

Porównanie umiejętności tworzenia aplikacji przez asystentów AI

Open AI Chat GPT vs Google Gemini

Spis treści

Zadanie 1 – Prosta witryna w przeglądarce	2
Chat GPT	2
Output	2
Rozwiązanie.....	3
Gemini	3
Output	3
Rozwiązanie.....	3
Podsumowanie	4
Zadanie 2 – Skomplikowana aplikacja użytkowa z dostępem do bazy danych.....	5
Chat GPT	5
Output	5
Rozwiązanie.....	6
Gemini	6
Output	6
Rozwiązanie.....	6
Podsumowanie	6
Uwagi.....	6

Zadanie 1 – Prosta witryna w przeglądarce

Prompt: Make a web application that displays a random image from the internet in the very middle of the page. The image should randomize upon every refresh of the page.

Chat GPT

Output



You

Make a web application that displays a random image from the internet in the very middle of the page. The image should randomize upon every refresh of the page.

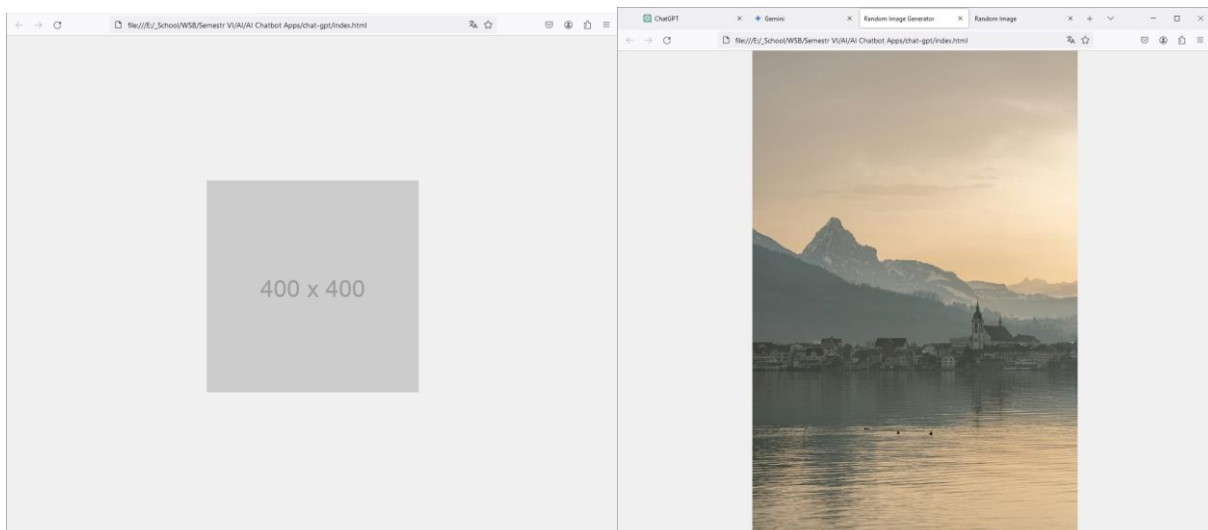


ChatGPT

Sure! Here's a simple HTML/CSS/JavaScript code snippet to create a web application that displays a random image from the internet in the center of the page, with the image randomizing upon every refresh:

```
html
Copy code

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Random Image Generator</title>
  <style>
    body, html {
      height: 100%;
      margin: 0;
      display: flex;
      justify-content: center;
      align-items: center;
      background-color: #f0f0f0;
    }
    #randomImage {
      max-width: 100%;
      max-height: 100%;
```




Rozwiązanie



Chat GPT stworzył działającą witrynę w pojedynczym pliku HTML, w którym zawarł minimalne style oraz kod JavaScript wymagany do losowania obrazów. Aplikacja domyślnie nie spełnia wymagań, ponieważ zostawia obrazy typu placeholder, które programista musi usunąć manualnie. Jest to rozwiązanie wymagające korekty, ale z przewidzianym miejscem na rozszerzenie funkcjonalności o dodatkowe repozytoria obrazów.

```
// Array of image URLs
var images = [
  "https://via.placeholder.com/400", // Placeholder image URL
  "https://source.unsplash.com/random",
  "https://placeimg.com/400/400/any",
  // Add more image URLs here
];
```

Gemini

Output


 Make a web application that displays a random image from the internet in the very middle of the page. The image should randomize upon every refresh of the page.

Show drafts  

◆ Here's a basic HTML and JavaScript code for a web application that displays a random image from the internet in the center of the page, refreshing on every page load:

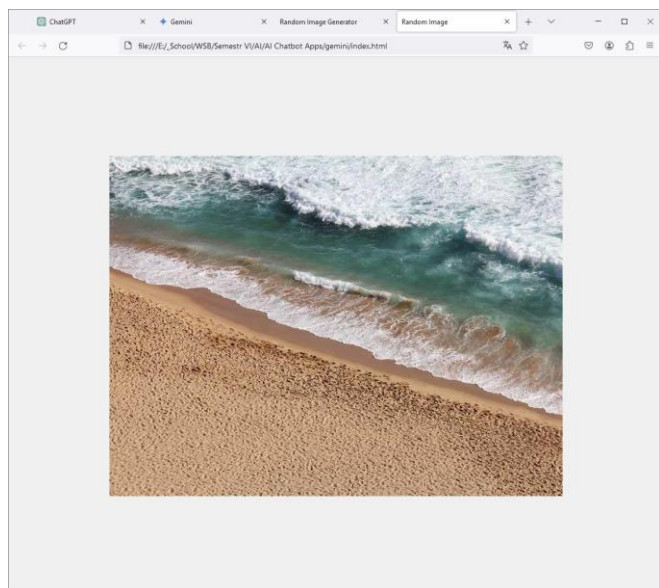
index.html:

```
HTML
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Random Image</title>
  <style>
    body {
      margin: 0;
      padding: 0;
      display: flex;
      justify-content: center;
      align-items: center;
      min-height: 100vh;
      background-color: #f0f0f0;
    }
    img {
      max-width: 80%;
      max-height: 80%;
    }
  </style>
</head>
<body>
  <img id="randomImage" src="" alt="Random Image">
  <script src="script.js"></script>
</body>
</html>
```

Use code [with caution](#). 

script.js:

```
JavaScript
const randomImage = document.getElementById('randomImage');
```



Rozwiązanie

Gemini stworzył witrynę w dwóch plikach: pliku HTML z zawartymi stylami oraz w pliku JavaScript z kodem wymagany do losowania obrazów. Domyślna implementacja aplikacji spełnia wymagania, ale nie pozostawia miejsca na dodanie innych repozytoriów obrazów. Dodatkowo, zamiast zagnieźdźać standardowo obraz ze źródła przez adres URL, Gemini wykorzystuje bardziej skomplikowany system pobierania, weryfikowania i zwracania adresów URL przed przypisywaniem ich do obiektu w przeglądarce.

Podsumowanie

Oba modele poradziły sobie z prostym zadaniem. Gemini zwróciło produkt działający od razu w końcowym stanie, Chat GPT uwzględnił chęć modyfikacji kodu przez programistę. Oba modele wykorzystały istniejące rozwiązania zwracające losowe obrazy i oba obudowały je w nadmierny sposób. Kod aplikacji mógłby być prostszy w obu przypadkach, jednakże formatowanie kodu z podziałem na pliki od Gemini to lepsza praktyka deweloperska. Mile widziane byłoby wydzielenie kodu CSS do oddzielnego pliku, ale nie było to uwzględnione w żadnym modelu.

Zadanie 2 – Skomplikowana aplikacja użytkowa z dostępem do bazy danych.

Prompt: Make a web application for managing invoices. The app should be able to store invoices that haven't been paid for yet. The app should make distinction between cyclically recurring payments and invoices that need to be paid for only once. The app should show reminders for the upcoming payments. The app should contain some example invoices with fictional data in both categories. The data of the invoices should be saved in the user's local storage in the browser and should persist between sessions. The user should be able to add their invoice information through the web view in an interactive form. The user should be able to click a button on an invoice to "pay" for the invoice and remove it from the list if it's a single payment invoice or advance the invoice into the next cycle if it's a recurring payment. The date of payment of the invoice should be formatted to be easily readable by a user, preferably in a DD-MM-YYYY format.

Chat GPT

Output

ChatGPT

Gemini

Invoice Manager

Invoice Manager

file:///E:/_School/WSB/Semestr VI/AI/AI Chatbot Apps/

Invoice Manager

Netflix
Amount: \$19.9
Due Date: 2024-08-26
Pay

Zakupy
Amount: \$141.13
Due Date: 2024-04-22
Pay

Add Invoice
Title:
Amount:
Due Date:
Recurrence (optional):

None

Add Invoice

You

Make a web application for managing invoices. The app should be able to store invoices that haven't been paid for yet. The app should make distinction between cyclically recurring payments and invoices that need to be paid for only once. The app should show reminders for the upcoming payments. The app should contain some dummy invoices with fictional data in both categories. The data of the invoices should be saved in the user's local store in the browser and should persist between sessions. The user should be able to add their invoice information through the web view. The user should be able to click a button on an invoice to "pay" for the invoice and remove it from the list if it's a single payment invoice or advance the invoice into the next cycle if it's a recurring payment.

ChatGPT

To create a web application with the described functionalities, we'll use HTML, CSS, and JavaScript along with the browser's localStorage to store invoice data locally. We'll implement features for adding, viewing, paying, and managing invoices. Here's the code:

html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Invoice Manager</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 20px;
    }
    h1 {
      text-align: center;
    }
    .invoice-container {
      display: flex;
      justify-content: center;
      align-items: center;
      flex-wrap: wrap
    }
  </style>
</head>
<body>
  <h1>Invoice Manager</h1>
  <div class="invoice-container">
    <div>
      <b>Netflix</b>
      <br>Amount: $19.9
      <br>Due Date: 2024-08-26
      <br><button>Pay</button>
    </div>
    <div>
      <b>Zakupy</b>
      <br>Amount: $141.13
      <br>Due Date: 2024-04-22
      <br><button>Pay</button>
    </div>
  </div>
  <div>
    <b>Add Invoice</b>
    <br>Title: <input type="text">
    <br>Amount: <input type="text">
    <br>Due Date: <input type="text">
    <br>Recurrence (optional):
    <br><div>None</div>
    <br><button>Add Invoice</button>
  </div>
</body>
</html>
```

ChatGPT

Gemini

Invoice Manager

Invoice Manager

file:///E:/_School/WSB/Semestr VI/AI/AI Chatbot Apps/

Invoice Manager

Web Hosting
Amount: \$50
Due Date: 2024-04-30
Pay

Subscription Service
Amount: \$20
Due Date: 02-07-2024
Pay

Membership Fee
Amount: \$100
Due Date: 2024-05-10
Pay

Add Invoice
Title:
Amount:
Due Date:
Recurrence (optional):

None

Add Invoice

Rozwiązanie

Chat GPT zwrócił funkcjonalny kod przy pierwszym podejściu, podejście kolejne z poprawionym zapytaniem zwróciło kod w pełni spójny z wymaganiami. Ponownie, Chat GPT zwraca kod w jednym pliku, niezależnie od jego długości. Style CSS zaimplementowane zostały szczątkowo, ale w estetyczny sposób. Z drobnymi zmianami byłaby to aplikacja gotowa do użytku przez niedoświadczonego użytkownika.

Gemini

Output

Make a web application for managing invoices. The app should be able to store invoices that haven't been paid for yet. The app should make distinction between cyclically recurring payments and invoices that need to be paid for only once. The app should show reminders for the upcoming payments. The app should contain some dummy invoices with fictional data in both categories. ...

Here's a basic structure for a web application managing invoices with the features you described. This is a simplified version without a full-fledged user interface, but it demonstrates the core functionalities.

index.html:

```
HTML<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Invoice Manager</title>
  <style>
    /* Add basic styling for the invoice list and buttons */
  </style>
</head>
<body>
  <h1>Invoice Manager</h1>
  <ul id="invoiceList"></ul>
  <button id="addInvoiceBtn">Add Invoice</button>
  <script src="script.js"></script>
</body>
</html>
```

script.js:

```
JavaScript
const invoiceList = document.getElementById('invoiceList');
```

Invoice Manager

- Electricity Bill - 100 - Due: 2024-05-19T22:00:00.000Z (Recurring) Pay & Advance
- Internet Service - 50 - Due: 2024-05-14T22:00:00.000Z (Recurring) Pay & Advance
- Book Purchase - 20 - Due: 2024-05-24T22:00:00.000Z (One-Time) Pay
- undefined - undefined - Due: undefined (One-Time) Pay
- undefined - undefined - Due: undefined (One-Time) Pay
- undefined - undefined - Due: undefined (One-Time) Pay

Add Invoice

Rozwiązanie

Kod zwrócony przez model Gemini, pomimo kilku podejść, nigdy nie był funkcjonalny. Pierwsze podejście miało brakujące znaki na końcu pliku, bez których strona funkcjonowała jeszcze gorzej. W pierwszym podejściu możliwe było dodawanie pustych faktur, w następnych został dodany formularz który nie przekazywał swoich wartości do kodu. Pomimo lepszych praktyk dzielenia kodu na pliki, nie można uznać żadnego rozwiązania za nawet częściowo poprawne.

Podsumowanie

Model GPT jest znacznie lepiej przetrenowany pod kątem pisania kodu użytkowego w technologiach web. Model rozumie skomplikowane polecenia lepiej i formatuje je w przystępny dla użytkownika sposób, przynajmniej w porównaniu z Gemini. Drugi model, pomimo ewidentnych podwalin pod lepszy sposób przekazywania kodu w kilku plikach, nie rozumie logiki jego funkcjonowania w dostateczny sposób, żeby móc wykonywać jakkolwiek skomplikowane polecenia.

Uwagi

Kod źródłowy zwrócony przez chatboty jest dostępny w katalogach równoległych z tym sprawozdaniem. Zapytanie zostały skomponowane w języku angielskim żeby wyeliminować nieporozumienia wynikające z tłumaczenia maszynowego.