Frontend-Backend Integration Guide

Overview

This guide details how to modify the sports analytics application to:

1. POST sample data to the database via the backend
2. GET data from the database filtered by season
3. Dynamically populate the season dropdown from database values

Backend Changes

1. Update `backend/main.py`

```
from fastapi import FastAPI, Depends, HTTPException
from sqlalchemy.orm import Session
from sqlalchemy import distinct
from typing import List
from datetime import datetime
from database import get_db
from models import Game

# ... existing imports and setup ...

# New endpoint to get unique seasons
@app.get("/api/seasons")
def get_seasons(db: Session = Depends(get_db)):
    try:
        seasons = db.query(Game.season).distinct().order_by(Game.season.desc()).all()
        return {"seasons": [season[0] for season in seasons]}
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))

# New endpoint to get games by season
@app.get("/api/games/{season}")
def get_games_by_season(season: str, db: Session = Depends(get_db)):
    try:
        games = db.query(Game).filter(Game.season == season).all()
        return {
            "games": [{
                "id": game.id,
                "game_date": game.game_date,
                "attendance": game.attendance,
                "ticket_price": float(game.ticket_price),
                "day_of_week": game.day_of_week,
                "temperature": float(game.temperature),
                "precipitation": float(game.precipitation),
                "opponent": game.opponent,
                "promotion": game.promotion,
                "season": game.season
            } for game in games]
        }
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))

# New endpoint to add sample data
@app.post("/api/games/bulk")
def add_bulk_games(games: List[dict], db: Session = Depends(get_db)):
    try:
        for game_data in games:
            # Convert string date to datetime
            game_date = datetime.strptime(game_data["game_date"], "%Y-%m-%d").date()

            game = Game(
                game_date=game_date,
                attendance=game_data["attendance"],
                ticket_price=game_data["ticket_price"],
                day_of_week=game_data["day_of_week"],
                temperature=game_data["temperature"],
                precipitation=game_data["precipitation"],
                opponent=game_data["opponent"],
                promotion=game_data["promotion"],
                season=game_data["season"]
            )
            db.add(game)

        db.commit()
        return {"message": "Sample data added successfully"}
    except Exception as e:
        db.rollback()
        raise HTTPException(status_code=500, detail=str(e))
```

Frontend Changes

1. Create new API service file `src/services/api.js`

```
const API_URL = import.meta.env.VITE_API_URL;

export const api = {
  // Get all available seasons
  getSeasons: async () => {
    const response = await fetch(`${API_URL}/api/seasons`);
    if (!response.ok) throw new Error('Failed to fetch seasons');
    const data = await response.json();
    return data.seasons;
  },

  // Get games for a specific season
  getGamesBySeason: async (season) => {
    const response = await fetch(`${API_URL}/api/games/${season}`);
    if (!response.ok) throw new Error('Failed to fetch games');
    const data = await response.json();
    return data.games;
  },

  // Post sample data
  addSampleData: async (games) => {
    const response = await fetch(`${API_URL}/api/games/bulk`, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(games),
    });
    if (!response.ok) throw new Error('Failed to add sample data');
    return await response.json();
  },
};
```

2. Update `src/App.jsx`

```
import { useState, useEffect } from 'react';
import { LineChart, Line, XAxis, YAxis, CartesianGrid, Tooltip, Legend } from 'recharts';
import { api } from './services/api';
import './App.css';

function App() {
  const [seasons, setSeasons] = useState([]);
  const [selectedSeason, setSelectedSeason] = useState('');
  const [gameData, setGameData] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  // Sample data to be moved to database
  const sampleGames = [
    {
      "game_date": "2022-04-08",
      "attendance": 15789,
      "ticket_price": 25.00,
      "day_of_week": "Friday",
      "temperature": 62.5,
      "precipitation": 0.0,
      "opponent": "River Rats",
      "promotion": "Opening Day",
      "season": "2022"
    },
    // ... rest of your sample data ...
  ];

  // Load available seasons on component mount
  useEffect(() => {
    fetchSeasons();
  }, []);

  // Fetch games when season changes
  useEffect(() => {
    if (selectedSeason) {
      fetchGameData(selectedSeason);
    }
  }, [selectedSeason]);

  const fetchSeasons = async () => {
    try {
      const availableSeasons = await api.getSeasons();
      setSeasons(availableSeasons);
      if (availableSeasons.length > 0) {
        setSelectedSeason(availableSeasons[0]);
      }
    } catch (err) {
      setError('Failed to fetch seasons');
      console.error('Error fetching seasons:', err);
    }
  };

  const fetchGameData = async (season) => {
    setLoading(true);
    try {
      const games = await api.getGamesBySeason(season);
      const sortedGames = games.sort((a, b) =>
        new Date(a.game_date) - new Date(b.game_date)
      );
      setGameData(sortedGames);
      setError(null);
    } catch (err) {
      setError('Failed to fetch game data');
      console.error('Error fetching game data:', err);
    } finally {
      setLoading(false);
    }
  };

  const handleAddSampleData = async () => {
    try {
      await api.addSampleData(sampleGames);
      // Refresh the seasons list and current season data
      await fetchSeasons();
      if (selectedSeason) {
        await fetchGameData(selectedSeason);
      }
    } catch (err) {
      setError('Failed to add sample data');
      console.error('Error adding sample data:', err);
    }
  };

  return (
    <div className="container mx-auto p-4">
      <h1 className="text-3xl font-bold mb-4">Metro Miners Analytics Dashboard</h1>
```

```
{/* Add Sample Data Button */}
<button
  onClick={handleAddSampleData}
  className="mb-4 bg-blue-500 text-white px-4 py-2 rounded hover:bg-blue-600"
>
  Add Sample Data
</button>

{/* Season Selector */}
<div className="mb-4">
  <label htmlFor="season" className="mr-2">Select Season:</label>
  <select
    id="season"
    value={selectedSeason}
    onChange={(e) => setSelectedSeason(e.target.value)}
    className="border rounded p-1"
  >
    {seasons.map((season) => (
      <option key={season} value={season}>{season}</option>
    ))}
  </select>
</div>

{/* Error Display */}
{error && (
  <div className="text-red-500 mb-4">
    {error}
  </div>
)}

{/* Loading Display */}
{loading ? (
  <div className="text-gray-500">Loading...</div>
) : (
  /* Attendance Chart */
  <div className="w-full h-[400px]">
    <LineChart
      width={800}
      height={400}
      data={gameData}
      margin={{ top: 5, right: 30, left: 20, bottom: 5 }}
    >
      <CartesianGrid strokeDasharray="3 3" />
      <XAxis
```

```
                dataKey="game_date"
                tickFormatter={(date) => new Date(date).toLocaleDateString()}
              />
              <YAxis />
              <Tooltip
                labelFormatter={(date) => new Date(date).toLocaleDateString()}
              />
              <Legend />
              <Line
                type="monotone"
                dataKey="attendance"
                stroke="#8884d8"
                name="Attendance"
              />
            </LineChart>
          </div>
        )}
      </div>
    );
}

export default App;
```

Testing the Integration

1. Start the backend server:

```
cd backend
uvicorn main:app --reload
```

2. Start the frontend development server:

```
npm run dev
```

3. Test the functionality:
    - Click "Add Sample Data" to populate the database
    - Use the season dropdown to view different years
    - Verify that the chart updates with the correct data

Data Flow

1. Initial Load:

    - Frontend fetches available seasons from `/api/seasons`
    - Populates dropdown with seasons
    - Loads data for the first season

2. Adding Sample Data:

    - Click triggers POST request to `/api/games/bulk`
    - Backend adds records to database
    - Frontend refreshes seasons and current view

3. Changing Seasons:

    - Selection triggers GET request to `/api/games/{season}`
    - Backend filters database by season
    - Frontend updates chart with new data

Error Handling

The implementation includes:

- Loading states for async operations
- Error messages for failed requests
- Try-catch blocks for both frontend and backend
- Database transaction management
- Input validation

Next Steps

Consider adding:

1. Data validation on the frontend
2. Loading spinners for better UX
3. Success messages for operations
4. Pagination for large datasets
5. More detailed error messages
6. Data caching
7. Delete/Update functionality
8. More complex queries and filters