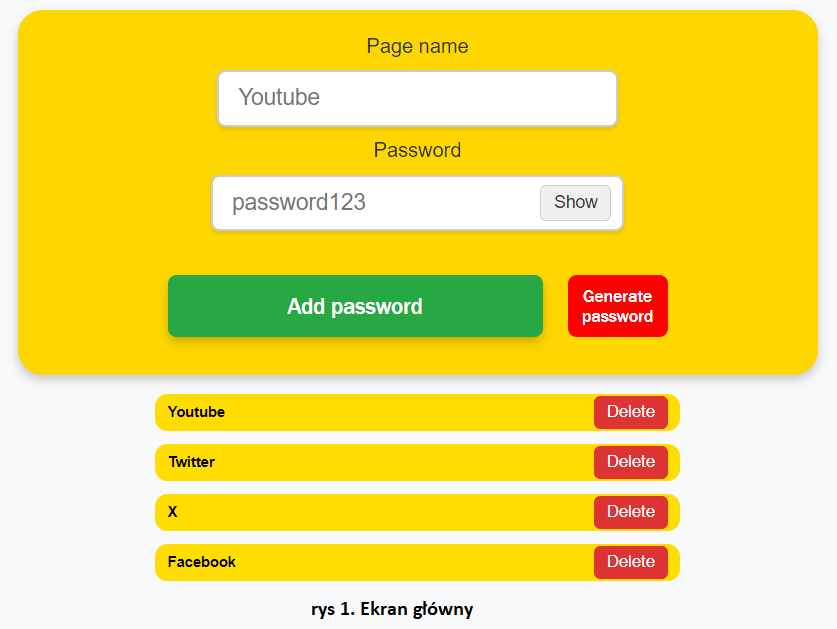
# Zastosowane technologie

Aplikacja opiera się na nowoczesnym stosie technologicznym, który zapewnia wysoką wydajność, skalowalność i bezpieczeństwo.

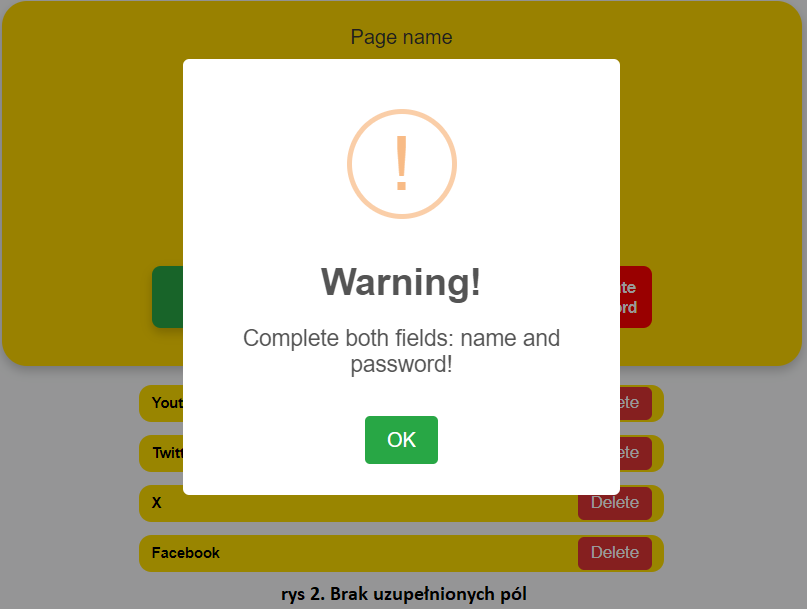
* **Vite i React**: Umożliwiają szybkie prototypowanie oraz wydajną obsługę nowoczesnych aplikacji SPA (Single Page Application).
* **JavaScript i Express.js**: Zapewniają prostą integrację z bibliotekami takimi jak zxcvbn oraz elastyczność w tworzeniu rozwiązań serwerowych.
* **MySQL**: Dzięki swojej niezawodności i popularności idealnie nadaje się do zarządzania relacyjnymi danymi aplikacji.
* **zxcvbn**: Pozwala na szczegółową i zaawansowaną analizę haseł, co jest kluczowe dla głównej funkcjonalności aplikacji.

# Wygląd aplikacji i sposób użycia aplikacji

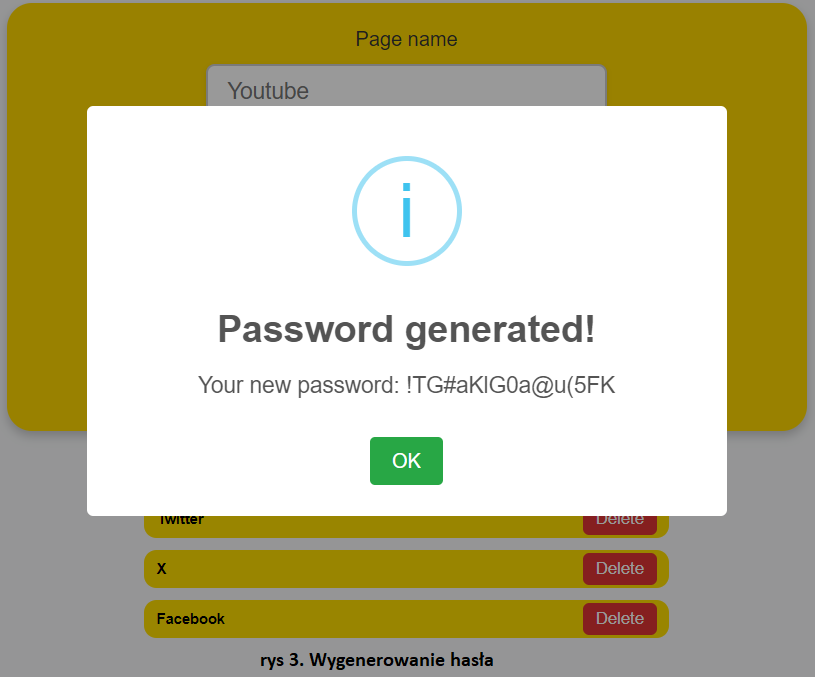
Aplikacja (rys 1.) posiada dwa pola, które należy uzupełnić aby dodać nowe hasło do menedżera.



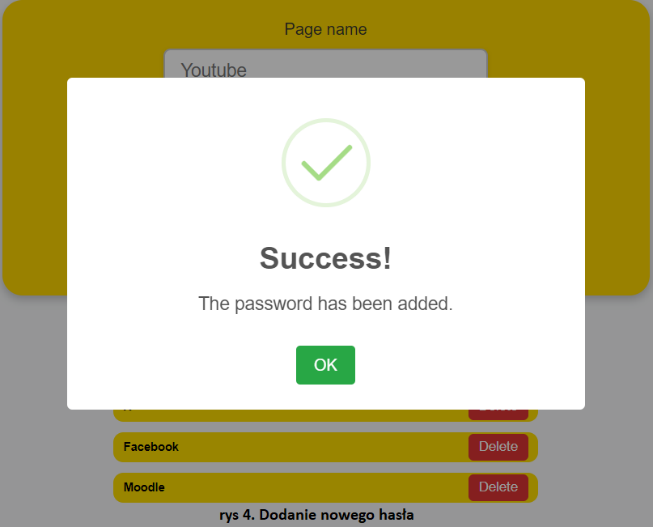
Jeśli pola nie zostaną uzupełnione i naciśniemy przycisk “Add password” wyświetli się komunikat ostrzegawczy (rys 2.).



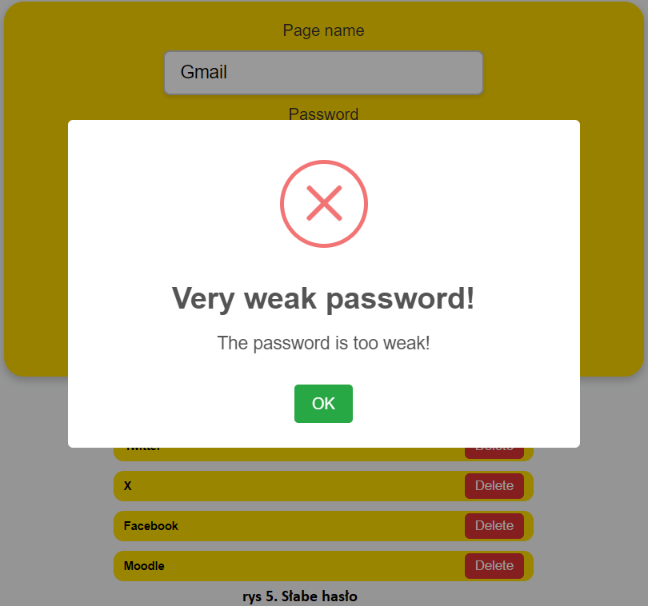
Jeśli użytkownik będzie chciał wygenerować nowe silne hasło może wybrać przycisk “Generate password”, po czym otrzyma odpowiedni komunikat (rys 3.).



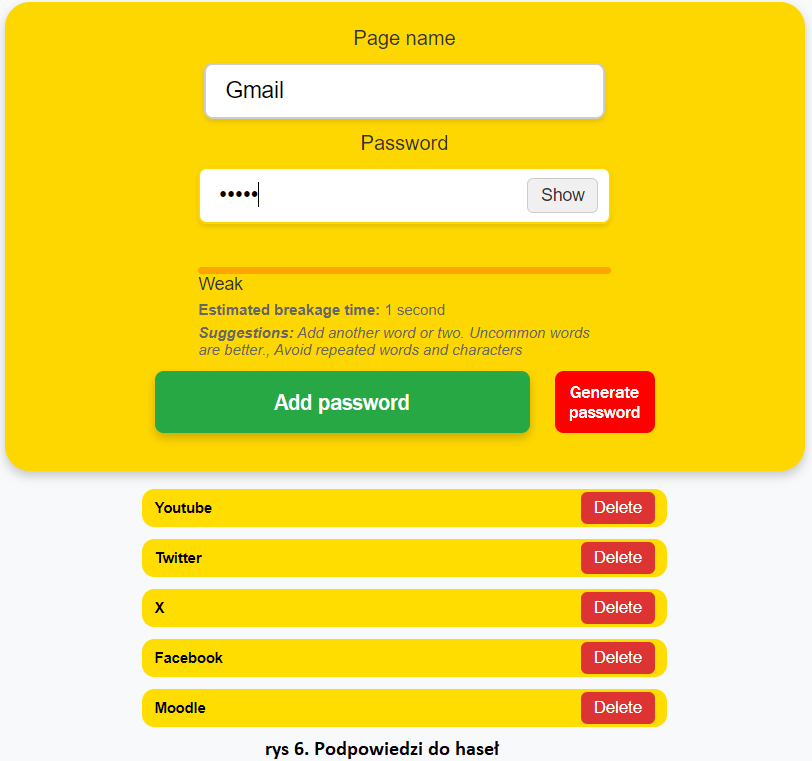
Po wypełnieniu obu pól i wybraniu przycisku “Add password” wyświetli się odpowiedni komunikat (rys 4.).



Natomiast w przypadku wpisania zbyt słabego hasła otrzymamy komunikat, iż podane hasło jest zbyt słabe (rys 5.).



Gdy użytkownik wpisuje hasło pod polem hasła wyświetlają się pasek siły hasła, czas potrzebny na złamanie hasła oraz podpowiedzi jak można poprawić hasło (rys 6.).



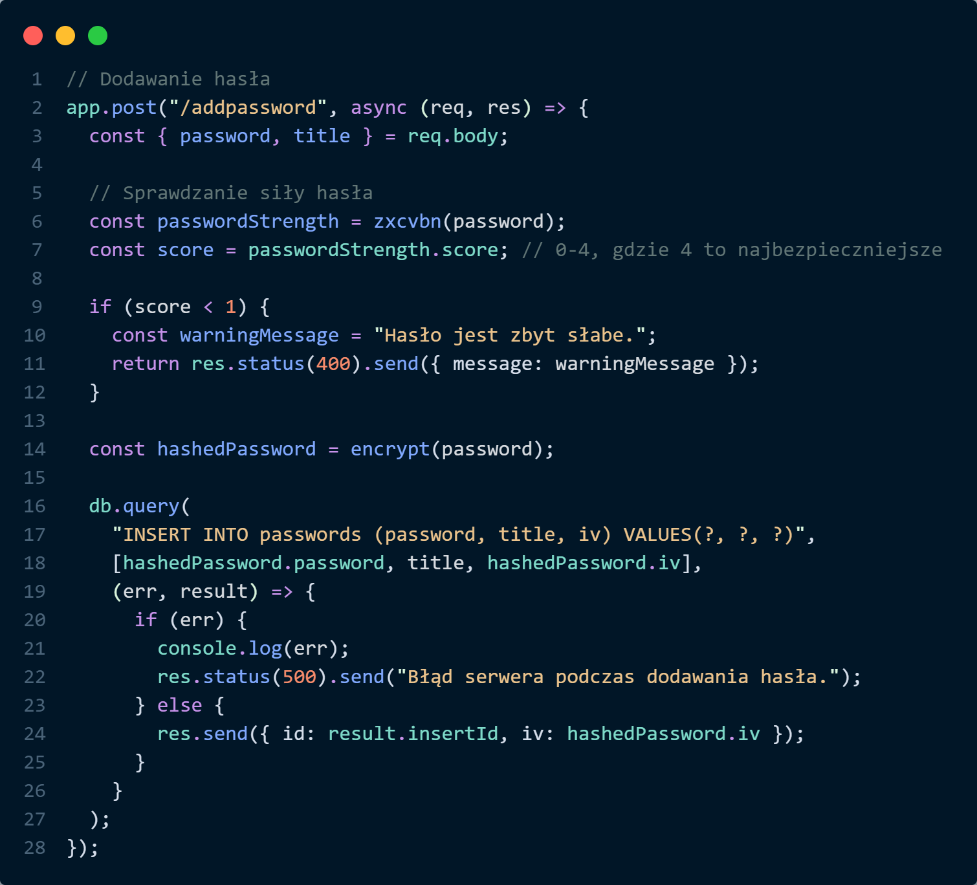
Wpisywane hasło można również podejrzeć wybierając przycisk “Show” lub ukryć przyciskiem “Hide”. Wcześniej zapisane hasła można podejrzeć poprzez naciśnięcie na nie w tabeli po czy można je ponownie ukryć ponownym kliknięciem. Dodane hasła można usunąć z zapisanych.

# Najważniejsze fragmenty kodu

**Strona serwera:**

* **Dodawanie hasła** (POST /addpassword)  
  Endpoint obsługujący dodawanie nowego hasła do bazy danych.

1. **Walidacja siły hasła:** Sprawdza siłę hasła za pomocą biblioteki zxcvbn. Jeżeli siła hasła jest niewystarczająca (score < 1), zwraca odpowiedź z kodem błędu 400 i odpowiednim komunikatem.
2. **Szyfrowanie hasła:** Hasło jest szyfrowane za pomocą funkcji encrypt, co zapewnia jego bezpieczeństwo.
3. **Zapisywanie w bazie danych:** Zaszyfrowane hasło, jego tytuł i wektor inicjalizacyjny (IV) są zapisywane w tabeli passwords. W przypadku sukcesu zwracany jest identyfikator nowego rekordu i IV; w przypadku błędu zgłaszany jest błąd serwera (kod 500).



* **Funkcja szyfrująca hasło (encrypt)**

1. Tworzy unikalny wektor inicjalizacyjny (IV) za pomocą losowych bajtów.
2. Używa algorytmu AES-256-CTR z tajnym kluczem (secret) oraz IV do zaszyfrowania hasła.
3. Zwraca obiekt zawierający:
   * Zaszyfrowane hasło w formacie tekstowym (hex).
   * Wektor inicjalizacyjny (IV), również w formacie hex.

* **Funkcja deszyfrująca hasło (decrypt)**

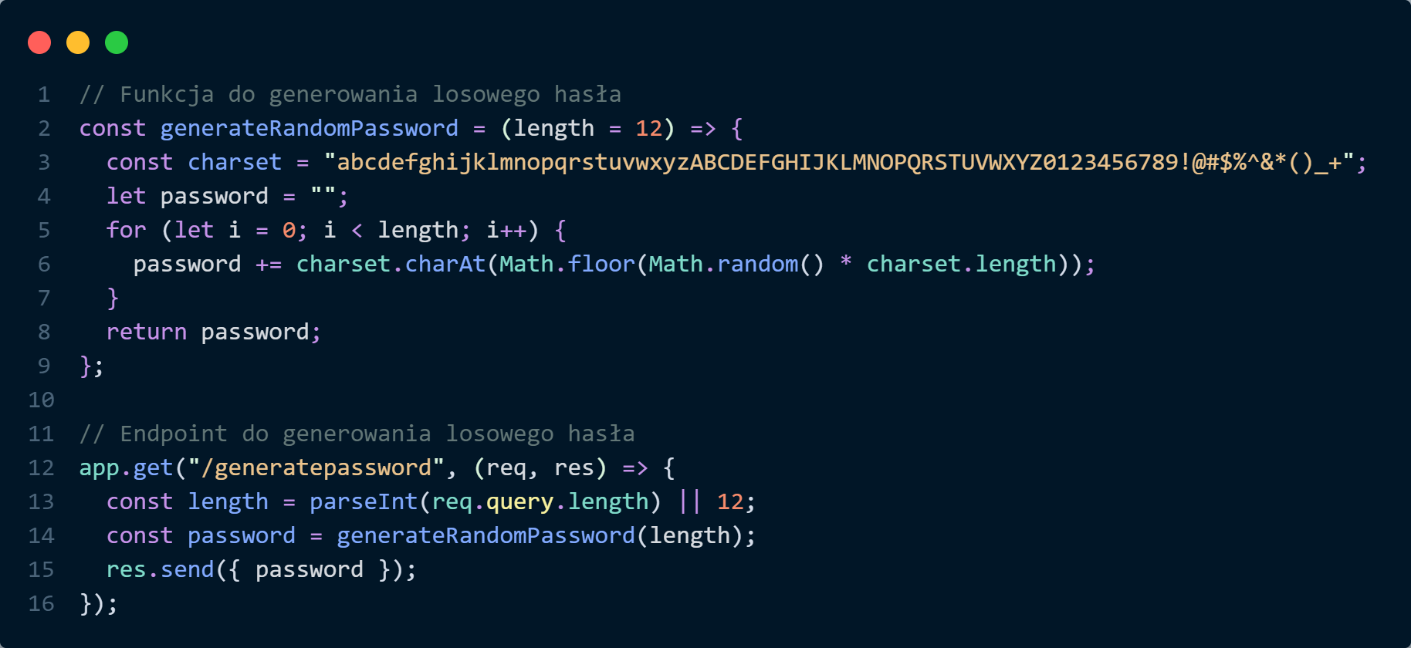
1. Odtwarza algorytm AES-256-CTR, wykorzystując dostarczony tajny klucz (secret) oraz wektor inicjalizacyjny (IV).
2. Rozszyfrowuje zaszyfrowane hasło na podstawie obiektu zawierającego password (zaszyfrowane) i IV.
3. Zwraca odszyfrowane hasło jako zwykły tekst.



**Strona klienta:**

* **Funkcja do generowania losowego hasła (generateRandomPassword)**  
  Funkcja *generateRandomPassword* tworzy losowe hasło o podanej długości (domyślnie 12 znaków). Wykorzystuje zbiór znaków składający się z liter (małych i wielkich), cyfr oraz symboli specjalnych. Hasło jest generowane w pętli poprzez losowe wybieranie znaków z określonego zestawu.
* **Endpoint do generowania losowego hasła (GET /generatepassword)**  
  Endpoint zwraca losowo wygenerowane hasło na żądanie:

1. **Długość hasła:** Odczytuje parametr length z zapytania (domyślnie 12**).**
2. **Generowanie**: Tworzy hasło przy użyciu funkcji *generateRandomPassword*.
3. **Odpowiedź**: Zwraca hasło w formacie JSON.



* **Dodawanie hasła (addPassword)**

Funkcja wysyła nowe hasło i jego nazwę na serwer:

1. Sprawdza, czy pola są wypełnione – w razie braku wyświetla ostrzeżenie (Swal).
2. Wysyła żądanie POST na /addpassword.
   * **Sukces**: Wyświetla komunikat, aktualizuje listę haseł, resetuje pola i odświeża dane.
   * **Błąd**: Wyświetla ostrzeżenie o słabym haśle i loguje błąd.



* **Generowanie losowego hasła** (generatePassword)

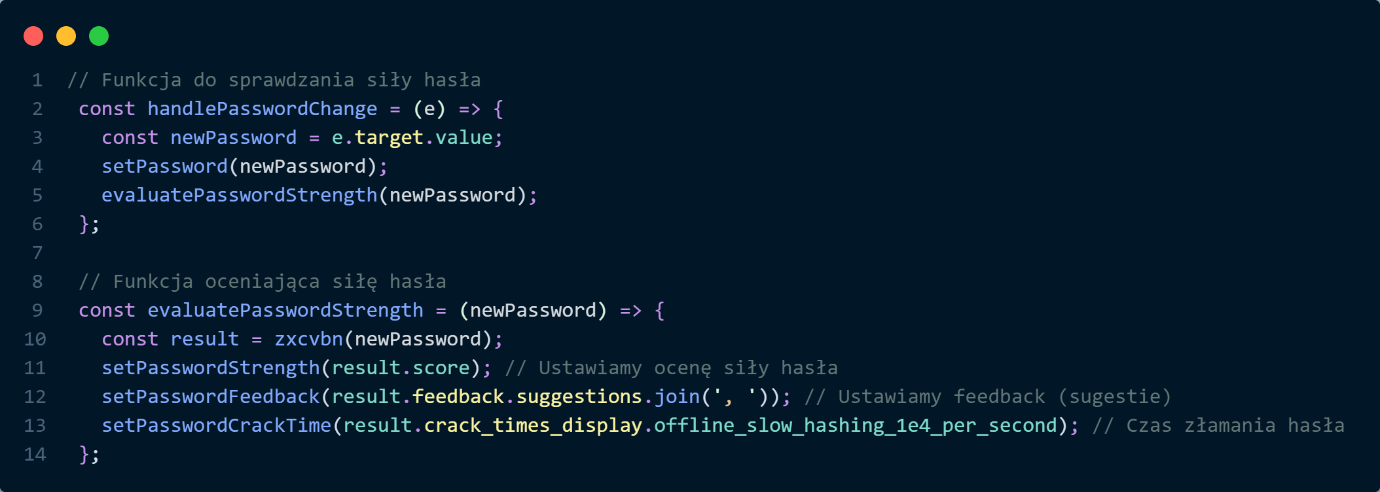
Funkcja wysyła żądanie GET na endpoint /generatepassword, prosząc o hasło o długości 16 znaków:

1. **Sukces**:
   * Ustawia wygenerowane hasło w stanie (setPassword).
   * Wyświetla komunikat z hasłem (Swal).
   * Ocena siły wygenerowanego hasła (evaluatePasswordStrength).
2. **Błąd**: Loguje szczegóły błędu w konsoli.



* **Obsługa i ocena siły hasła**

1. **Zmiana hasła (handlePasswordChange)**
   * Aktualizuje stan hasła podczas wpisywania przez użytkownika.
   * Wywołuje funkcję oceny siły hasła (evaluatePasswordStrength).
2. **Ocena siły hasła (evaluatePasswordStrength)**
   * Korzysta z zxcvbn do analizy hasła.
   * Aktualizuje:
     + Wynik siły hasła (setPasswordStrength).
     + Sugestie poprawy hasła (setPasswordFeedback).
     + Szacowany czas złamania hasła (setPasswordCrackTime).



# Podsumowanie

W projekcie zaimplementowano mechanizmy odpowiadające za umożliwienie bezpiecznego przechowywania haseł użytkownika. Wykorzystano sprawdzone algorytmy szyfrowania oraz biblioteki umożliwiające walidację i sprawdzanie siły dodawanych haseł. Aplikacja pozwala na przechowywanie zaszyfrowanych haseł i ich weryfikację po wcześniejszym odszyfrowaniu hasła. Architektura tego rozwiązania została rozbita na trzy kluczowe elementy:

* Warstwa klienta – odpowiada za mniej wrażliwe operacje dotyczące zarządzania hasłami, obsługę interfejsu użytkownika, dodawanie haseł i wyświetlanie informacji dotyczących ich siły
* Warstwa serwera – jest odpowiedzialna za szyfrowanie i odszyfrowywanie haseł, zapisywanie ich do bazy danych oraz dostarczanie danych warstwie klienta
* Warstwa bazy danych – odpowiada za przechowywanie danych w formie dostarczonej przez warstwę serwera.

Całość została zintegrowana i pozwala na bezpieczne oraz wygodne przechowywanie i zarządzanie hasłami użytkownika.