Projekt Informatyczny: Zostań Twórcą Aplikacji z AI!

Przedmiot wiodący: Informatyka

Klasa: 8, Szkoła Podstawowa

Czas realizacji: 6-8 godzin lekcyjnych

Forma pracy: Zespoły 3-osobowe

Spis treści

Wprowadzenie dla Nauczyciela

Projekt "Kulinarny Kreator AI" to interdyscyplinarne przedsięwzięcie, które odzwierciedla nowoczesne podejście do nauczania informatyki, oparte na metodzie projektowej (PBL - Project-Based Learning). Jego celem jest przeprowadzenie uczniów przez cały, realistyczny proces tworzenia aplikacji internetowej – od **pierwotnego pomysłu**, przez **fazę projektowania**, **programowanie**, aż po **finalną prezentację i refleksję**. Uczniowie, pracując w małych, trzyosobowych zespołach, stworzą w pełni funkcjonalny generator przepisów kulinarnych, wykorzystując do tego ogólnodostępną i potężną sztuczną inteligencję **Google Gemini**.

Projekt ten wykracza poza tradycyjne nauczanie składni języków programowania. Kładzie nacisk na **praktyczne umiejętności rozwiązywania problemów**, **myślenie algorytmiczne** oraz **głębokie zrozumienie, jak działają i do czego mogą służyć współczesne technologie AI**. Uczniowie nie tylko nauczą się pisać kod, ale także planować, współpracować i kreatywnie wykorzystywać narzędzia, które kształtują naszą cyfrową rzeczywistość. To doskonała okazja do rozwinięcia kompetencji przyszłości w angażujący i motywujący sposób.

Cele Projektu i Powiązanie z Podstawą Programową

Główne cele (Informatyka):

- **Projektowanie i tworzenie stron internetowych:** Uczniowie zbudują semantyczną strukturę strony za pomocą **HTML**, definiując jej logiczne części, a następnie nadadzą jej profesjonalny wygląd, korzystając z podstaw nowoczesnego frameworka **CSS**, jakim jest Tailwind. (Podstawa Programowa: V.2)
- **Programowanie i rozwiązywanie problemów:** Uczniowie "ożywią" swoją stronę za pomocą języka **JavaScript**. Nauczą się manipulować elementami strony (DOM), tworzyć interaktywne elementy (obsługa przycisków, formularzy) i zaimplementują kluczową logikę aplikacji, w tym dynamiczne generowanie treści na podstawie danych otrzymanych z zewnętrznego źródła. (PP: I.3, I.4)
- **Zrozumienie i wykorzystanie AI:** Uczniowie dowiedzą się, czym jest **API** (Interfejs Programowania Aplikacji) i w praktyce wykorzystają je do komunikacji z modelem językowym Google Gemini. Zrozumieją rolę AI jako "mózgu" aplikacji i nauczą się podstaw **inżynierii promptów** (prompt engineering), czyli sztuki precycycynego formułowania zapytań, aby uzyskać jak najlepsze rezultaty. (PP: III.2)
- **Praca zespołowa i zarządzanie projektem:** Uczniowie nauczą się efektywnie dzielić zadaniami, komunikować postępy, wspólnie rozwiązywać napotkane problemy i planować pracę w zespole, korzystając z prostych metodyk zwinnych. (PP: II.4)

Rozwijane Kompetencje Kluczowe:

- **Myślenie krytyczne:** Uczniowie będą musieli analizować odpowiedzi generowane przez AI, oceniać ich sensowność i logikę. Nauczą się, że AI jest narzędziem, które wymaga weryfikacji, a nie bezbłędną wyrocznią.
- **Umiejętność komunikacji:** Praca w zespole wymusi na uczniach jasne formułowanie myśli, argumentowanie swoich decyzji projektowych oraz skuteczne prezentowanie efektów swojej pracy.
- **Rozwiązywanie problemów (Testowanie i Debugowanie):** Faza testowania i naprawiania błędów ("debugowania") będzie kluczowym elementem projektu, uczącym systematycznego podejścia do znajdowania i eliminowania problemów w kodzie.

Cele interdyscyplinarne:

- **Edukacja dla zdrowia / Biologia:** Projekt bezpośrednio wiąże się z tematyką **zdrowego odżywiania**. Uczniowie, implementując funkcje dotyczące wartości odżywczych i alergenów, zgłębią wiedzę na temat kaloryczności, białek, tłuszczów, węglowodanów oraz wpływu różnych składników na organizm. To doskonały pretekst do rozmowy o zbilansowanej diecie i świadomych wyborach żywieniowych. Wskazówka dla nauczyciela: Faza 1 projektu to idealny moment na zaproszenie na fragment lekcji nauczyciela biologii, który może przeprowadzić krótką, 15-minutową dyskusję na temat kluczowych składników odżywczych i najczęstszych alergenów. Dzięki temu uczniowie będą mieli solidne podstawy merytoryczne do projektowania funkcji swojej aplikacji.
    - **Powiązanie z PP Biologii (Klasa 7-8):** VI. Funkcjonowanie organizmu człowieka. Uczeń: 5) przedstawia zasady prawidłowego odżywiania się i wyjaśnia rolę składników pokarmowych w organizmie; 8) przedstawia negatywny wpływ niektórych substancji na organizm człowieka (w kontekście składników niejadalnych).
- **Język polski:** Uczniowie będą ćwiczyć umiejętność **precyzyjnego i jednoznacznego formułowania poleceń** (tzw. "promptów") dla sztucznej inteligencji. Odkryją, jak zmiana jednego słowa może wpłynąć na wynik, co jest doskonałym ćwiczeniem w zakresie logiki i jasności wypowiedzi.
    - **Powiązanie z PP Języka Polskiego (Klasa 7-8):** II. Kształcenie językowe. Uczeń: 1.4) posługuje się w wypowiedziach pisemnych poprawnymi formami gramatycznymi; 2.1) tworzy spójne wypowiedzi w następujących formach gatunkowych: instrukcja.
- **Technika / Plastyka:** Faza projektowania interfejsu (UI/UX) pozwoli uczniom na **kreatywne zaplanowanie wyglądu i użyteczności** ich aplikacji. Będą mogli zaprojektować prosty logotyp, dobrać spójną paletę kolorów i zastanowić się, jak sprawić, by ich narzędzie było przyjazne dla użytkownika.
    - **Powiązanie z PP Plastyki (Klasa 4-7):** I. Opanowanie zagadnień z zakresu języka plastyki i terminologii plastycznej. Uczeń: 1) wymienia dziedziny sztuk plastycznych; 2) rozróżnia cechy i rodzaje kompozycji.
- **Etyka i Bezpieczeństwo:** Projekt poruszy dwa kluczowe aspekty bezpieczeństwa cyfrowego:

- **Bezpieczeństwo danych:** Uczniowie dowiedzą się, dlaczego klucze API są poufne i jak się je chroni w profesjonalnych aplikacjach.
- **Odpowiedzialność za treść:** Implementując mechanizm sprawdzający niebezpieczne składniki, uczniowie zrozumieją etyczną odpowiedzialność twórcy za bezpieczeństwo użytkownika.

Struktura Projektu – Krok po Kroku

Wymagania wstępne: Zaleca się, aby projekt był realizowany po przeprowadzeniu lekcji wprowadzających do absolutnych podstaw języka HTML (czym są znaczniki) oraz CSS (czym są style i klasy).

Faza 1: Koncepcja i Projekt (2 godziny lekcyjne)

1. **Wprowadzenie (Kick-off):** Nauczyciel prezentuje działającą aplikację "Kreator Przepisów" jako inspirację i namacalny cel projektu. Omawia główne założenia i harmonogram.
2. **Formowanie Zespołów:** Podział klasy na 3-osobowe grupy, z zachowaniem równowagi kompetencji (jeśli to możliwe).
3. **Warsztat Koncepcyjny (Burza Mózgów):** Każdy zespół definiuje swój projekt, odpowiadając na pytania:
   - *Jaki jest główny cel naszej aplikacji i kto jest jej użytkownikiem?*
   - *Jakie funkcje są absolutnie niezbędne (Minimum Viable Product)?*
   - *Jakie dodatkowe funkcje "premium" moglibyśmy dodać, jeśli starczy nam czasu?*
   - **Pytanie o bezpieczeństwo:** *Jakie potencjalne zagrożenia musimy wziąć pod uwagę? (np. ktoś wpisze niejadalny składnik, jak "szkło" lub "muchomor").*
4. **Projektowanie Interfejsu (Szkicowanie makiety):** Zespoły na kartkach papieru szkicują szczegółowy wygląd swojej aplikacji. To fundament wizualny całego projektu. Nauczyciel wyjaśnia, że jest to tzw. "wireframe" – prosty schemat rozmieszczenia elementów, bez kolorów i szczegółów graficznych.
5. **Podział Ról i Odpowiedzialności w Zespole (sugerowany):**
   - **Lider Projektu:** Odpowiada za spójność projektu, pilnuje harmonogramu i zadań.
   - **Projektant Wyglądu Aplikacji / Kreatywny Dyrektor:** Ta osoba skupia się na wyglądzie i estetyce aplikacji. Jej zadaniem jest przełożenie szkicu na konkretne decyzje: jaką wybrać paletę kolorów, jakie czcionki będą czytelne, jak rozmieścić przyciski, aby korzystanie z aplikacji było intuicyjne. Odpowiada też za wizualną stronę prezentacji końcowej.
   - **Programista / Inżynier Oprogramowania:** Koncentruje się na logice działania kodu i integracji z API.

Faza 2: Budowa Szkieletu – HTML i CSS (2 godziny lekcyjne)

1. **Wprowadzenie do HTML:** Nauczyciel w 15 minut tłumaczy podstawowe, semantyczne znaczniki: `<h1>`, `<p>`, `<textarea>`, `<button>`, `<div>`, `<header>`, `<main>`.
2. **Tworzenie Struktury:** Zespoły, bazując na swoich szkicach, tworzą plik `index.html`.
3. **Wprowadzenie do Stylów:** Nauczyciel wyjaśnia, jak dodać do projektu bibliotekę **Tailwind CSS** i pokazuje, jak za pomocą prostych klas można szybko nadać wygląd elementom.
4. **Stylowanie Aplikacji:** Zespoły "ubierają" swoją aplikację, nadając jej kolory, czcionki i odpowiednie odstępy.

Faza 3: Ożywienie Aplikacji – JavaScript i AI (3 godziny lekcyjne)

1. **Wprowadzenie do JavaScript:** Nauczyciel tłumaczy kluczowe pojęcia: zmienne, funkcje i nasłuchiwanie na zdarzenia (`addEventListener`).
2. **Czym jest API i dlaczego klucz jest ważny?** Nauczyciel używa analogii (np. karta do hotelu), aby wyjaśnić, czym jest klucz API. Następnie prowadzi krótką dyskusję: *Co by się stało, gdyby ktoś ukradł wasz klucz? Dlaczego nie powinniśmy go publikować?*
3. **Implementacja Logiki (programowanie w parach):**
    - Nauczyciel udostępnia gotowy szablon funkcji do komunikacji z API.
    - Każdy zespół generuje swój darmowy **klucz API** w Google AI Studio.
    - Zespoły piszą kod, który po kliknięciu przycisku realizuje następujący algorytm:
        1. Pobierz tekst z pola `<textarea>`.
        2. **(Nowy krok!)** Wywołaj funkcję sprawdzającą bezpieczeństwo składników. Jeśli odpowiedź jest negatywna, wyświetl ostrzeżenie i przerwij działanie.
        3. Zbuduj precycyjne polecenie ("prompt") dla AI.
        4. Wywołaj funkcję wysyłającą zapytanie do Gemini.
        5. Odbierz odpowiedź i dynamicznie stwórz elementy HTML.
        6. Wstaw stworzone elementy na stronę.
4. **Testowanie i Debugowanie:** To najważniejszy etap. Uczniowie celowo testują aplikację, wpisując nietypowe dane (np. "kamienie", "brudna woda"), aby sprawdzić, czy ich zabezpieczenia działają. Uczą się analizować błędy w konsoli i wspólnie szukać rozwiązań. Wskazówka dla nauczyciela: Zorganizuj sesję "Polowania na Błędy" (Bug Hunt). Każdy zespół przekazuje swoją aplikację sąsiedniej grupie na 10 minut. Zadaniem jest znalezienie jak największej liczby błędów lub nietypowych zachowań. To nie tylko uczy testowania, ale także daje cenną informację zwrotną i promuje zdrową rywalizację.

Faza 4: Prezentacja i Podsumowanie (1 godzina lekcyjna)

1. **Demo Day:** Każdy zespół ma 5 minut na zaprezentowanie swojej działającej aplikacji.
2. **Dyskusja i Refleksja:**
    - *Co było najtrudniejszym, a co najciekawszym elementem projektu?*

- *Jak poradziliście sobie z podziałem zadań w zespole?*
- *Dlaczego kwestie bezpieczeństwa, takie jak sprawdzanie składników i ochrona klucza API, są tak ważne przy tworzeniu aplikacji?*
- *Jakie inne problemy moglibyśmy rozwiązać za pomocą AI?*

Kryteria Oceny

- **Funkcjonalność Aplikacji (40%):** Czy aplikacja poprawnie generuje przepisy i jej kluczowe funkcje działają, w tym mechanizm bezpieczeństwa?
- **Jakość Kodu i Projektu (30%):** Czy kod jest czytelny i dobrze zorganizowany? Czy interfejs jest przemyślany i estetyczny?
- **Praca w Zespole (20%):** Jak przebiegała współpraca, komunikacja i podział zadań? Czy zespół potrafił wspólnie i konstruktywnie rozwiązywać problemy, wspierając się nawzajem? Ocenie podlegać będzie zaangażowanie każdego członka zespołu, umiejętność słuchania innych oraz wspólne dochodzenie do rozwiązań, a nie tylko finalny efekt.
- **Prezentacja Końcowa (10%):** Sposób zaprezentowania produktu i omówienia procesu pracy, w tym wniosków dotyczących bezpieczeństwa i etyki.

Projekt "Kulinarny Kreator AI" – Materiały dla Uczniów

Wasza Misja: Stworzyć Inteligentną Aplikację!

Cześć! Przed Wami niezwykła przygoda. Zaczyna się ona od znanego pytania: otwierasz lodówkę, a tam... kilka przypadkowych produktów. Co z tego zrobić? Waszym zadaniem będzie stworzenie aplikacji, która odpowie na to pytanie!

W ciągu najbliższych lekcji zostaniecie prawdziwymi twórcami technologii. Zbudujecie od zera w pełni działającego **'Kreatora Przepisów'** – inteligentnego asystenta, który potrafi wymyślać pyszne dania na podstawie dowolnej listy składników. To nie jest zwykłe zadanie – to Wasz pierwszy krok w świat profesjonalnego tworzenia narzędzi, z których sami chcielibyście korzystać!

Co będziemy robić? Przejdziemy przez cały proces, tak jak robią to programiści w firmach takich jak Google czy YouTube:

1. **Wymyślimy i zaprojektujemy** naszą aplikację. Zastanowimy się, jak ma wyglądać i działać, aby była przyjazna i użyteczna.
2. **Zbudujemy jej wygląd** za pomocą języków **HTML** (który jest jak szkielet budynku) i **CSS** (który jest jak farby i meble).
3. **Nauczymy ją myśleć**, podłączając do niej "mózg" w postaci sztucznej inteligencji **Google Gemini**. To ona będzie naszym kreatywnym szefem kuchni.
4. **Przetestujemy ją** i wcielimy się w rolę detektywów szukających błędów, aby nasza aplikacja działała niezawodnie.
5. Na koniec **zaprezentujemy światu** nasze dzieło! Każdy zespół pokaże swoją aplikację i opowie o swojej pracy.

Będziecie pracować w zespołach, dzieląc się zadaniami i pomysłami. To nie tylko lekcja programowania, ale także kreatywności, współpracy i rozwiązywania problemów. Zdobędziecie prawdziwą "supermoc" – umiejętność tworzenia z kodu czegoś, co działa, pomaga i robi wrażenie. Przygotujcie się na wyzwania, ale przede wszystkim na świetną zabawę i ogromną satysfakcję!

Karta Pracy 1: Faza Koncepcji i Projektu

Cel: Dokładne zaplanowanie aplikacji, zdefiniowanie jej kluczowych funkcji, przemyślenie kwestii bezpieczeństwa i stworzenie pierwszego, szczegółowego szkicu. Dobry plan to połowa sukcesu!

Zadania dla zespołu:

1. **Nazwa i Cel Aplikacji:**
    - Jak nazwiecie Wasz zespół?

- Jaki jest główny cel Waszej aplikacji? Komu ma pomagać? (np. *pomagać rodzicom szybko wymyślić obiad*, *uczyć nastolatków gotowania z tego, co jest w lodówce*).
- **Wskazówka:** Pomyślcie o chwytliwej nazwie dla samej aplikacji, która kojarzy się z jedzeniem, technologią lub kreatywnością (np. "Smak AI", "Lodówkowy Geniusz", "Kreator Smaku").

2. **Lista Funkcji (Burza Mózgów):**
   - **Funkcje niezbędne (Minimum):** Wypiszcie **3-4 funkcje, które są absolutnie niezbędne**, aby aplikacja w ogóle działała. To jest Wasz fundament.
     - *Przykład:* Pole tekstowe na składniki, przycisk "Generuj", miejsce na wyświetlenie przepisu.
   - **Funkcje dodatkowe ("Fajne dodatki"):** Wypiszcie **2-3 dodatkowe funkcje**, które chcielibyście dodać, jeśli starczy Wam czasu. To miejsce na Waszą kreatywność!
     - *Przykład:* Przycisk drukowania, losowy "przepis dnia", informacja o kaloriach, sugestie napojów.
3. **Bezpieczeństwo i Odpowiedzialność:**
   - Pomyślcie, co mogłoby pójść nie tak. Co się stanie, jeśli ktoś wpisze do listy składników coś niejadalnego lub trującego, np. "kamienie", "plastik", "muchomor sromotnikowy"?
   - Jak Wasza aplikacja powinna na to zareagować? Czy powinna mimo wszystko próbować stworzyć przepis? Jak możecie ochronić użytkownika? Zanotujcie pomysły na mechanizm zabezpieczający.
4. **Szkic Aplikacji (Makieta):**
   - Na dużej kartce papieru narysujcie prosty, ale czytelny schemat Waszej aplikacji. Wyobraźcie sobie, że to ekran telefonu lub komputera.
   - Użyjcie prostych kształtów (prostokąty, okręgi) i podpiszcie każdy element: "Tytuł", "Pole na składniki", "Przycisk", "Karta z przepisem".
   - Zastanówcie się nad przepływem informacji: co się dzieje po kliknięciu przycisku? Gdzie pojawia się animacja ładowania? Gdzie wyświetlają się wyniki? To Wasza mapa, która poprowadzi Was przez kolejne etapy.
5. **Podział Ról:**
   - Ustalcie, kto w zespole będzie pełnił jaką rolę. Pamiętajcie, że i tak pracujecie razem i pomagacie sobie nawzajem!
     - **Lider Projektu:** Dba o to, by wszyscy wiedzieli, co mają robić, i pilnuje, czy trzymacie się planu.
     - **Projektant Wyglądu:** Odpowiada za to, żeby aplikacja była ładna i łatwa w obsłudze.
     - **Programista:** Skupia się na napisaniu kodu, który sprawi, że wszystko będzie działać.

Karta Pracy 2: Faza Budowy (HTML i CSS)

Cel: Stworzenie wizualnej struktury aplikacji – jej "szkieletu" (HTML) i "ubrania" (CSS). Na tym etapie aplikacja jeszcze nie będzie działać, ale zacznie wyglądać tak, jak ją zaprojektowaliście.

Zadania dla zespołu:

1. **Struktura HTML (plik `index.html`):**
   - Stwórzcie plik `index.html`.
   - Używając znaczników HTML, zbudujcie "szkielet" strony zgodnie z Waszym szkicem. Pamiętajcie o logicznym podziale (nagłówek, główna treść).
   - **Checklista znaczników do użycia:**
     - `<header>`: Na tytuł i opis aplikacji.
     - `<main>`: Na główną część – formularz i miejsce na wyniki.
     - `<h1>`, `<h2>`, `<p>`: Do tekstów.
     - `<textarea>`: Dla pola do wpisywania składników.
     - `<button>`: Dla przycisku "Generuj Przepisy".
     - `<div>`: Do grupowania i organizowania elementów (np. osobny `div` na całą sekcję z wynikami, osobny `div` na każdą kartę przepisu).
2. **Wygląd CSS (Style):**
   - Dodajcie do pliku jedną linijkę kodu, która załaduje bibliotekę Tailwind CSS (nauczyciel pokaże, jak to zrobić).
   - Używając klas Tailwind, nadajcie wygląd Waszej aplikacji. Eksperymentujcie!
   - **Podpowiedzi – klasy, które warto wypróbować:**
     - **Układ:** `flex`, `grid`, `justify-center`, `items-center` (do centrowania).
     - **Tło:** `bg-gray-100`, `bg-blue-50`, `bg-white`.
     - **Odstępy (wewnętrzne):** `p-4` (mały), `p-8` (duży).
     - **Odstępy (zewnętrzne):** `m-4`, `mt-8` (margines górny), `space-y-4` (odstęp między elementami w pionie).
     - **Tekst:** `text-3xl`, `font-bold`, `text-center`, `text-gray-600`.
     - **Przyciski i Pola:** `rounded-xl` (zaokrąglone rogi), `shadow-lg` (cień), `border`, `focus:ring-2` (efekt po kliknięciu).
     - **Efekty po najechaniu myszką:** `hover:bg-blue-700`, `hover:scale-105`, `transition-all`.

Karta Pracy 3: Faza Ożywienia (JavaScript i AI)

Cel: Zaprogramowanie "mózgu" aplikacji. Na tym etapie sprawimy, że strona stanie się interaktywna, inteligentna i zacznie naprawdę działać!

Zadania dla zespołu:

1. **Podstawy JavaScript:**

- W pliku `index.html`, na samym dole (przed `</body>`), dodajcie znacznik `<script>`. Cały Wasz kod JavaScript będzie się znajdował wewnątrz niego.
- Stwórzcie zmienne, które będą "trzymać" Wasze elementy HTML (np. przycisk, pole tekstowe, kontener na wyniki). Użyjcie `document.getElementById('ID_ELEMENTU')`. Zmienna to jak pudełko z etykietą, w którym trzymamy jedną informację.

2. **Logika Aplikacji (Algorytm):**
   - Napiszcie funkcję, która uruchomi się po kliknięciu przycisku "Generuj". Funkcja to jak przepis na ciasto – zestaw instrukcji, które wykonują się po kolei. Waszym celem jest zrealizowanie poniższego planu:
     1. **Pobierz** tekst z pola `<textarea>` i zapisz go w zmiennej.
     2. **Sprawdź**, czy pole nie jest puste. Jeśli jest, wyświetl komunikat dla użytkownika.
     3. **Wyświetl** animację ładowania, aby użytkownik wiedział, że coś się dzieje.
     4. **(Bezpieczeństwo!)** Stwórz **prompt** (polecenie) do AI, który zapyta, czy składniki są bezpieczne (np. *"Czy na liście 'cebula, szkło, pomidor' jest coś niejadalnego? Odpowiedz TAK lub NIE"*).
     5. Wyślij zapytanie. Jeśli AI odpowie "TAK", pokaż błąd i zakończ działanie funkcji (`return`).
     6. Stwórz **główny, szczegółowy prompt**, który poprosi AI o wygenerowanie przepisów w określonym formacie (JSON). Im dokładniejsze polecenie, tym lepszy wynik!
     7. Wyślij zapytanie i **odbierz dane**. Zapisz je w zmiennej.
     8. Napisz pętlę (`forEach`), która przejdzie przez wszystkie otrzymane przepisy.
     9. Wewnątrz pętli, dla każdego przepisu, **stwórz nowy element `<div>`** (`document.createElement('div')`) i wypełnij go danymi (nazwa, składniki, instrukcje) za pomocą `innerHTML`.
     10. **Dodaj** (`appendChild`) stworzony `<div>` do głównego kontenera na wyniki.
     11. Po zakończeniu pętli **ukryj** animację ładowania.
3. **Podpowiedzi – Co zrobić, gdy...**
   - **...przycisk nie działa?**
     - Sprawdź, czy na pewno dodałeś `addEventListener` do właściwej zmiennej z przyciskiem.
     - Użyj `console.log("Kliknięto!");` na samym początku funkcji, aby zobaczyć w konsoli przeglądarki (F12), czy kliknięcie jest rejestrowane.
   - **...przepisy się nie pojawiają?**
     - Użyj `console.log(zmienna_z_danymi)` zaraz po odebraniu danych z AI. Zobacz w konsoli, czy w ogóle coś przyszło i jaką ma strukturę.

        - Sprawdź, czy poprawnie odwołujesz się do danych (np. `recipe.recipeName`, `recipe.ingredients`). Wielkość liter ma znaczenie!
    - **...pojawia się błąd związany z API?**
        - Upewnij się, że poprawnie wkleiłeś swój klucz API.
        - Sprawdź, czy Twój prompt jest napisany jasno i zrozumiale dla AI. Czasem wystarczy przeformułować zdanie.
    - **Super-wskazówka:** `console.log()` to Twój najlepszy przyjaciel! Używaj go na każdym kroku, aby sprawdzać, co znajduje się w Twoich zmiennych i w którym miejscu kod przestaje działać.

Prompt startowy dla AI -

Twoim zadaniem jest wcielenie się w rolę eksperta ds. tworzenia aplikacji internetowych. Celem jest stworzenie w pełni funkcjonalnej, samodzielnej strony internetowej o nazwie "Kreator Przepisów". Aplikacja ma służyć jako inteligentny asystent, który generuje propozycje kulinarne na podstawie listy składników dostarczonej przez użytkownika.

Zaprojektuj interfejs zgodnie z poniższymi wytycznymi, kładąc nacisk na prostotę, czytelność i intuicyjną obsługę.

- **Sekcja Główna:**
  - **Tytuł:** Wyraźny nagłówek "Kreator Przepisów".
  - **Opis:** Krótka instrukcja dla użytkownika, np. "Wprowadź posiadane składniki, aby otrzymać propozycje dań."
  - **Pole Wprowadzania Danych:** Pojedyncze, duże pole tekstowe przeznaczone do wpisania listy składników. Jako przykład dla użytkownika, pole może zawierać tekst pomocniczy: "np. pomidory, makaron, cebula, oliwa z oliwek".
  - **Przycisk Akcji:** Widoczny przycisk z etykietą "Generuj Przepisy". Przycisk powinien wizualnie reagować na interakcję użytkownika (np. zmiana koloru po najechaniu myszą).
- **Wizualizacja Procesu:**
  - Po aktywacji przycisku, interfejs musi wyraźnie komunikować stan przetwarzania. Oczekiwany jest komunikat tekstowy (np. "Przetwarzam Twoje składniki...") wraz z graficznym wskaźnikiem ładowania.
- **Prezentacja Wyników:**
  - Wygenerowane przepisy muszą być zaprezentowane w formie oddzielnych kart. Każda karta powinna być wizualnie odseparowana, posiadać zaokrąglone rogi i subtelny cień, aby stworzyć wrażenie głębi.
  - **Zawartość Karty Przepisu:**
    - **Nazwa Przepisu:** Wyeksponowana na górze karty, pogrubioną czcionką.
    - **Lista Składników:** Poprzedzona nagłówkiem "Składniki:". Każdy składnik powinien być przedstawiony w nowej linii.
    - **Instrukcje Przygotowania:** Poprzedzone nagłówkiem "Instrukcje:". Treść powinna być sformatowana jako czytelny paragraf lub lista kroków.

Każda wygenerowana karta przepisu musi posiadać dwie dodatkowe, niezależne funkcje:

- **Funkcja Modyfikacji Przepisu:**
  - Uruchamiana przyciskiem z etykietą "Modyfikuj Przepis" oraz ikoną symbolizującą Gemini AI.

    - Po aktywacji, udostępnia użytkownikowi pole tekstowe do wprowadzenia prośby o modyfikację (np. "Poproszę o wersję wegańską" lub "Zaproponuj zamiennik dla makaronu").
    - Po zatwierdzeniu prośby, treść na danej karcie (nazwa, składniki, instrukcje) powinna zostać zaktualizowana zgodnie z wprowadzonym poleceniem.
- **Funkcja Drukowania:**
    - Uruchamiana przyciskiem z etykietą "Drukuj Przepis" oraz ikoną drukarki.
    - Aktywacja tej funkcji powinna otworzyć okno dialogowe drukowania z wersją przepisu przystosowaną do wydruku. Format wydruku powinien zawierać wyłącznie treść przepisu (nazwę, składniki, instrukcje), bez elementów interfejsu strony takich jak przyciski czy nagłówki.

Strona ma być estetyczna tj. projekt powinien być minimalistyczny i estetyczny, a aplikacja musi być w pełni funkcjonalna i czytelna zarówno na ekranach komputerów stacjonarnych, jak i na urządzeniach mobilnych (smartfony, tablety).

Zmodyfikuj istniejący kod aplikacji "Kreator Przepisów", aby każdy wygenerowany przepis zawierał dodatkowe, kluczowe informacje:

1. **Czas przygotowania i poziom trudności.**
2. **Szacunkowe wartości odżywcze** (kalorie, białko, tłuszcze, węglowodany).
3. Listę potencjalnych alergenów.