Music tracker dokumentation

## Music Tracker

### Version 1.0.0, Datum: 5. Juni 2023

Hier kannst du einen Printscreen deiner App einfügen.

Ein Bild, das Text, Screenshot, Handy, Multimedia enthält.

Automatisch generierte BeschreibungEin Bild, das Text, Screenshot, Multimedia, Handy enthält.

Automatisch generierte BeschreibungEin Bild, das Screenshot, Text, Multimedia, Software enthält.

Automatisch generierte Beschreibung

Abstract (Kurzbeschreibung)

Music Tracker ist eine App, die Nutzern ermöglicht, ihre Lieblingsmusik einfach zu finden und zu geniessen. Mit der Suchfunktion können Benutzer mühelos nach Songs, Alben und Künstlern suchen. Die App bietet auch die Möglichkeit, eine Vorschau der gefundenen Musikstücke abzuspielen, um einen ersten Eindruck zu gewinnen. Besonders praktisch ist die Favoritenseite, auf der Nutzer ihre Lieblingslieder speichern und schnell wiederfinden können. Music Tracker kombiniert somit die Funktionen einer Musikdatenbank und eines persönlichen Musikarchivs in einer benutzerfreundlichen Anwendung.

user stories

An dieser Stelle alle Anforderungen als User Stories auflisten:

* Als Benutzer möchte ich bei einem gewünschten Lied auf «Play» drücken können damit ich das Lied anhören kann.
* Als Benutzer möchte ich bei einem gewünschten Lied auf Favoriten drücken damit ich das Lied in eine Favoriten Liste anschauen kann.
* Als Benutzer möchte ich bei eine Search Eingabe nach Liede Suchen damit ich das Lied finden kann.
* Als Benutzer möchte ich eine freundliche Benutzeroberfläche haben, um die App einfach bedienen zu können.
* Als Benutzer möchte ich einen Homescreen haben, auf dem ich bereits vorhandene Lieder sehen kann, ohne danach suchen zu müssen.

Mockups

An dieser Stelle den Prototypen deiner App beschreiben. Füge einen Printscreen deiner Screens ein und beschreibe jeden Screen einzeln.

Ein Bild, das Text, Screenshot, Zahl, Diagramm enthält.

Automatisch generierte Beschreibung

Technische Realisierung

Die Music Tracker App wurde mit React Native entwickelt, um sowohl Android- als auch iOS-Smartphones zu unterstützen, wobei bisher nur die Android-Version getestet wurde. Sie greift auf die iTunes Search API zurück, um alle relevanten Daten wie Titel, Albumcover und Musikpreviews abzurufen. Diese API-Anfragen werden während der Suche nach Liedern in der App gestellt, um die entsprechenden Informationen dynamisch zu laden.

* **Start Seite (Home) ------------ API**

Auf der Startseite der App wird eine zufällige Nummer generiert, die als ID für die Funktionen "fetchSongsById" und "fetchAlbumsById" dient. Dadurch werden dem Nutzer zufällige Lieder und Alben präsentiert, um die Vielfalt der Musik zu zeigen.

* **Suche ------------ API**

Für die Suche nach Liedern verwendet die App einen vom Nutzer eingegebenen Suchbegriff ("Search term"), um eine Liste von Liedern mit ähnlichem Namen über eine API-Anfrage abzurufen.

* **Navigation**

Die Navigation innerhalb der App sowie das Abspielen von Musikpreviews erfolgen über Expo. Insbesondere wird das Paket "expo-av" verwendet, das durch den Befehl "expo install expo-av" installiert werden kann. Dieses Paket bietet eine einfache Integration von Audio-Player-Funktionen, die essentiell für das Abspielen der Musikvorschauen sind.

* **Speichern (Favoriten) ------------ Async**

Zur Speicherung von Favoriten wird Async Storage genutzt. Diese Methode erlaubt es, JSON-Daten der favorisierten Lieder lokal zu speichern und zu verwalten. Ein zusätzliches Attribut wird hinzugefügt, das anzeigt, ob ein Lied als Favorit markiert wurde (true oder false). Der Prozess des Favorisierens eines Liedes funktioniert wie folgt: Wenn ein Benutzer ein Lied als Favorit markiert, wird das JSON-Objekt des Liedes lokal gespeichert und das Favoriten-Attribut entsprechend gesetzt. Bei späteren Suchen wird überprüft, ob das Lied bereits als Favorit gespeichert wurde, damit es in den Suchergebnissen entsprechend markiert und angezeigt werden kann.

Async Storage ist technisch gesehen nicht die beste Wahl für ein Favoriten-Feature in einer Musik-App aus mehreren Gründen. Die Speichermenge ist begrenzt und es ermöglicht nur die lokale Speicherung, ohne Unterstützung für die Nutzung auf mehreren Geräten. Wir haben uns dennoch dafür entschieden, weil wir bisher keine Erfahrung damit hatten und diese Gelegenheit nutzen wollten, um neue Techniken kennenzulernen.

Testing

In diesem Kapitel definiert ihr die manuellen Tests, die durchgeführt werden. Es müssen mind. 4 manuelle Tests definiert werden.

Hier ein Beispiel eines Tests:

|  |  |
| --- | --- |
| **Abschnitt** | **Inhalt** |
| ID | ST-01 |
| Anforderungen | Als Benutzer möchte ich bei einem gewünschten Lied auf «Play» drücken können damit ich das Lied anhören kann. |
| Vorbedingungen | Der Emulator muss mit der aktuellsten Version des Master-Branches am Laufen sein. |
| Ablauf | Der Benutzer soll von jeder Seite aus auf den Play-Button eines Liedes klicken können, um das Lied mit dem Media Player abzuspielen. |
| Erwartetes Resultat | Das Lied, das ausgewählt wurde, sollte beginnen zu spielen. |

|  |  |
| --- | --- |
| ID | ST-02 |
| Anforderungen | Als Benutzer möchte ich einen Homescreen haben, auf dem ich bereits vorhandene Lieder sehen kann, ohne danach suchen zu müssen. |
| Vorbedingungen | Der Emulator muss mit der aktuellsten Version des Master-Branches am Laufen sein. |
| Ablauf | Der Benutzer soll beim aufstarten direkt auf eine home screen landen und ein paar zufällige Lieder bekommen. |
| Erwartetes Resultat | Es sollten ein paar zufällige Leider auftauchen. |

|  |  |
| --- | --- |
| ID | ST-03 |
| Anforderungen | Als Benutzer möchte ich bei eine Search Eingabe nach Liede Suchen damit ich das Lied finden kann. |
| Vorbedingungen | Der Emulator muss mit der aktuellsten Version des Master-Branches am Laufen sein. |
| Ablauf | Benützer soll auf die Such Seite wechseln und dort ein Lied suchen. |
| Erwartetes Resultat | Das gesuchte Leid soll angezeigt werden. |

|  |  |
| --- | --- |
| ID | ST-04 |
| Anforderungen | Als Benutzer möchte ich bei einem gewünschten Lied auf Favoriten drücken damit ich das Lied in eine Favoriten Liste anschauen kann. |
| Vorbedingungen | Der Emulator muss mit der aktuellsten Version des Master-Branches am Laufen sein. |
| Ablauf | Benutzer soll auf das like Button von ein Lied klicken können und dann in der Favoriten seite das Leid sehen und auch unliken. |
| Erwartetes Resultat | Wie oben beschrieben soll die App Sachen Erfolg reichlich liken und Unliken können. |

|  |  |
| --- | --- |
| ID | ST-05 |
| Anforderungen | Als Benutzer möchte ich eine freundliche Benutzeroberfläche haben, um die App einfach bedienen zu können. |
| Vorbedingungen | Der Emulator muss mit der aktuellsten Version des Master-Branches am Laufen sein. |
| Ablauf | Vorherigen Test fälle machen. |
| Erwartetes Resultat | Benutzer soll mit dem Ui klar gekommen sein. |

Testprotokoll

In diesem Kapitel wird die Testausführung protokolliert. Welche Tests wurden wann von wem mit welchem Resultat durchgeführt.

Hier ein Beispiel eines Eintrages:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Person / Datum | Erfolgreich | Bemerkungen |
| ST-01 | D. Muster 18.06.2023 | Ja | Der Testfall war erfolgreich, der Testperson 1 ist jedoch aufgefallen, dass es in der angezeigten Fehlermeldung noch einen Rechtschreibfehler gibt. |
| ST-01 | Thuyan Satheeswaran  5.7.2024 | Ja |  |
| ST-02 | Thuyan Satheeswaran  5.7.2024 | Ja |  |
| ST-03 | Thuyan Satheeswaran  5.7.2024 | Ja |  |
| ST-04 | Thuyan Satheeswaran  5.7.2024 | Ja |  |
| ST-05 | Thuyan Satheeswaran  5.7.2024 | Ja | Suche Logo und Name anpassen. |

fazit

* **Was lief gut/schlecht?**

Wir haben die Mehrheit unserer gesetzten Ziele erreicht und dabei wertvolle Erfahrungen mit React Native und anderen Paketen wie Expo gesammelt.

* **Wie seid ihr mit dem Endergebnis zufrieden?**

Obwohl die App grundsätzlich funktioniert, sind wir nicht ganz zufrieden damit. Dies liegt an mehreren Punkten:

* + Fehlende Album- und Playlist-Funktionalitäten: Die App kann derzeit nur Alben anzeigen und hat keine erweiterten Funktionen für Alben und Playlists.
  + Probleme mit dem AudioPlayer-Komponenten: Die Komponente "AudioPlayer.js" wurde sehr schnell implementiert und funktioniert nicht eigenständig als Komponente. Sie muss in eine andere Komponente wie "SongList.js" integriert werden. Dies kostet nicht nur Zeit, sondern führt auch zu möglichen Inkonsistenzen im Design des Players und besiegt den Zweck einer separaten Komponente, wenn sie in andere Komponenten eingebunden und angepasst werden muss.
  + App-Idee: Die Grundidee der App ist nicht sehr überzeugend, da sie kein Problem löst, das nicht bereits von einer anderen App besser gelöst wurde. Obwohl wir verstehen, dass dies eine Übung sein sollte, ist es uns dennoch wichtig, gute Ideen zu entwickeln, auch wenn sie nicht vollständig umgesetzt werden.
* **Was habt ihr gelernt?**

Bis jetzt haben wir nur mit .Net Maui genützt, um für Smartphones zu entwickeln. Obwohl .Net Maui ein sehr gutes und nützliches Framework ist, hat es auch seine Probleme. Dank diesem Modul konnten wir einen guten jedoch kurzen Einblick in eine andere Option für Mobile Development. Wir haben die Viele vorteile an React Native sehen können.

* **Rückblick**

Insgesamt war die Erfahrung gut. Unsere einzige Beschwerde ist die kurze Entwicklungszeit, die keine grösseren Kurskorrekturen erlaubt und allgemein sehr knapp bemessen ist für die Entwicklung.