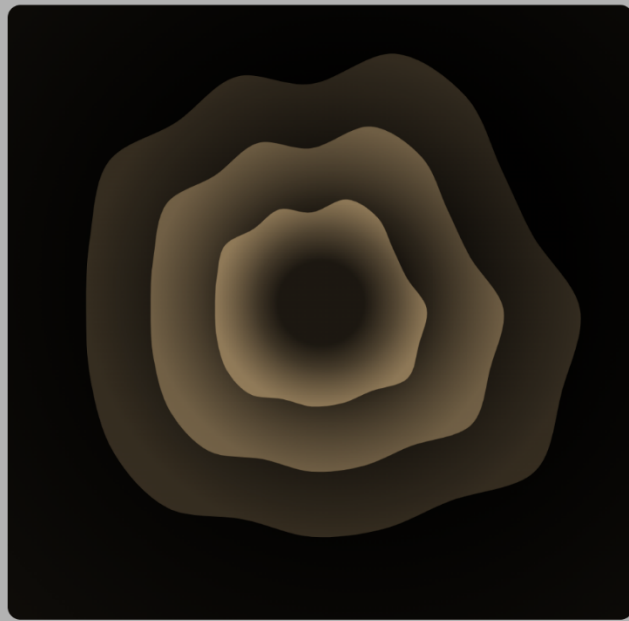


SPOTIFY



Studio Web & Mobile 1. H24

Bucher Lukas, Giaimo Laura,
Hirschi Marin, Lisa Landolt

Inhaltsverzeichnis

Unser Team	3
Konzept.....	3
Kurzbeschrieb	3
Fragestellung	3
Idee	3
Zielgruppe	3
Testings.....	4
Unit-Testing Protokoll	4
Cypress End-to-end Testing	5
Performance Testing.....	6
Deklaration vom Geistigem Eigentum	7

Unser Team

Bucher Lukas: Frontend + Backend Developer

Giaimo Laura: Designer

Hirschi Marin: Frontend + Backend Developer

Landolt Lisa: Frontend Developer + Testings

Konzept

Kurzbeschreibung

In diesem Modul entwickelten wir als Gruppe eine Webapplikation, die auf APIs (Spotify und Google Gemini) basiert. Das Ziel war es, eine funktionale und kreative Lösung zu entwickeln, die Daten aus einer ausgewählten API holt und einen erkennbaren Mehrwert für die Nutzer bietet.

Fragestellung

Wie kann die Visualisierung des Musikgeschmacks in Form einer interaktiven Zeitreise das persönliche Musikerlebnis vertiefen?

Idee

Im Fokus steht die „Music Journey“ – eine Timeline, die die letzten Jahre des persönlichen Musikgeschmacks aufzeichnet. Die Reise beginnt in der Gegenwart und ermöglicht es den Nutzern, in die Vergangenheit zu scrollen, um frühere Phasen ihres Musikgeschmacks zu erkunden. Dabei können sie Zeitabschnitte wie die letzten 4 Wochen, die letzten 6 Monate, das vergangene Jahr sowie die letzten drei Jahre (2023–2021) anhand der Wrapped-Playlists einsehen.

Die Musikreise wird grafisch durch eine dynamische Bubble visualisiert, die sich je nach Song der Nutzer verändert. Zusätzlich werden die Top-Songs, Top-Künstler und Top-Genres der jeweiligen Zeitperiode angezeigt. Nach den visualisierten Jahren folgen gesammelte Badges, die auf der gehörten Musik basieren.

Am Ende der Reise gibt es sowohl einen Share- als auch einen Download-Button. Damit können Nutzer ein oder mehrere Bilder ihrer gesamten „Music Journey“ erstellen und in sozialen Medien teilen.

Zielgruppe

Unsere Zielgruppe sind Premium-Spotify-Nutzer, insbesondere jüngere Generationen, die gerne tiefer in ihre Musikhistorie eintauchen, sowie langjährige Spotify-Nutzerinnen und -Nutzer, die ein Interesse an Musikvisualisierungen haben.


Testings

Unit-Testing Protokoll


Beschreibung:

Testet die Funktion `getArtist`. Die Tests decken verschiedene Szenarien ab, einschliesslich fehlender Zugriffstokens, fehlgeschlagener Abrufe, ungültiger Antworten, erfolgreicher Abrufe und fehlender IDs.


Test: should return null if access token is missing

- **Status:**  Bestanden
- **Details:** Stellt sicher, dass die Funktion null zurückgibt und eine Fehlermeldung ausgibt, wenn das Zugriffstoken nicht in `localStorage` gefunden wird.
- **Überprüfte Logs:** Access token is undefined


Test: should return null if fetch fails

- **Status:**  Bestanden
- **Details:** Simuliert einen Netzwerkfehler während des Abrufs und prüft, ob die Funktion null zurückgibt und eine entsprechende Fehlermeldung ausgibt.
- **Überprüfte Logs:** Error fetching Artist:


Test: should return null if response is not valid

- **Status:**  Bestanden
- **Details:** Simuliert eine ungültige Antwort von der Spotify API (z. B. HTTP-Status ok: false) und stellt sicher, dass die Funktion null zurückgibt und eine Warnung ausgibt.
- **Überprüfte Logs:** Artist with ID id not found.

Test: should return artist data if fetch is successful

- **Status:**  Bestanden
- **Details:** Simuliert einen erfolgreichen Abruf von Künstlerinformationen. Prüft, ob die Funktion die richtigen Daten (`mockArtist`) zurückgibt.
- **Erwartetes Ergebnis:** { id: 'id', name: 'Mock Artist' }

Test: should return null if no id is provided, even with a valid token

- **Status:**  Bestanden
- **Details:** Prüft, ob die Funktion null zurückgibt und eine Fehlermeldung ausgibt, wenn keine ID übergeben wird, obwohl ein gültiges Zugriffstoken vorhanden ist.
- **Überprüfte Logs:** Error: Artist ID is required

```

DEV v2.1.8 C:/Spotify/sputify/frontend

✓ src/tests/AccessTokenGetArtistTest.spec.js (5)
✓ getArtist (5)
  ✓ should return null if access token is missing
  ✓ should return null if fetch fails
  ✓ should return null if response is not valid
  ✓ should return artist data if fetch is successful
  ✓ should return null if no id is provided, even with a valid token

Test Files 1 passed (1)
Tests 5 passed (5)
Start at 13:03:17
Duration 563ms (transform 33ms, setup 0ms, collect 50ms, tests 6ms, environm
ent 0ms, prepare 259ms)

```

Figure 1 Screenshot von allen bestandenen Unit-Tests

Cypress End-to-end Testing

Test1: Responsiveness Test für Spotify-Login-Seite

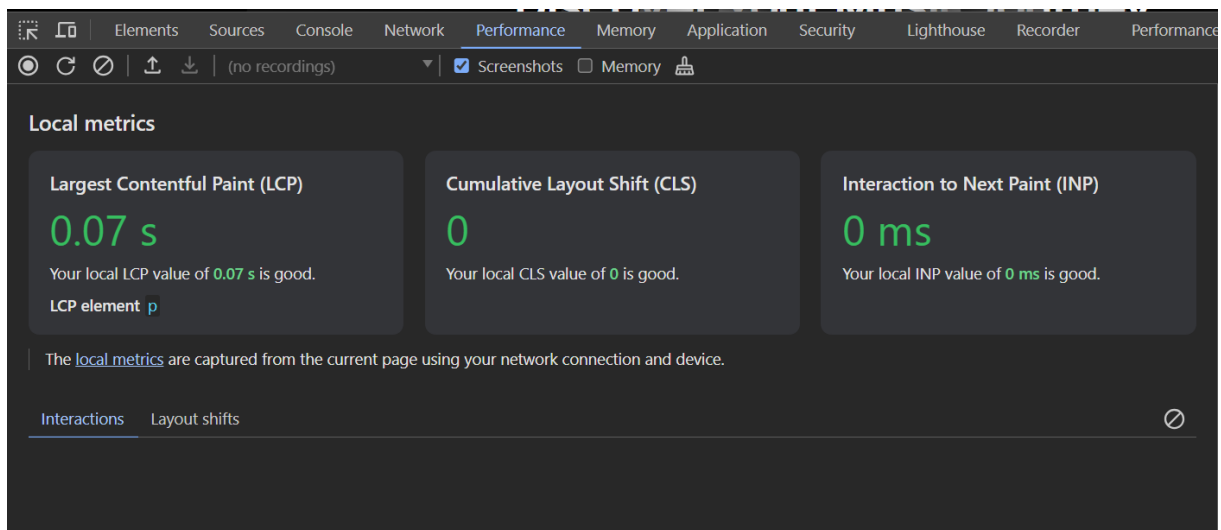
Überprüfung, ob die Spotify-Login-Seite auf verschiedenen Bildschirmgrößen korrekt dargestellt wird und wichtige Elemente sichtbar sind.

Testfall	Viewport	Status	Details
Grosser Bildschirm	1280x800	Bestanden	Alle Elemente (logo, button, intro) sichtbar.
Tablet-Bildschirm	768x1024	Bestanden	Alle Elemente sichtbar und korrekt angepasst.
Mobiler Bildschirm	375x667	Bestanden	Alle Elemente sichtbar, Layout für Mobilgeräte optimiert.
Kleiner Bildschirm (Layout-Anpassung)	320x480	Bestanden	Alle Elemente sichtbar, Layout passt sich kleinen Bildschirmen an.

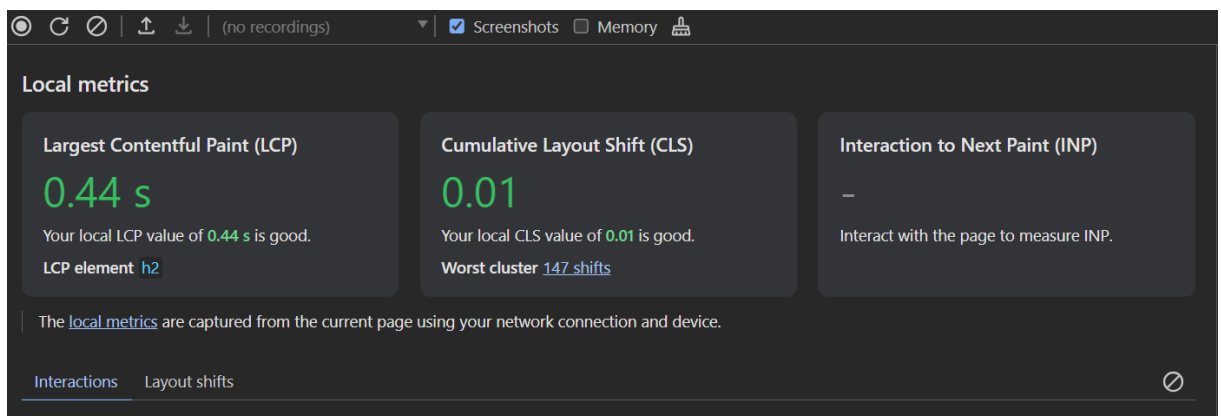
Test2: Login

Der Test überprüft den Spotify-Login-Flow. Zunächst wird die Startseite aufgerufen und sichergestellt, dass der Login-Button sichtbar und klickbar ist. Nach dem Klick auf den Login-Button wird die Spotify-Anmeldeseite geladen, auf der die Eingabe von Benutzername und Passwort überprüft wird. Anschließend wird getestet, ob der Login-Button korrekt geklickt werden kann. Ein bekannter Fehler besteht darin, dass Cypress keine doppelten Weiterleitungen innerhalb eines Tests unterstützt, weshalb die Weiterleitung zur Journey-Seite in diesem Test nicht geprüft werden konnte.

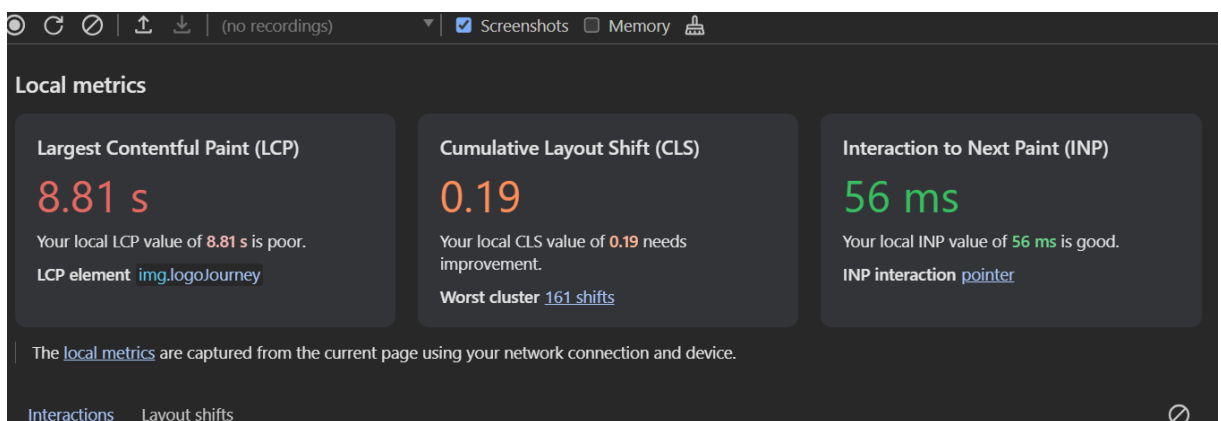
Performance Testing



Vor dem Login



Während dem Login



Nach dem Login

Ein guter LCP-Wert sollte unter 2,5 Sekunden liegen. Der Wert von 8,81 Sekunden ist schlecht und weist auf langsame Ladezeiten hin. Ein guter CLS-Wert sollte unter 0,1 liegen. Der Wert von 0,19 ist verbesserungswürdig, da Layout-Verschiebungen zu einem schlechten Nutzererlebnis führen können. Da wir zuerst die Spotify Daten laden mussten wir einen Ladeanimation einbauen. Die Musik wird zuerst abgespielt und danach visualisiert.

Deklaration vom Geistigem Eigentum

Der Text sowie der Code für diese Website wurden von uns selbst erstellt. Für die Funktionalität nutzen wir die Spotify API und die Google Gemini API. Zusätzlich wurde ChatGPT als unterstützendes Tool eingesetzt, um bestimmte Code Herausforderungen zu bewältigen.