

Opis Aplikacji: Generator Promptów JSON

Cel i Zakres

Aplikacja "Generator Promptów JSON" to statyczna strona internetowa hostowana na GitHub Pages (z opcją migracji na Netlify), służąca do tworzenia strukturyzowanych plików JSON na podstawie użytkownika inputu. JSON ten jest przeznaczony do użycia w modelach LLM text-to-image (np. DALL-E, Stable Diffusion, Midjourney, GPT-4o, Flux), gdzie struktura pomaga w precyzyjnym definiowaniu parametrów generowania obrazów. Użytkownik zaczyna od podania krótkiego opisu (prompt), wybiera wartości dla predefiniowanych kluczy (np. styl, oświetlenie), może dodać custom klucze z wartościami, a następnie generuje JSON. Aplikacja ma charakter edukacyjny poprzez tooltips wyjaśniające każdy klucz, co czyni ją przydatną dla internautów, w tym studentów na zajęciach z prompt engineeringu lub AI.

Zakres:

- Brak przetwarzania danych użytkownika poza przeglądarką (wszystko client-side).
- Język: angielski.
- Brak integracji z zewnętrznymi API lub serwisami.
- Responsywny design, minimalistyczny (czysty, prosty layout z akcentem na usability).
- Nie jest to pełna PWA, ale rozważ podstawowe cechy offline (np. cache via service worker, jeśli to pasuje – ale nie priorytet).
- Wymagania niefunkcjonalne: Wysoka wydajność (szybkie ładowanie na GitHub Pages), ochrona przed XSS i innymi lukami (np. sanitizacja inputów), dostępność (ARIA labels dla formularzy).

Docelowi użytkownicy: Internauci zainteresowani AI image generation, w tym edukacyjnie (studenci). Zakładamy dorosłą publiczność bez dodatkowych restrykcji.

Funkcje

Aplikacja składa się z jednego głównego widoku (single-page app) z formularzem, sekcją generowania JSON i przyciskami akcji. Oto szczegółowa lista funkcji:

1. Główny Formularz Wprowadzania Danych:

- Pole tekstowe dla "prompt" (wymagane): Użytkownik wpisuje krótki opis obrazu (np.

"a serene mountain landscape"). Tooltip: "Core description of the image, e.g., 'a futuristic city at sunset'. This is the main input for the LLM."

- Pole tekstowe dla "negative_prompt" (opcjonalne): Co unikać na obrazie. Tooltip: "Elements to exclude, e.g., 'blurry, low quality' – common in models like Stable Diffusion to refine outputs."
- Accordion (zwijane sekcje) dla 8 pozostałych predefiniowanych kluczy (aby nie przytłaczać UI; domyślnie zwinięte, z wyjątkiem pierwszych 2-3 dla ułatwienia startu):

- **style** (dropdown select): Default: "realistic". Wartości: ["realistic", "cartoon", "oil painting", "digital art", "photorealistic", "abstract", "pixel art", "watercolor", "sketch"].

- "cyberpunk"]. Tooltip: "Artistic style influences the rendering technique, e.g., 'cyberpunk' for neon-futuristic vibes."
 - **lighting** (dropdown select): Default: "natural". Wartości: ["natural daylight", "dramatic shadows", "soft ambient", "harsh sunlight", "neon glow", "candlelight", "backlit", "volumetric godrays", "studio lighting", "moonlight"]. Tooltip: "Lighting sets the scene's illumination, affecting shadows and highlights for better realism."
 - **mood** (dropdown select): Default: "neutral". Wartości: ["joyful", "mysterious", "serene", "dramatic", "eerie", "whimsical", "adventurous", "melancholic", "energetic", "peaceful"]. Tooltip: "Mood conveys emotional tone, helping models interpret the atmosphere."
 - **color_scheme** (multi-select dropdown): Default: ["neutral"]. Wartości: ["monochrome", "vibrant", "pastel", "earth tones", "cool blues", "warm oranges", "high contrast", "sepia", "neon", "grayscale"]. Tooltip: "Color palette defines dominant hues; multi-select for combinations."
 - **composition** (dropdown select): Default: "balanced". Wartości: ["rule of thirds", "centered", "symmetrical", "asymmetrical", "wide angle", "close-up", "panoramic", "depth of field", "leading lines", "framed"]. Tooltip: "Composition guides element placement for visual balance."
 - **aspect_ratio** (dropdown select): Default: "1:1". Wartości: ["1:1 (square)", "16:9 (widescreen)", "9:16 (portrait)", "4:3", "3:2", "2:1 (panorama)", "custom"]. Tooltip: "Aspect ratio controls image dimensions; 'custom' allows free input if needed."
 - **quality** (dropdown select): Default: "high". Wartości: ["low detail", "medium detail", "high detail", "ultra HD", "8K", "photorealistic", "stylized", "minimalist", "intricate", "sharp focus"]. Tooltip: "Quality level impacts detail and resolution in generated images."
 - **seed** (input liczbowy): Default: losowa liczba (generowana przy otwarciu, np. `Math.random() * 1000000`). Tooltip: "Fixed seed ensures repeatable results in diffusion models."
- Sekcja dla custom kluczy: Przycisk "Add Custom Key" otwiera inputy (nazwa klucza: `CustomKey`, wartość: `CustomValue`)

string, wartości: input tekstowy dla stringa/liczby lub multi-input dla array).

Użytkownik może dodać wiele. Walidacja: Nie pozwala na duplikaty nazw kluczy (predefiniowane lub istniejące custom) – wyświetl komunikat błędu (np. alert lub inline error: "Key already exists, please choose a unique name.").

2. Generowanie JSON:

- Przycisk "Generate": Zbiera dane z formularza, buduje obiekt JSON (np. { "prompt": "...", "style": "realistic", ... , custom_key: "value" }). Wyświetla go w textarea (readonly) poniżej formularza dla podglądu i edycji ręcznej jeśli potrzeba.
- Struktura JSON: Prosty obiekt z kluczami jako properties, wartościami jako stringi/liczby/array (zależnie od typu).

3. Opcje Eksportu:

- Ikona kopiowania (clipboard) w prawym górnym rogu textarea: Kopiuje JSON do schowka (użyj navigator.clipboard.writeText).
- Przycisk "Download JSON": Pobiera plik .json z wygenerowanym obiektem (użyj Blob i URL.createObjectURL).

4. Reset:

- Przycisk "Start Over": Przywraca stan domyślny – czyści pola tekstowe, resetuje dropdowny do defaultów, usuwa wszystkie custom klucze, generuje nowe losowe seed, czyści textarea i historię.

5. Dodatkowe Funkcje (Zatwierdzone):

- **Edukacyjne Tooltips:** Dla każdego pola/klucza – hover/popover z krótkim wyjaśnieniem (jak w przykładach powyżej). Implementacja: Natywne title attribute lub custom JS tooltip library (np. lightweight jak Tippy.js, jeśli Vanilla JS).
- **Walidacja Client-Side:** Wymagany prompt (nie pusty), poprawne typy (np. seed jako liczba), brak duplikatów kluczy. Wyświetl błędy inline lub w alertach.
- **Lokalne Sugestie:** Dla dropdownów – bazuj na predefiniowanych listach (z researchu), bez AI (statyczne).
- **Historia:** Użyj LocalStorage do zapisywania ostatnich 3 wygenerowanych JSON-ów (klucz: "jsonHistory" jako array stringów). Wyświetl w małej sekcji poniżej textarea (np. "Recent JSONs" z przyciskami do załadowania). Automatycznie dodawaj po generacji, bez wysyłania danych nigdzie.

6. UI/UX Szczegóły:

- Minimalistyczny: Białe tło, proste fonty (np. sans-serif), niebieskie akcenty dla przycisków. Responsywny: Mobile-first (użyj media queries lub framework).
- Layout: Top: Nagłówek "JSON Prompt Generator". Środek: Formularz z accordion. Dół: Przyciski Generate/Start Over/Download, textarea, historia.
- Dostępność: ARIA labels, keyboard navigation dla accordion i dropdownów.

Technologie

- **Core:** Vanilla JavaScript (dla interaktywności: event listeners, DOM manipulation),
HTML5, CSS3 (dla responsywności i stylów)

• • • • •, • • • • (dla responsywności i stylów).

- **Frameworki/Biblioteki:**

- CSS: Tailwind CSS (dla szybkiego, minimalistycznego stylingu) lub Bootstrap (dla accordion i responsywności).
- Bezpieczeństwo: DOMPurify (do sanitizacji inputów przed wyświetleniem, zapobieganie XSS).
- Tooltips: Natywne lub lekka biblioteka jak Tippy.js (jeśli potrzeba advanced hover).
- Deployment: GitHub Pages (repo z index.html, assets w folders). CI/CD: GitHub Actions dla auto-build. Opcja: Netlify dla łatwiejszego drag-and-drop i custom domains.
- Brak: Serwer backend, bazy danych – wszystko client-side. Dla LocalStorage: Tylko lokalnie w przeglądarce.

Wskazówki dla Programistów

- **Struktura Kodu:**

- Folders: /src (JS files), /css, /assets (ikony). Główny plik: index.html z <script> tagami.
- JS: Modułowy – np. functions.js dla generowania JSON, ui.js dla event handlers (np. addEventListener na przyciski).
- Accordion: Użyj

1. HTML lub Bootstrap collapse.

/ • Najlepsze Praktyki:

- Bezpieczeństwo: Sanitizuj wszystkie inputy (DOMPurify.sanitize) przed insertem do DOM. Dodaj CSP meta tag: <meta http-equiv="Content-Security-Policy" content="default-src 'self'; script-src 'self';">. Unikaj eval/innerHTML.
- Wydajność: Minimalizuj bundle size (no heavy libs), lazy-load jeśli potrzeba. Testuj na mobile (Chrome DevTools).
- Walidacja: W funkcji generateJSON() sprawdź: if (!prompt) { alert("Prompt is required"); return; }. Dla custom: Sprawdź Object.keys(existing).includes(newKey) – jeśli tak, error.
- Historia: JSON.parse/localStorage.getItem na load, push do array (max 3, shift old), stringify/setItem po generate.
- Edge Cases: Handle puste wartości (użyj defaults), duże inputy (limit textarea size), browser compatibility (ES6+).
- Testy: Manualne (unit tests w console dla functions), lub simple Jest jeśli repo pozwala. Sprawdź XSS: Wpisz <script>alert(1)</script> – powinno być sanitized.
- Deployment: Commit do main, włącz GitHub Pages w settings. Dla Netlify: Link repo i deploy.
- Czas Estymowany: 10–20 godzin dla junior dev (prosty app). Skup się na czystym kodzie, comments w JS.

Plan Prac Programistycznych i Milestone'y

Plan jest podzielony na 4 główne etapy (milestone'y), z estymowanym czasem na podstawie całkowitego zakresu (15 godzin dla jednego developera). Każdy etap zawiera

szczegółowy wykaz zadań. Po każdym milestone następuje faza testów, w której programista przeprowadza manualne i automatyczne testy (jeśli適用ne). Testy skupiają się na funkcjonalności, bezpieczeństwie, wydajności i edge cases. Zakładamy iteracyjny rozwój: po testach, jeśli błędy, wracaj do zadań. Użyj Git dla version control (branch per milestone). Cały plan: 2-3 dni pracy (przy 5-8h/dzień).

Milestone 1: Przygotowanie i Setup (Estymowany czas: 2 godziny)

Wykaz zadań:

- Utwórz repozytorium GitHub i skonfiguruj GitHub Pages (włącz w settings).
- Zainstaluj potrzebne narzędzia lokalnie (np. VS Code, Node.js jeśli dla minifikacji).
- Utwórz strukturę plików: index.html, styles.css, script.js, /assets (dla ikon clipboard).
- Dodaj podstawowy HTML skeleton (nagłówek, sekcja formularza, footer).
- Zintegruj biblioteki: DOMPurify (via CDN), Tailwind CSS lub Bootstrap (CDN dla prostoty).
- Dodaj CSP meta tag do .

Plan Testów po Milestone 1:

- Manualne: Otwórz stronę lokalnie – sprawdź czy ładuje się bez błędów konsoli, czy CSS działa (podstawowy layout widoczny).
 - Bezpieczeństwo: Wpisz testowy input z HTML (**test**) w dowolne pole – upewnij się, że nie renderuje jako bold (sanitizacja).
 - Wydajność: Zmierz czas ładowania (DevTools <1s).
 - Edge: Sprawdź na mobile emulatorze (responsywność podstawowa).
 - Automatyczne: Brak (za wcześnie); jeśli Jest, testuj importy. Czas testów: 30 min.
- Kryterium sukcesu: Strona otwiera się bez błędów, setup gotowy.

Milestone 2: Implementacja UI i Formularza (Estymowany czas: 5 godzin)

Wykaz zadań:

- Zbuduj formularz: Pole prompt i negative_prompt (textarea/input).

- Implementuj accordion dla predeterminowanych kluczy (użyj Bootstrap collapse lub native)
- Dodaj dropdownny/select dla kluczy (z defaultami i listami wartości).
- Sekcja custom kluczy: Przycisk "Add Custom Key" – dynamicznie dodawaj inputy via JS (createElement).
- Dodaj tooltips (title attribute lub Tippy.js).
- Stylizuj: Minimalistyczny design, responsywny (media queries dla mobile).
- Dodaj przyciski: Generate, Start Over, Download (hidden na start).
- Wyświetl textarea dla JSON (readonly).

Plan Testów po Milestone 2:

- Manualne: Wypełnij formularz – sprawdź czy accordion zwija/rozwijaja, dropdowny wybierają wartości, custom dodaje pola bez duplikatów (wyświetl error jeśli duplikat).
- Funkcjonalność: Hover na tooltips – widać wyjaśnienia.
- Bezpieczeństwo: Wpisz XSS (<script>alert(1)</script>) w custom key – nie wykonuje się, sanitizowane.
- Responsywność: Test na różnych rozdzielcościach (desktop/mobile) – elementy nie overflow.
- Edge: Dodaj 10 custom kluczy – UI nie psuje się; pusty custom – brak błędu.
- Automatyczne: Jeśli Jest, testuj DOM elements (np.
`expect(document.querySelector('#prompt')).toBeDefined();`). Czas testów: 1 godzina.
Kryterium sukcesu: Formularz wizualnie kompletny, interaktywny bez crashy.

Milestone 3: Implementacja Logiki JS i Funkcji (Estymowany czas: 6 godzin)

Wykaz zadań:

- Event listeners: Na "Generate" – zbierz dane, waliduj (wymagany prompt, no duplikaty), zbuduj JSON obiekt (JSON.stringify). Wyświetl w textarea.

- Na "Start Over" – resetuj pola, generuj nowe seed, czyść LocalStorage historię.
- Kopiowanie: Ikona clipboard – navigator.clipboard.writeText z textarea.
- Download: Utwórz Blob z JSON, URL.createObjectURL, symuluj click na .
- Historia: LocalStorage – zapisz po generate (max 3), wyświetl listę z przyciskami load (wypełnij formularz z zapisanego JSON).
- Walidacja: Inline errors dla błędów (np. div z klasą error).
- Seed: Generuj losowo na load/reset.

Plan Testów po Milestone 3:

- Manualne: Wypełnij, generate – JSON poprawny w textarea; kopuj/download działa (sprawdź plik). Reset czyści wszystko.
- Funkcjonalność: Dodaj custom, generate – klucz w JSON; historia zapisuje/ładuje ostatnie 3.
- Bezpieczeństwo: Test XSS w prompt/custom – JSON escaped, nie wykonuje kodu.
- Wydajność: Generate z wieloma custom – <100ms.
- Edge: Brak prompt – error; duplikat key – error; load z historii – pola wypełniają się poprawnie.
- Automatyczne: Jest tests dla funkcji (np. generateJSON() returns expected object; reset clears DOM). Czas testów: 1 godzina. Kryterium sukcesu: Pełna funkcjonalność client-side działa.

Milestone 4: Finalne Testy, Optymalizacja i Deployment (Estymowany czas: 2 godziny)

Wykaz zadań:

- Optymalizuj: Minifikuj CSS/JS (jeśli tool), dodaj ARIA labels dla dostępności.

- Pełne testy cross-browser (Chrome, Firefox, Safari).
- Deploy: Push do GitHub, sprawdź GitHub Pages URL.
- Konfiguruj CI/CD (GitHub Actions: build na push).
- Dodaj README z instrukcjami.

Plan Testów po Milestone 4:

- Manualne: Pełny user flow – od otwarcia do generate/download; test na live GitHub Pages.
- Bezpieczeństwo: Skany narzędzi (np. browser devtools security tab) – brak vulnerabilities.
- Wydajność: Lighthouse score >90 (accessibility, performance).
- Edge: Offline (LocalStorage persists); multi-device.
- Automatyczne: Full suite Jest jeśli zaimplementowane; inaczej manualne coverage wszystkich funkcji. Czas testów: 1 godzina. Kryterium sukcesu: Aplikacja live, bez błędów – gotowa do użycia.