Sprint 4: Reporting-Dashboard für Studiobetreiber – Nutzungshäufigkeit der Geräte

Gym Progress Tracking App

February 8, 2025

Einleitung

In diesem Sprint wird ein Reporting-Dashboard für Studiobetreiber entwickelt, das die Nutzungshäufigkeit der Trainingsgeräte anzeigt. Konkret soll für jedes Gerät aggregiert werden, wie viele Trainingseinheiten insgesamt von allen Nutzern durchgeführt wurden. Dies dient als Grundlage für zukünftige Erweiterungen, wie z. B. Leaderboards oder detailliertere Analysen.

1 Backend: Aggregation der Nutzungshäufigkeit

1. Datenbankabfrage entwickeln

Die erforderlichen Daten stammen aus der Tabelle `training_history`. Für die Nutzungshäufigkeit interessiert uns vor allem die Anzahl der Trainingseinheiten pro Gerät. Eine geeignete SQL-Abfrage könnte wie folgt aussehen:

```
SELECT device_id, COUNT(*) AS usage_count
FROM training_history
GROUP BY device_id;
```

2. API-Endpunkt implementieren

Füge in deiner `server.js` einen neuen Endpunkt hinzu, der die aggregierten Daten liefert:

```
app.get('/api/reporting/usage', async (req, res) => {
  try {
    const result = await pool.query(
      'SELECT device_id, COUNT(*) AS usage_count FROM training_history GROUP BY devic
    );
    res.json({ message: 'Nutzungshäufigkeit erfolgreich abgerufen', data: result.rows
  } catch (error) {
    console.error('Fehler beim Abrufen der Nutzungshäufigkeit:', error.message);
    res.status(500).json({ error: 'Serverfehler beim Abrufen der Nutzungshäufigkeit'
  }
});
```

Stelle sicher, dass dieser Endpunkt **vor** deiner Catch-All-Route definiert ist.

2 Frontend: Visualisierung des Reporting-Dashboards

1. Auswahl einer Visualisierungsbibliothek

Für eine schnelle und anschauliche Darstellung eignet sich **Chart.js** in Kombination mit dem React-Wrapper `react-chartjs-2`. Alternativ kann auch eine einfache Tabelle verwendet werden.

2. Installation der Bibliotheken (optional)

Führe folgenden Befehl im Frontend-Verzeichnis aus:

```
npm install chart.js react-chartjs-2
```

3. Erstellung der Reporting-Komponente

Erstelle eine neue Komponente namens `ReportingDashboard.js` im `src`-Ordner:

```
import React, { useState, useEffect } from 'react';
import { Bar } from 'react-chartjs-2';
import { API_URL } from './config';

function ReportingDashboard() {
  const [reportData, setReportData] = useState([]);

  useEffect(() => {
    async function fetchReportData() {
      try {
        const response = await fetch(`${API_URL}/api/reporting/usage`);
        const result = await response.json();
        if (response.ok && result.data) {
          setReportData(result.data);
        } else {
          console.error(result.error);
        }
      } catch (error) {
        console.error('Fehler beim Abrufen der Nutzungshäufigkeit:', error);
      }
    }
    fetchReportData();
  }, []);

  // Aufbereitung der Daten für ein Balkendiagramm
  const chartData = {
    labels: reportData.map(item => `Gerät ${item.device_id}`),
    datasets: [
      {
        label: 'Anzahl Trainingseinheiten',
        data: reportData.map(item => item.usage_count),
```

```
        backgroundColor: 'rgba(75, 192, 192, 0.6)',
      }
    ]
  };

  return (
    <div>
      <h2>Reporting-Dashboard: Nutzungshäufigkeit</h2>
      <Bar data={chartData} />
      <h3>Detailübersicht</h3>
      <table border="1" cellPadding="5" style={{ margin: '0 auto', width: '90%' }}>
        <thead>
          <tr>
            <th>Gerät</th>
            <th>Nutzungshäufigkeit</th>
          </tr>
        </thead>
        <tbody>
          {reportData.map(item => (
            <tr key={item.device_id}>
              <td>Gerät {item.device_id}</td>
              <td>{item.usage_count}</td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  );
}

export default ReportingDashboard;
```

4. Integration in das Gesamtsystem

1. **Routing anpassen:**
   Öffne deine `App.js` und füge eine neue Route für das Reporting-Dashboard hinzu:

   ```
   <Route path="/reporting" element={<ReportingDashboard />} />
   ```

2. **Navigation erweitern:**
   Füge in der globalen Navigation einen Link zum Reporting-Dashboard hinzu, z. B.:

   ```
   <li>
     <Link to="/reporting">Reporting-Dashboard</Link>
   </li>
   ```

5. Testen und Validieren

1. Starte Backend und Frontend neu.
2. Rufe den neuen API-Endpunkt direkt (z. B. mit Postman) auf, um sicherzustellen, dass die aggregierten Daten (Geräte-Nutzungshäufigkeit) korrekt zurückgegeben werden.
3. Überprüfe im Frontend das Reporting-Dashboard:
    - Das Balkendiagramm sollte für jedes Gerät die Anzahl der Trainingseinheiten anzeigen.
    - Die Detailtabelle sollte die Geräte-IDs und die jeweilige Nutzungshäufigkeit anzeigen.
4. Passe bei Bedarf die Darstellung oder Filter-/Sortierfunktionen an.

Fazit

Mit dieser Anleitung implementierst du zunächst ein Reporting-Dashboard, das die Nutzungshäufigkeit der Geräte aggregiert anzeigt. Für jedes Gerät wird die Gesamtzahl der Trainingseinheiten (über alle Nutzer) ermittelt und in einem Balkendiagramm sowie in einer Detailtabelle dargestellt. Diese Basis lässt sich später erweitern, um weitere Metriken oder Filterfunktionen zu integrieren.