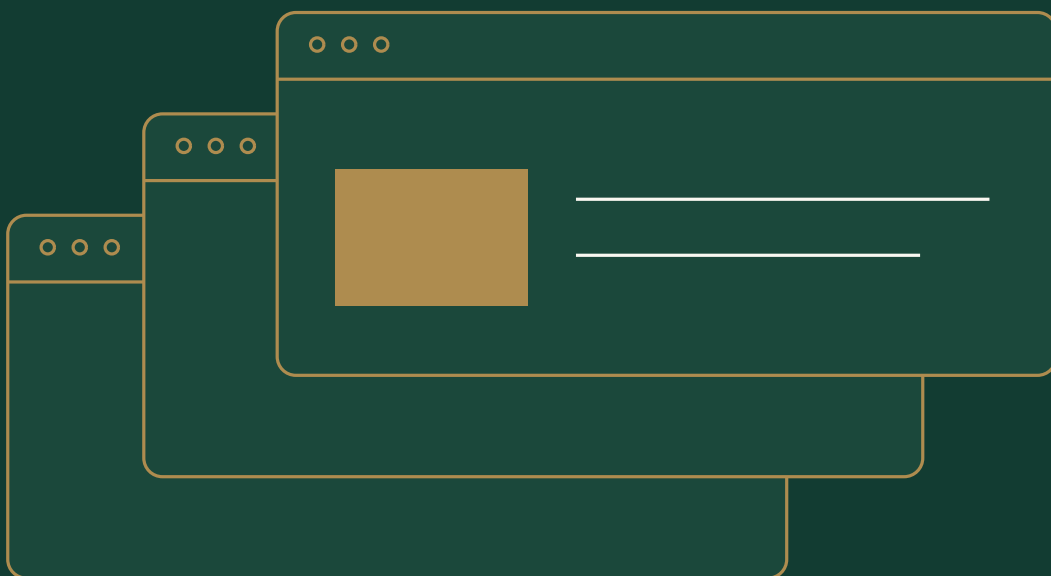


WŁASNA APLIKACJA Z AI

Od pomysłu do działającego prototypu



PROJEKT TWORZY CZŁOWIEK.
AI WSPIERA PROCES.

02

MAPA PORADNIKA

Spis treści

Aplikacja zaczyna się od potrzeby	3
Opisz problem, zanim opiszesz funkcje	4
MVP: mała wersja, pełny obieg pracy	5
Od zdania do sprawdzonej wersji	6
Zaprojektuj ekran do jednej pracy	7
Dane: co właściwie zapisujemy	8
Wygeneruj kompletny plik aplikacji	9
Zapisz HTML i otwórz go w przeglądarce	10
Opcjonalnie: prosty lokalny serwer	11
Sprawdź zachowanie na konkretnych danych	12
Zgłaszaj błąd tak, by dało się go odtworzyć	13
Poprawiaj użyteczność małymi krokami	14
Zapis lokalny ma granice	15
Eksport: dane poza przeglądarką	16
Import: najpierw sprawdź, potem zastąp	17
Przećwicz utratę i odzyskanie	18
Bezpieczeństwo: sprawdź wejście i wyjście	19
Przenieś plik na statyczny hosting	20
Odbierz publikację jak nową wersję	21
Rozwijaj na podstawie użycia	22
Twój samodzielny sprint	23
Źródła i ścieżka dalszej nauki	24

Kliknij tytuł, aby przejść do rozdziału. Kliknięcie nazwy poradnika w stopce wraca do spisu.

WPROWADZENIE

Aplikacja zaczyna się od potrzeby

AI może pomóc przełożyć pomysł na działającą stronę. Ty określasz jej zadanie i sprawdzasz, czy wynik rzeczywiście pomaga.

W tym poradniku zbudujesz prostą aplikację WWW do zarządzania własnymi zadaniami. Będzie mieć pole tytułu, dwa statusy, filtry i zapis w przeglądarce. Następnie dodasz eksport oraz import pliku JSON, przygotujesz kopię danych i poznasz drogę do publikacji na statycznym hostingu. Wszystko zacznie się od jednego pliku index.html, bez instalowania rozbudowanego środowiska programistycznego.

Nie musisz rozumieć każdej linijki kodu, aby wykonać pierwszy prototyp. Musisz jednak umieć opisać oczekiwane zachowanie, zachować poprzednią wersję i odtworzyć błąd. AI może wygenerować kod, ale jego odpowiedź jest propozycją wymagającą sprawdzenia. Ładny ekran nie potwierdza poprawnego zapisu, bezpiecznego wyświetlania treści ani możliwości odzyskania danych.

Co będzie oznaczać sukces

- Dodajesz zadanie „Przygotować plan warsztatu” i widzisz je na liście.
- Zmieniasz status na „Gotowe”, a filtry pokazują właściwe zadania.
- Po odświeżeniu zadania pozostają w wybranym środowisku przeglądarki.
- Eksportujesz listę, usuwasz dane testowe i odtwarzasz je z kopii.

Pracuj wyłącznie na fikcyjnych zadaniach. To aplikacja osobista, bez kont, współdzielenia i synchronizacji między urządzeniami. AI pomaga ją stworzyć; gotowa aplikacja nie musi zawierać modelu AI ani wysyłać mu danych. Przygotuj przeglądarkę, edytor zwykłego tekstu oraz osobny folder projektu. Zaplanuj kilka krótkich sesji: opis potrzeb, pierwsza wersja, testy i zabezpieczenie danych. Przechodź dalej dopiero wtedy, gdy potrafisz samodzielnie powtórzyć poprzedni krok.

Na każdej stronie znajdziesz decyzję lub działanie do wykonania. Nie pomijaj prób odzyskania danych: należą do projektu tak samo jak ekran.

POMYSŁ

Opisz problem, zanim opiszesz funkcje

Dobre polecenie dla AI odpowiada na pytania: kto korzysta, co robi dziś i po czym pozna, że jest łatwiej.

Ogólne „zrób aplikację do produktywności” zostawia modelowi zbyt wiele decyzji. W naszym projekcie użytkownik pracuje sam, zapisuje krótkie sprawy i chce wiedzieć, co jeszcze zostało do wykonania. Nie potrzebuje kalendarza, kont zespołu ani automatycznych powiadomień. Takie ograniczenie jest użyteczne: pozwala zakończyć pierwszy eksperyment i ocenić jego sens.

GOTOWY PROMPT / Prompt 1 • karta potrzeby

Cel: zamienić obserwację w konkretny opis potrzeb. Uzupełnij fragment [DANE] własnymi, niesensytywnymi przykładami.

Pomóż mi opisać małą aplikację WWW. [DANE] Korzystam samodzielnie na laptopie. Zadania zapisuję na luźnych kartkach i zapominam, które są zakończone. Przykłady: przygotować plan warsztatu, sprawdzić prezentację, kupić materiały. Chcę szybko dodać tytuł, oznaczyć wykonanie i zobaczyć rzeczy do zrobienia. Nie potrzebuję kont ani współpracy. Oczekiwany wynik: karta potrzeby zawierająca użytkownika, problem, trzy główne czynności, ograniczenia oraz trzy mierzalne warunki sukcesu. Nie proponuj jeszcze kodu. Weryfikacja: oddziel moje fakty od swoich założeń; każde założenie oznacz i wskaż, jak mogę je sprawdzić.

KONTROLA: Sprawdź, czy każda funkcja wynika z problemu. Wykreśl propozycje, których nie użyjesz podczas pierwszego tygodnia.

Przeczytaj wynik jak zamówienie, za które odpowiadasz. Jeśli AI dopisze synchronizację, usuń ją z briefu. Jeżeli zaproponuje terminy zadań, zapisz je na liście „później”. Poproś osobę testującą o przykład sytuacji, w której otworzyłaby taki ekran. Brak przekonującego przykładu jest sygnałem, że trzeba poprawić pomysł, zanim zaczniesz generować kod. Zapisz zatwierdzoną kartę w osobnym pliku tekstowym; dołączysz ją do kolejnych poleceń.

ZAKRES

MVP: mała wersja, pełny obieg pracy

MVP powinno obsłużyć jedną rzeczywistą czynność od początku do końca, nawet jeśli robi to skromnie.

Pierwsza wersja pozwala dodać zadanie, zmienić jego status, przefiltrować listę i usunąć niepotrzebny wpis. Zapisuje dane w localStorage. To wystarczy, aby sprawdzić codzienne użycie. Eksport i import dołożysz po przetestowaniu podstaw, zanim powierzysz aplikacji zadania, których nie chcesz stracić. W ten sposób pierwsze polecenie pozostaje czytelne, a kopia zapasowa staje się osobnym, kontrolowanym etapem.

Kontrakt pierwszej wersji

- Tytuł jest obowiązkowy, po usunięciu spacji na brzegach ma od 1 do 120 znaków.
- Status ma tylko dwie wartości: „Do zrobienia” i „Gotowe”. Nowe zadanie zaczyna od pierwszej.
- Filtry to „Wszystkie”, „Do zrobienia” i „Gotowe”. Filtr zmienia widok, nie usuwa danych.
- Każde zadanie ma własny identyfikator, nawet jeśli dwa tytuły są takie same.
- Usunięcie pojedynczego zadania wymaga potwierdzenia; nie ma jeszcze historii zmian.

Poza zakresem pozostają daty, załączniki, logowanie, udostępnianie i połączenia z zewnętrznymi usługami. Nie oznacza to, że są zbędne w każdej aplikacji. Po prostu każda z nich otwiera nowe decyzje, testy i możliwe błędy. Ustal, że podczas pierwszego etapu AI nie może samodzielnie ich dodawać.

Zapisz warunek zakończenia: wszystkie podstawowe testy działają, aplikacja pokazuje błąd zapisu, a użytkownik rozumie, gdzie znajdują się dane. Dopiero wtedy przejdź do kopii JSON. Jeśli model proponuje kilkanaście plików, bibliotekę lub bazę danych, przypomnij o jednym pliku HTML. Prostota konstrukcji umożliwia łatwiejsze cofnięcie zmiany i przeniesienie projektu.

Każdy dodatkowy warunek zapisuj jawnie, aby kolejne polecenie nie zmieniało przypadkiem wcześniejszych ustaleń.

MAPA PRACY

Od zdania do sprawdzonej wersji

Każdy etap kończy się czymś, co możesz obejrzeć lub powtórzyć. Dzięki temu nie oddajesz modelowi całego projektu naraz.

Pięć etapów budowy**01 Potrzeba**

Jedno zdanie o problemie i trzy czynności użytkownika.

02 Zakres

Mały kontrakt zachowania oraz rzeczy odłożone na później.

03 Prototyp

Jeden plik HTML uruchomiony w przeglądarce.

04 Sprawdzenie

Konkretne wejścia, oczekiwane wyniki i zapis błędów.

05 Utrwalenie

Kopia danych, wersja kodu i decyzja o publikacji.

Połączenia i numeracja pokazują kolejność pracy, ale powrót do wcześniejszego etapu jest normalny. Gdy test ujawnia, że usuwanie jest zbyt łatwe, wracasz do kontraktu zachowania. Gdy osoba testująca nie rozumie statusu, poprawiasz ekran. Nie wymieniaj od razu całego pomysłu; ustal, którą decyzję naprawdę podważył wynik.

Na końcu każdej sesji zapisz krótką notatkę: wersja pliku, co działa, co nie działa i następny pojedynczy krok. Przykład: „v01: dodawanie i filtr działają; zapis po odświeżeniu do sprawdzenia; jutro test localStorage”. To wystarczy, żeby wrócić do projektu bez odtwarzania całej rozmowy z AI.

Oddziel plik aplikacji od pliku danych. HTML zawiera instrukcje działania i wygląd; JSON będzie zawierał listę zadań. Kopia jednego nie zastępuje kopii drugiego. Przed zmianą zachowaj obie rzeczy, jeśli masz już dane warte odzyskania. Na początku możesz zapisywać wersje jako index-v01.html i index-v02.html, a aktualną wersję utrzymywać pod nazwą index.html.

INTERFEJS

Zaprojektuj ekran do jednej pracy

Użytkownik powinien od razu wiedzieć, gdzie wpisać zadanie, jak je zakończyć i gdzie znaleźć pozostałe wpisy.

Ekran ma nagłówek „Moje zadania”, krótką informację o lokalnym zapisie, podpisane pole tytułu i przycisk „Dodaj”. Pod formularzem są trzy filtry oraz lista. Każdy wiersz pokazuje tytuł, tekst statusu i przyciski zmiany statusu oraz usunięcia. Pusta lista wyjaśnia, co zrobić dalej. Pusty wynik filtra informuje, że w danej kategorii nie ma zadań.

GOTOWY PROMPT / Prompt 2 · opis interfejsu

Cel: ustalić układ przed generowaniem kodu. Wklej swoją kartę potrzeby w miejsce [DANE].

Zaprojektuj tekstowo jeden ekran aplikacji „Moje zadania”. [DANE] Osobista lista krótkich zadań; statusy Do zrobienia i Gotowe; filtry; brak kont i synchronizacji. Oczekiwany wynik: kolejność elementów od góry, etykiety przycisków, tekst pustej listy, komunikat błędnego tytułu, informacja o miejscu zapisu oraz zachowanie na telefonie. Uwzględnij podpis pola, obsługę klawiaturą, widoczny fokus i status opisany słowem, nie samym kolorem. Nie dodawaj nowych funkcji. Weryfikacja: przejdź przez dodanie, ukończenie i usunięcie zadania, wskazując element użyty w każdym kroku.

KONTROLA: Sprawdź, czy można opowiedzieć całą pracę bez zgadywania znaczenia ikon. Usuń ozdobniki utrudniające odczytanie listy.

W pierwszej wersji wybierz spokojny wygląd: czytelny tekst, wyraźne przyciski i odstępy między wierszami. Węższy ekran powinien układać elementy jeden pod drugim, bez poziomego przewijania. Przycisk usuwania nie powinien dominować nad zmianą statusu. Po dodaniu zadania pole można wyczyścić i przywrócić do niego fokus, aby kolejny wpis był szybki. Te drobne decyzje wpływają na użyteczność bardziej niż efektowna animacja.

MODEL DANYCH

Dane: co właściwie zapisujemy

Przeglądarka potrzebuje jednoznacznej struktury. Zrozumienie trzech pól ułatwi testy, import i późniejsze zmiany.

Pojedyncze zadanie opisują: id, title i status. Id jest unikalnym tekstowym identyfikatorem; tytuł widzisz na ekranie; status przyjmuje wartość todo albo done. Polska etykieta statusu należy do interfejsu. Dzięki temu nie trzeba zmieniać danych, gdy poprawisz napis „Gotowe”. Dwa wpisy o takim samym tytule nadal są różnymi zadaniami, ponieważ mają inne id.

Ustal format kopii

Plik kopii ma obiekt z version: 1 oraz tasks: listą zadań. Przykładowa zawartość: `{"version":1,"tasks":[{"id":"t1","title":"Sprawdzić prezentację","status":"todo"}]}`. JSON to format tekstowy; nazwy pól i wartości tekstowe są w podwójnych cudzysłowach. Nie poprawiaj ręcznie większej kopii bez zachowania oryginału. Nawet brak jednego przecinka może sprawić, że plik nie da się odczytać.

Aplikacja używa własnego klucza localStorage, na przykład labpercepcji.tasks.v1. Przy odczycie sprawdza, czy zapis ma oczekiwaną strukturę. Uszkodzonych danych nie należy automatycznie zastępować pustą listą: użytkownik powinien zobaczyć komunikat i mieć możliwość sięgnięcia po kopię. Brak zapisu oznacza pierwszy start i pustą listę; błędny zapis oznacza problem do wyjaśnienia.

- Limit projektu: maksymalnie 500 zadań i 120 znaków tytułu.
- Każdy status spoza todo i done jest błędem danych.
- Duplikaty identyfikatorów trzeba odrzucić podczas importu.
- Nie dodawaj haseł, tokenów, danych klientów ani dokumentów do struktury.

Limity są umową naszego prototypu, a nie deklaracją pojemności przeglądarki. Ułatwiają przewidywalne zachowanie i sensowne testy. Zapisz je w briefie, aby model nie wybierał innych wartości podczas kolejnej poprawki.

Gdy model zmieni nazwy pól, zatrzymaj zmianę i poproś o zachowanie uzgodnionego formatu.

W PRAKTYCE

Wygeneruj kompletny plik aplikacji

Poniższe polecenie łączy zakres, ekran i reguły danych. Wklej je do wybranego asystenta AI obsługującego tekst oraz kod.

GOTOWY PROMPT / Prompt 3 • pierwsza wersja index.html

Cel: wygenerować kompletny, lokalny plik zgodny z ustalonym kontraktem.

Stwórz pełny index.html: HTML, CSS i JavaScript wewnątrz, interfejs po polsku. [DANE] „Moje zadania”: bez kont, bibliotek i sieci. Zadanie: unikalne tekstowe id, title 1–120 znaków po trim(), status todo/done; limit 500 wpisów. Dodawanie przyciskiem i Enterem, statusy, trzy filtry, usuwanie po potwierdzeniu. localStorage: klucz labpercepcji.tasks.v1, format {version:1,tasks:[...]}, zapis po zmianie, odczyt przy starcie. Waliduj dane; obsługuj błędy odczytu/zapisu komunikatem; uszkodzonego zapisu nie nadpisuj. Renderuj tytuły przez textContent. Podpis pola, tekstowe statusy, fokus, klawiatura, responsywność, puste widoki. Informacja: dane lokalne, brak synchronizacji, potrzebna kopia. Eksport/import później. Oczekiwany wynik: kompletny blok od <!doctype html> do </html>, bez skrótów, oraz instrukcja uruchomienia. Weryfikacja: pięć testów manualnych; nie twierdź, że wykonano je.

KONTROLA: Sprawdź kompletność pliku i brak zależności od dodatkowych zasobów.

Jeżeli odpowiedź urwie się w połowie, poproś o ponowne przesłanie całego pliku w jednej odpowiedzi. Nie sklejaj przypadkowo dwóch wariantów: mogą mieć różne nazwy funkcji lub strukturę danych. Zachowaj również samo polecenie. To twoja specyfikacja, do której wrócisz, gdy model zaproponuje zmianę wykraczającą poza projekt.

Dołącz również kartę potrzeb i kontrakt ze wcześniejszych stron. Jeśli zapis się nie powiedzie, ekran nie może sugerować, że dane są utrwalone. Przejrzyj objaśnienia AI: przyjęte założenia powinny odpowiadać regułom projektu. Gdy pojawiają się nowe funkcje, poproś o ich usunięcie przed zapisaniem.

ZRÓB SAM

Zapisz HTML i otwórz go w przeglądarce

Pierwsze uruchomienie zaczyna się od zwykłego pliku. Rozszerzenie i tryb zapisu mają większe znaczenie niż nazwa edytora.

- Utwórz folder „moje-zadania”. W nim będziesz trzymać aplikację, wersje kodu i kopie testowe.
- Otwórz edytor zwykłego tekstu. Na macOS wybierz tryb zwykłego tekstu; dokument sformatowany nie jest odpowiednim miejscem na kod.
- Skopiuj tylko zawartość bloku kodu: od `<!doctype html>` do `</html>`. Nie kopiuj znaczników `` ani objaśnień asystenta.
- Zapisz plik jako `index.html`, w kodowaniu UTF-8. Jeśli edytor proponuje typ pliku, wybierz wszystkie pliki lub HTML.
- Sprawdź pełną nazwę. `index.html.txt` nadal jest plikiem tekstowym; włącz pokazywanie rozszerzeń w menedżerze plików.
- Otwórz `index.html` w przeglądarce, na przykład przeciągając plik do jej okna.

Powinieneś zobaczyć formularz i pustą listę, a w adresie początek `file://`. Dodaj próbny tytuł „Pierwsze zadanie”. Jeżeli widzisz kod zamiast aplikacji, sprawdź rozszerzenie oraz to, czy kod nie został zapisany jako tekst z dodatkowymi znacznikami. Jeżeli ekran jest pusty, wróć do kompletności odpowiedzi i sprawdź konsolę według kolejnego etapu diagnozy.

Otwieranie z `file://` jest wygodne do pierwszego obejrzenia. Trwałość `localStorage` dla takich adresów nie jest jednak jednolicie określona i może różnić się między przeglądarkami. Nie traktuj pozytywnego testu jako obietnicy trwałości. Przed dalszym użyciem przejdź na lokalny serwer albo sprawdź docelowy adres hostingu.

Zrób kopię `index-v01.html` przed pierwszą poprawką. Sama rozmowa z AI nie jest wygodną kopią roboczego pliku. Folder z projektem powinien pozwalać wrócić do poprzedniego działania bez ponownego generowania całej aplikacji.

URUCHOMIENIE

Opcjonalnie: prosty lokalny serwer

Adres `http://localhost` daje bardziej przewidywalne środowisko testów niż otwieranie pliku bezpośrednio z dysku.

Ta ścieżka wymaga zainstalowanego Pythona 3. Jeśli go nie masz, możesz pozostać przy pierwszym podglądzie pliku i przeprowadzić test zapisu po publikacji statycznej. Nie pobieraj przypadkowych „naprawiaczy” środowiska. Python pobieraj z oficjalnej strony `python.org`, jeśli zdecydujesz się go instalować.

- Otwórz Terminal na macOS lub Linux albo PowerShell na Windows.
- Wpisz `cd`, spację i ścieżkę do folderu projektu w cudzysłowie, np. `cd "C:\Users\Anna\Documents\moje-zadania"`. Na macOS przykład to `cd "/Users/anna/Documents/moje-zadania"`.
- Na Windows uruchom: `py -m http.server 8000 --bind 127.0.0.1`. Na macOS i Linux: `python3 -m http.server 8000 --bind 127.0.0.1`.
- Zostaw okno terminala otwarte. W przeglądarce wpisz dokładnie `http://localhost:8000/index.html`.
- Po pracy zatrzymaj serwer klawiszami `Ctrl+C` w terminalu.

Serwer udostępnia pliki z folderu, w którym został uruchomiony. Dlatego używaj osobnego folderu projektu, bez dokumentów osobistych. Parametr `--bind 127.0.0.1` ogranicza nasłuchiwanie do tego komputera. Jeśli port jest zajęty, wybierz 8001 w poleceniu i w adresie. Komunikat o nierozpoznanym `py` lub `python3` oznacza, że wymagane polecenie nie jest dostępne; wróć do instrukcji instalacji, zamiast zmieniać kod aplikacji.

Adres i port mają znaczenie dla miejsca zapisu. `localhost:8000` oraz `localhost:8001` to różne przestrzenie danych. Także `127.0.0.1` i `localhost` nie są tym samym adresem dla `localStorage`. Używaj jednego adresu podczas całej serii testów. Pusty ekran po zmianie adresu nie musi oznaczać utraty poprzednich zadań.

Po zapisaniu nowej wersji HTML po prostu odśwież stronę w przeglądarce. Nie uruchamiaj drugiego serwera, jeśli pierwszy nadal działa poprawnie.

TESTY AKCEPTACYJNE

Sprawdź zachowanie na konkretnych danych

Test opisuje wejście i oczekiwany wynik. Zapisz rzeczywisty rezultat, zamiast oceniać aplikację wyłącznie na podstawie wyglądu.

Czynność / wejście	Oczekiwany wynik
Dodaj „Przygotować plan warsztatu”.	Jeden nowy wpis ze statusem Do zrobienia; pole zostaje wyczyszczone.
Spróbuj dodać pusty tytuł oraz trzy spacje.	Brak nowego wpisu; czytelny komunikat o wymaganym tytule.
Dodaj „Sprawdzić slajdy ”.	Tytuł bez spacji na początku i końcu.
Dodaj dwa wpisy „Telefon”. Zakończ tylko drugi.	Pierwszy pozostaje Do zrobienia, drugi jest Gotowe.
Wybierz Gotowe, następnie Do zrobienia.	Każdy filtr pokazuje tylko odpowiednią kategorię.
Odśwież stronę pod tym samym adresem.	Tytuły i statusy pozostają, jeśli zapis działa w tym środowisku.
Usuń wpis i anuluj potwierdzenie.	Wpis nadal istnieje. Ponowna próba z potwierdzeniem go usuwa.

Przed testami zanotuj nazwę i wersję przeglądarki, adres aplikacji i wersję pliku. Zaczynij od pustej listy albo dokładnie zapisanej listy startowej. Po każdej czynności odnotuj „zgodne” lub „niezgodne” oraz krótki opis różnicy. Zrzut ekranu pomaga przy wyglądzie, ale nie zastępuje kroków odtworzenia błędu.

Dodatkowo wklej tytuł złożony ze 121 znaków: powinien zostać odrzucony. Tytuł o długości 120 znaków powinien zostać przyjęty. Użyj zwykłych liter, aby uniknąć niejasności liczenia złożonych emoji. Sprawdź Enter w polu oraz przechodzenie klawiszem Tab między kontrolkami. Zmniejsz szerokość okna i upewnij się, że przyciski nie wypadają poza ekran.

Przejsięcie tych testów jest podstawą kolejnego etapu. Nie świadczy jeszcze o pełnym bezpieczeństwie ani odporności na każdą awarię. Jeśli odświeżenie gubi wpisy, najpierw ustal sposób uruchomienia i ewentualny błąd zapisu; dopiero później poproś o poprawkę kodu.

POPRAWKI

Zgłaszaj błąd tak, by dało się go odtworzyć

Model potrzebuje objawu, sekwencji działań i aktualnego pliku. Sama wiadomość „nie działa” zwykle prowadzi do zgadywania.

Najpierw powtórz problem. Czy dotyczy wszystkich filtrów, jednego zadania czy tylko odświeżenia? Czy występuje pod file:// i pod lokalnym serwerem? Otwórz narzędzia deweloperskie przeglądarki z jej menu i wybierz Konsolę. Skopiuj pierwszy istotny komunikat błędu, jeśli istnieje. Brak czerwonego komunikatu nie dowodzi, że zachowanie jest poprawne.

GOTOWY PROMPT / Prompt 4 · diagnoza bez zgadywania

Cel: ustalić przyczynę jednego błędu i uzyskać kontrolowaną poprawkę. Zastąp pola w nawiasach własnymi obserwacjami.

Pomóż zdiagnozować błąd w pojedynczym HTML. [DANE] Przeglądarka: [nazwa i wersja]. Adres: [file:// lub dokładny localhost]. Wersja: [v01]. Stan początkowy: [dwa zadania]. Kroki: [dodaj wpis, odśwież]. Oczekiwałem: [wpis pozostaje]. Otrzymałem: [lista pusta]. Konsola: [komunikat albo brak]. Poniżej wklejam cały aktualny kod: [KOD]. Oczekiwany wynik: najbardziej prawdopodobna przyczyna oparta na kodzie, krótki test rozróżniający inne możliwości i dopiero potem pełny poprawiony plik. Zachowaj funkcje, klucz zapisu i format danych. Nie ukrywaj błędów pustą listą. Weryfikacja: podaj kroki ponownego testu błędu oraz dwa testy wcześniejszych funkcji. Nie deklaruje, że testy wykonano.

KONTROLA: Porównaj diagnozę ze swoimi obserwacjami. Jeśli model nie ma potrzebnych danych, uzupełnij raport zamiast akceptować losową przebudowę.

Przed podmianą pliku zachowaj kopię kodu i, gdy eksport już działa, kopię danych. Zmieniaj jedną przyczynę naraz. Po poprawce powtórz test, który wcześniej się nie powiódł, oraz dodawanie i filtrowanie. Gdy błąd nadal występuje, przekaż rzeczywisty wynik tej próby. Nie proś od razu o „lepszą wersję wszystkiego”.

INTERFEJS W PRAKTYCE

Poprawiaj użyteczność małymi krokami

Gdy podstawowe funkcje działają, sprawdź, czy osoba korzystająca potrafi ich użyć bez twoich objaśnień.

Daj komuś trzy zadania: dodać wpis, oznaczyć go jako gotowy i pokazać tylko rzeczy do zrobienia. Poproś o mówienie na głos, czego szuka. Nie wskazuj palcem przycisków. Zanotuj momenty zawahania, niewidoczny status oraz miejsca, w których osoba pomyliła filtrowanie z usuwaniem. To materiał do zmiany interfejsu, a nie ocena umiejętności testującego.

GOTOWY PROMPT / Prompt 5 · jedna poprawka ekranu

Cel: usunąć zaobserwowany problem bez rozszerzania zakresu. Wklej aktualny kod i krótki opis próby.

Popraw użyteczność aplikacji przy zachowaniu działania i danych. [DANE] W próbie użytkownik nie zauważył aktywnego filtra i uznał, że zadania zniknęły. Aktualny kod: [KOD]. Oczekiwany wynik: pełny plik HTML z wyraźną nazwą aktywnego filtra, liczbą widocznych zadań oraz komunikatem pustego wyniku. Stan filtra ma być rozpoznawalny tekstowo i programowo, nie tylko kolorem. Zachowaj podpisy pól, widoczny fokus i obsługę klawiatury. Nie zmieniaj klucza localStorage, formatu danych ani zasad usuwania. Weryfikacja: pokaż, jak sprawdzić trzy filtry, pusty wynik i przejście klawiszem Tab. Osobno wypisz dokładnie, jakie fragmenty zachowania zmieniłeś.

KONTROLA: Sprawdź, czy poprawka odpowiada obserwowanemu problemowi i nie wprowadza funkcji, których nie zamawiałeś.

Po podmianie kodu powtórz próbę na wąskim ekranie i przy powiększeniu strony do 200%. Tytuły powinny się zawijać, a przyciski pozostawać dostępne. Przejdź całą podstawową pracę samą klawiaturą. Nie usuwaj obramowania fokusu tylko dlatego, że wydaje się mniej eleganckie. Wyraźny komunikat po dodaniu albo błędzie pomaga użytkownikowi zrozumieć wynik działania. Jeśli zmiana nie usuwa problemu, wróć do obserwacji i sprawdź, czy przyczyna została dobrze nazwana.

TRWAŁOŚĆ DANYCH

Zapis lokalny ma granice

localStorage przechowuje dane dla konkretnego miejsca w przeglądarce. To wygodne do prototypu, ale wymaga świadomego zarządzania kopiami.

Lista nie przechodzi automatycznie do innej przeglądarki, profilu ani urządzenia. W trybie prywatnym może zniknąć po zamknięciu sesji. Usunięcie danych witryny usuwa również lokalny zapis. Zmiana adresu, portu lub protokołu może otworzyć inną przestrzeń danych, dlatego po publikacji nie zobaczysz automatycznie listy z localhost.

Jak powinien wyglądać błąd zapisu

Aplikacja może działać na ekranie, choć przeglądarka odmówiła utrwalenia zmiany. Oczekuj wyraźnego komunikatu: „Nie udało się zapisać zmian. Zrób kopię i nie zamykaj strony”. Model powinien obsłużyć zarówno niedostępność magazynu, jak i wyjątek podczas zapisu. Nie akceptuj komunikatu „Zapisano”, jeśli kod go pokazuje niezależnie od wyniku.

Po każdej istotnej poprawce powtórz cykl: dodaj zadanie, zmień status, odśwież i sprawdź wynik. Następnie zamknij kartę, otwórz ten sam adres i ponownie sprawdź listę. Próby wykonuj w zwykłym profilu, który będzie używany na co dzień. Inne ustawienia prywatności mogą dać odmienny rezultat.

- Używaj jednego stałego adresu podczas testowania.
- Zanotuj, gdzie działa sprawdzona wersja: przeglądarka, profil, adres.
- Trzymaj oddzielny plik kopii poza danymi witryny.
- Przed czyszczeniem przeglądarki eksportuj i sprawdź kopię.

To nie jest magazyn na poufne informacje. Kod działający w tej samej przestrzeni witryny może mieć dostęp do danych. Błąd typu XSS może umożliwić ich odczyt. Brak serwera z bazą danych nie stanowi sam w sobie gwarancji prywatności ani bezpieczeństwa.

Przy niepewnym wyniku testu zachowaj otwartą kartę i wyeksportuj aktualną listę, zanim zaczniesz kolejne próby.

KOPIA JSON

Eksport: dane poza przeglądarką

Kopia ma sens dopiero wtedy, gdy istnieje jako osobny plik i potrafisz sprawdzić, co zawiera.

Po podstawowych testach poproś AI o dodanie przycisku „Eksportuj JSON”. Eksport powinien pobierać całą listę, również zadania ukryte przez filtr. Format pozostaje zgodny z wcześniejszą umową: `version: 1` i `tasks`. Nazwa pobieranego pliku może zawierać datę oraz godzinę, na przykład `zadania-2026-10-13-1530.json`. To pomaga odróżniać kopie bez otwierania każdej z nich.

Polecenie do rozbudowy

Napisz: „Dodaj eksport wszystkich zadań do pobieranego pliku JSON `{version:1,tasks:[...]}`, bez wysyłania danych do sieci. Zachowaj aktualny kod i format. Eksport ma działać także przy aktywnym filtrze. Po błędzie zapisu `localStorage` umożliwiał eksport aktualnej listy z pamięci. Pokaż pełny HTML oraz trzy testy eksportu”. Dołącz obecny plik; nie zakładaj, że model pamięta wcześniejszą wersję.

- Dodaj trzy zadania i zakończ jedno. Wybierz filtr Gotowe, a następnie eksport.
- Otwórz pobrany JSON w edytorze tekstu. Sprawdź, że zawiera wszystkie trzy tytuły i prawidłowe statusy.
- Wyeksportuj pustą listę. Oczekuj poprawnego pliku z `tasks: []`, a nie uszkodzonego dokumentu.
- Przenieś kopię do folderu kopii projektu i zachowaj jej oryginalną nazwę.

Nie musisz ręcznie rozumieć całego JSON, ale liczba wpisów i kilka tytułów powinny zgadzać się z listą. Nie otwieraj kopii w przypadkowym internetowym walidatorze, jeśli zawiera prywatne sprawy. Na tym etapie plik jest kandydatem na kopię zapasową. Dopiero udany import i porównanie odtworzonej listy potwierdzą, że nadaje się do odzyskania danych.

Przed pobraniem sprawdź też, czy aplikacja nie dołącza przypadkowych informacji spoza ustalonej struktury kopii.

ODTWARZANIE

Import: najpierw sprawdź, potem zastąp

Import zastępujący listę jest prostszy do zrozumienia niż automatyczne łączenie. Wymaga jednak walidacji i jasnego potwierdzenia.

GOTOWY PROMPT / Prompt 6 • kontrolowany import

Cel: odtworzyć listę po sprawdzeniu kopii.

Dodaj import JSON do tej aplikacji. [DANE] Kod: [KOD]. Format: {version:1,tasks:[{id,title,status}]}. Import zastępuje całą listę. Limit pliku 1 MB, maksymalnie 500 zadań. Sprawdź przed zmianą: obiekt główny, version równe 1, tasks jako tablica, każde id jako niepusty tekst i unikalne, title jako tekst długości 1–120 po trim(), status wyłącznie todo/done. Kopiuj tylko dozwolone pola do nowych obiektów. Odrzuć całość przy błędzie; zachowaj bieżącą listę i zapis. Po walidacji pokaż liczbę importowanych wpisów i poproś o potwierdzenie zastąpienia, przypominając o eksporcie obecnych danych. Jeśli użytkownik anuluje, nic nie zmieniaj. Dopiero po udanym zapisie podmień listę na ekranie. Tytuły renderuj jako tekst. Oczekiwany wynik: pełny HTML, jasne komunikaty błędów i instrukcja odtwarzania. Weryfikacja: test poprawnego pliku, złego JSON, nieznanej wersji, duplikatu id, anulowania i pustej tablicy. Nie twierdź, że testy wykonano.

KONTROLA: Sprawdź, czy błędny plik zachowuje listę.

Przy imporcie poprawnej pustej listy komunikat powinien wyraźnie powiedzieć, że zastąpienie usunie wszystkie obecne zadania. Nie pomijaj potwierdzenia tylko dlatego, że plik ma prawidłową składnię. Poprawny JSON może zawierać niewłaściwe dane; samo JSON.parse nie sprawdza reguł aplikacji. Po imporcie odśwież stronę i ponownie porównaj tytuły oraz statusy. Próby rób na kopiach testowych, zanim zastąpisz używaną listę.

Przy wyborze pliku sprawdź jego nazwę i datę, aby przypadkiem nie odtworzyć wcześniejszego stanu.

PROCEDURA KOPII

Przećwicz utratę i odzyskanie

Odwracalna próba na fikcyjnych danych pokaże więcej niż samo posiadanie przycisku Eksportuj.

Obieg kopii zapasowej

01 Przygotuj

Trzy testowe zadania, zapisane tytuły i statusy.

02 Wyeksportuj

Osobny JSON; sprawdź liczbę wpisów i nazwę pliku.

03 Usuń

Usuń testowe wpisy przyciskami aplikacji i odśwież.

04 Odtwórz

Importuj kopię, potwierdź zastąpienie i odśwież.

05 Porównaj

Te same zadania oraz statusy oznaczają zgodny wynik próby.

Wykonuj ćwiczenie dopiero po sprawdzeniu eksportu i importu. Nie zaczynaj od usuwania danych witryny w ustawieniach przeglądarki. Najprościej usunąć trzy testowe zadania przyciskami aplikacji. Jeśli później dodasz „Usuń wszystkie dane”, żądaj osobnego potwierdzenia i usuwania wyłącznie klucza aplikacji, bez wywołania `localStorage.clear()`.

Zachowuj co najmniej ostatnią poprawną kopię oraz poprzednią. Nowy eksport nie powinien automatycznie nadpisywać jedyne go pliku, bo może zawierać przypadkowo pustą listę. Przy codziennym użyciu przyjmij prosty rytm: kopia po istotnych zmianach i obowiązkowo przed poprawką kodu, importem lub czyszczeniem przeglądarki.

Plik HTML pozwala odzyskać aplikację, a JSON jej dane. Przechowuj je razem jako komplet, opisując wersję kodu i format danych. Jeżeli kopia zawiera sprawy osobiste, traktuj ją jak zwykły prywatny dokument: pilnuj miejsca przechowywania i udostępniania. Eksport nie szyfruje automatycznie zawartości i nie przywraca pliku, który został usunięty ze wszystkich miejsc.

RYZYKA

Bezpieczeństwo: sprawdź wejście i wyjście

Tytuł zadania i importowany plik są danymi wejściowymi. Aplikacja powinna je kontrolować i wyświetlać bez uruchamiania jako kod.

Wpisz jako tytuł dokładnie: `<img src=x onerror=alert(1)>`. Oczekiwany wynik to widoczny tekst z nawiasami, bez obrazka i bez okna alert. Sprawdź ten sam tytuł po odświeżeniu oraz po eksporcie i imporcie. Jeśli pojawi się alert, zakończ próbę i poproś o poprawienie renderowania danych przez `textContent`. Nie używaj aplikacji z prawdziwymi informacjami przed usunięciem problemu.

To kontrolowany test konkretnego wzorca, nie pełny audyt. Nie dowodzi odporności na wszystkie rodzaje XSS. Poproś AI o wskazanie każdego miejsca, w którym tytuł lub zawartość pliku trafia na ekran. W tych miejscach dane nie powinny być sklejane z kodem HTML. Także komunikat błędu importu nie może bezpośrednio wstawiać nieznanej zawartości jako HTML.

Granice prototypu

- Nie przechowuj haseł, kluczy API, tokenów ani danych wrażliwych w kodzie lub `localStorage`.
- Nie wklejaj danych klientów do rozmowy z AI; przekazuj przykłady fikcyjne.
- Nie dodawaj analityki, reklam ani zewnętrznych skryptów bez świadomej decyzji.
- Importuj własne lub sprawdzone pliki; walidacja powinna działać również dla nich.

Kod po publikacji jest dostępny dla odwiedzających. Sekret umieszczony w HTML nie staje się ukryty dzięki trudnej nazwie zmiennej. Jeśli w przyszłości aplikacja ma wywoływać płatne API albo obsługiwać poufne dane wielu osób, potrzebuje innej architektury oraz fachowego przeglądu. Nie rozszerzaj tego pojedynczego pliku o prowizoryczne logowanie. Hasło sprawdzane wyłącznie przez JavaScript w przeglądarce nie tworzy rzeczywistej kontroli dostępu.

PUBLIKACJA

Przenieś plik na statyczny hosting

Stacyjny hosting udostępnia plik pod adresem HTTPS. Sama publikacja nie dodaje kont ani wspólnej listy zadań.

Przejdźcie do adresu WWW**01 Zachowaj**

Kopia działającego HTML i aktualny eksport JSON.

02 Przygotuj

Folder z plikiem index.html; bez prywatnych kopii danych.

03 Udostępnij

Wgraj plik metodą obsługiwaną przez wybrany hosting.

04 Sprawdź

Otwórz adres HTTPS i powtórz testy akceptacyjne.

05 Przenieś dane

Importuj kopię w nowej przestrzeni przeglądarki.

Wybierz usługę obsługującą statyczne strony HTML. Sposób przesłania zależy od usługi: może to być panel wgrywania, archiwum folderu lub repozytorium. Ten poradnik nie wymaga zakładania konta. Jeśli masz dostęp do hostingu, użyj jego oficjalnej instrukcji publikacji pojedynczego pliku i sprawdź zakres publicznego udostępnienia.

Do publikowanego folderu włóż tylko aplikację. Kopie JSON, raporty błędów i wcześniejsze wersje z prywatnymi komentarzami trzymaj poza nim. Przed wysłaniem przeczytaj kod pod kątem danych przykładowych, adresów i ewentualnych sekretów. W tym projekcie nie potrzebujesz komendy budowania ani serwera obsługującego bazę danych.

Nowy adres oznacza osobny lokalny magazyn; początkowo lista może być pusta. Przenieś ją przez eksport i import. Każda osoba odwiedzająca stronę zobaczy własne dane w swojej przeglądarce. Wysłanie linku nie udostępnia twojej listy. Sprawdź docelowy adres także na telefonie, używając danych testowych, zanim uznasz publikację za gotową do codziennego użycia.

CHECKLISTA

Odbierz publikację jak nową wersję

Wynik lokalny nie wystarczy. Pod adresem docelowym mogą ujawnić się problemy z plikiem, pamięcią przeglądarki lub aktualizacją.

- Adres zaczyna się od HTTPS i otwiera właściwą wersję aplikacji, bez listy plików projektu.
- Formularz, statusy, filtry i usuwanie działają zgodnie z testami podstawowymi.
- Dodane dane pozostają po odświeżeniu i ponownym otwarciu tego samego adresu.
- Eksport zawiera także wpisy ukryte przez filtr; import odtwarza komplet danych.
- Błędny import i anulowanie nie zmieniają istniejącej listy.
- Test z tytułem `<img src=x onerror=alert(1)>` wyświetla tekst i nie uruchamia alertu.
- Klawiatura, powiększenie i wąski ekran pozwalają wykonać podstawową pracę.
- Na ekranie jest informacja o lokalnym zapisie, braku synchronizacji i potrzebie kopii.

Zapisz wynik odbioru z datą, adresem, wersją pliku i przeglądarką. Jeśli widzisz starszy wygląd po podmianie pliku, spróbuj odświeżenia pomijającego pamięć podręczną według pomocy swojej przeglądarki. Sprawdź, czy hosting rzeczywiście przyjął nowy plik. Nie usuwaj danych witryny jako pierwszej metody naprawy, bo możesz skasować listę.

Przed aktualizacją wyeksportuj dane i zachowaj poprzedni HTML. Po aktualizacji odśwież i wykonaj krótki zestaw: dodanie, zmiana statusu, filtr, eksport. Jeśli aktualizacja zmienia format danych, wstrzymaj zwykłe użycie, dopóki nie masz planu migracji i odzyskania kopii. Cofnięcie kodu nie zawsze cofa zmianę danych.

Dopiero po własnej próbie opisuj wersję jako sprawdzoną. Poradnik podaje procedurę; nie potwierdza wykonania twojego projektu ani jego testów. W notatce odbioru odróżnij „test planowany”, „test wykonany” i „wynik zgodny”. Taki zapis jest użyteczny nawet dla jednej osoby, bo pozwala bezpieczniej wrócić do pracy po kilku tygodniach.

CO DALEJ

Rozwijaj na podstawie użycia

Następna funkcja powinna rozwiązywać zaobserwowany problem. Lista pomysłów nie jest kolejką rzeczy, które trzeba natychmiast zbudować.

Po kilku dniach zanotuj, ile razy otwierasz aplikację i w którym momencie brakuje ci działania. Jeśli często poprawiasz literówki, priorytetem może być edycja tytułu. Jeśli lista rośnie, może wystarczyć wyszukiwanie. Jeśli chcesz współdzielić zadania, problem przekracza obecny model lokalnego zapisu i wymaga osobnej decyzji o architekturze.

GOTOWY PROMPT / Prompt 7 · jedna kolejna funkcja

Cel: zaplanować mały rozwój z testami i możliwością cofnięcia. Wstaw obserwacje zamiast ogólnych życzeń.

Zaplanuj następną wersję aplikacji, bez generowania kodu. [DANE] Obecnie: tytuł/status, filtry, localStorage, eksport/import JSON v1. Obserwacja: poprawiam literówki, usuwając i ponownie dodając wpis. Chcę edycję tytułu. Oczekiwany wynik: opis działania edycji, anulowania i zatwierdzania; zasady walidacji; wpływ na id, status i format kopii; trzy testy akceptacyjne oraz plan powrotu do poprzedniej wersji. Zachowaj jeden plik i brak sieci. Weryfikacja: wskaż, czy zmiana wymaga migracji danych, i nazwij ryzyka. Jeśli coś wykracza poza obserwowany problem, oznacz jako opcjonalne zamiast dodawać do zakresu.

KONTROLA: Wybierz jedną funkcję. Zanim poprosisz o kod, sprawdź, czy umiesz opisać zarówno poprawny wynik, jak i anulowanie działania.

Zachowuj dziennik wersji w trzech zdaniach: co zmieniono, jak sprawdzono i jaki problem pozostał. Nie proś modelu o kompletną przebudowę tylko dlatego, że nowy styl wygląda atrakcyjniej. Po każdej zmianie powtarzaj test odświeżenia i odzyskania kopii. Jeśli rozwój wymaga kont, płatności, synchronizacji lub danych innych osób, potraktuj go jako nowy projekt z nowymi wymaganiami. Udany prototyp pomaga je ustalić, ale nie zastępuje projektu systemu produkcyjnego.

PODSUMOWANIE

Twój samodzielny sprint

Najważniejszy rezultat to aplikacja, której zakres rozumiesz, działanie potrafisz sprawdzić, a dane umiesz odtworzyć.

Przejdź przez cały projekt w małych sesjach. W pierwszej opisz potrzebę i zakres. W drugiej wygeneruj plik, zapisz go i otwórz. W trzeciej wykonaj testy oraz popraw pojedyncze błędy. W czwartej dodaj eksport i import, a potem przećwicz usunięcie i odzyskanie danych. Publikacja jest kolejną decyzją, gdy lokalny prototyp działa wystarczająco dobrze do twojego celu.

Gotowe, kiedy możesz odpowiedzieć „tak”

- Umieć powiedzieć, dla kogo jest aplikacja i co pozostaje poza zakresem.
- Mam kompletny index.html oraz kopię poprzedniej działającej wersji.
- Potrafię dodać, zakończyć, przefiltrować i usunąć zadanie.
- Sprawdziłem zapis w wybranej przeglądarce pod stałym adresem.
- Potrafię wyeksportować i odtworzyć listę, zachowując tytuły oraz statusy.
- Znam ograniczenia lokalnego magazynu i nie wpisuję danych wrażliwych.
- Mam zapis wyników testów, a nie tylko obietnicę modelu.

Jeżeli jedna odpowiedź brzmi „nie”, wróć do odpowiedniego etapu. Nie musisz przyspieszać przez dodanie kolejnej funkcji. Dobrze zrobiony mały obieg pracy jest lepszą podstawą decyzji niż duży ekran z niesprawdzonymi przyciskami. Osoba bez doświadczenia programistycznego może prowadzić taki eksperyment, jeśli traktuje opis, test i kopię jako część budowy.

W następnym projekcie zachowaj ten sam rytm: potrzeba, mały kontrakt, plik, konkretne próby i odzyskiwanie. Zmieniają się dane i funkcje, ale odpowiedzialność za wynik pozostanie po twojej stronie. AI skraca drogę do pierwszej wersji. Twoje obserwacje nadają jej kierunek, a sprawdzanie pozwala zdecydować, kiedy można na niej polegać w zamierzonym zakresie.

DO SPRAWDZENIA

Źródła i ścieżka dalszej nauki

Oficjalna dokumentacja pomaga wyjaśniać techniczne decyzje. Używaj jej do konkretnego pytania, zamiast próbować przeczytać wszystko przed pierwszym eksperymentem.

Źródła poniżej dotyczą mechanizmów użytych w projekcie. MDN opisuje zapis w przeglądarce, lokalne uruchomienie, wyświetlanie tekstu i odczyt JSON. Dokumentacja Pythona wyjaśnia polecenie serwera. Materiał przedstawia procedurę budowy i sprawdzania; nie jest raportem z wykonania aplikacji ani audytem bezpieczeństwa wygenerowanego kodu.

Najpierw nazwij pytanie: „Dlaczego po zmianie portu lista jest pusta?” albo „Czy poprawny JSON ma prawidłową strukturę zadań?”. Otwórz odpowiadającą mu stronę dokumentacji. Poproś AI o objaśnienie fragmentu na fikcyjnym przykładzie oraz wskazanie konsekwencji dla twojego pliku. Oddziel informację ze źródła od propozycji implementacji. Link wygenerowany przez model trzeba otworzyć i porównać z odpowiedzią. Dokumentacja nie zastępuje sprawdzenia zachowania w twojej przeglądarce. Zachowaj adres użytego źródła wraz z notatką o decyzji, zwłaszcza gdy dotyczy kopii lub formatu danych.

Bibliografia

MDN: Window — localStorage —

<https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage>

MDN: How do you set up a local testing server? — https://developer.mozilla.org/en-US/docs/Learn_web_development/Howto/Tools_and_setup/set_up_a_local_testing_server

MDN: Node — textContent —

<https://developer.mozilla.org/en-US/docs/Web/API/Node/textContent>

MDN: JSON.parse() — https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/parse

Python 3: http.server — <https://docs.python.org/3/library/http.server.html>

Interfejsy hostingu mogą się zmieniać; przed publikacją korzystaj z aktualnej instrukcji operatora. Gdy potrzebujesz kont, współdzielenia lub synchronizacji, wykorzystaj opis potrzeb i wyniki testów jako brief nowego projektu. Jeden plik HTML pozostaje tu świadomym ograniczeniem ćwiczenia, które ułatwia uruchomienie, poprawianie i cofanie zmian.

Zacznij od dokumentacji problemu, który właśnie obserwujesz. Wiedza techniczna będzie rosła razem z kolejnymi próbami. Nie musisz opanować całego programowania, aby nauczyć się zamawiać małe zmiany i rzetelnie oceniać ich wynik.