Database Homework 5

Name : Phuong Hoang Tran

Id: 010796179

Date: April 22, 2024

Deployed app url: https://db-hw-5-production.up.railway.app/

The Tech Stack

For this project, Javascript was chosen as the programming language due to its simplicity and versatility. The frontend framework for this project is React because of its continued support from Meta and its wide adoption in the industry and it was the easiest for me to implement. For the backend framework I choose Express.js along with Yup schema validation and Knex ORM because I'm most familiar with those technologies and modules. The relational database chosen was PostgreSQL because of its ability to return the newly created entry after insertion and its wide range of useful built in functions. To host the application on the web, I choose Railway.app because it's the cheapest option that offers a static ip address compared to spinning up a docker container on GCP or a linux box and getting a static IP address.

Database Design

There are three separate database tables: players, team, and games. I decided to use three separate tables because I prefer to strictly follow the rule of separation of concerns. This is not the most efficient way of implementing the database because computing the "conference win" and the "win" computed variables requires me to join multiple tables and format the data afterward. The data types for most fields are integer or string, except for the date field. Integer is used for anything related to ID or score or anything that can be added or incremented. Strings are used for names or special kinds of ids.

Error Checking

For error checking, instead of logging out the error when the user clicks submit. I choose to limit what the user can enter. To implement error checking in the backend I use database constraint along with Yup validation. Yup validates the input before it gets sent to the database using Knex ORM, if the input fails Yup validation, it will return a 400 status code along with the appropriate error message. The ORM filters out most of the malicious query before it gets sent to the PostgreSQL database. If something failed to get inserted into the database, it will simply return a 500 error message and a generic message.

Home page / Game page

Games

Please Select a Filter ▾

| 102 | ☆ | **Cobras**
Boston | V.S | **Dragons**
San Francisco | 97 |

4-14-2024

| 110 | ☆ | **Dragons**
San Francisco | V.S | **Sharks**
Miami | 104 |

4-16-2024

| 105 | ☆ | **Sharks**
Miami | V.S | **Cobras**
Boston | 100 |

4-18-2024

| 198 | ☆ | **Sharks**
Miami | V.S | **Eagles**
Philadelphia | 96 |

Action 1

Press F11 to exit full screen

125%   — +   Reset

Add a Game

team_id1: _____

team_id2: _____

score1: _____

score2: _____

date: 04/15/2024

Submit

Action 2

Press F11 to exit full screen

Add a Player

team_id: _____   Name: _____

Position:
Quarterback ▾

Submit

Action 3 and 4

Players

Please Select a Filter ▾

Please Select a Filter
Get all player from a certain team
Get all player for a certain position

sition: Quarterback

Name: Bob Williams   **Position:** Running Back

Name: Charlie Davis   **Position:** Wide Receiver

Name: David Garcia   **Position:** Tight End

Action 5

Teams

Please select a sorting by ⌄
Please select a sorting by
Sort By number of wins
Sort by the number of conference wins

nickname: Cobras   **conference:** Eastern   **division:** Atlantic   **total wins:** 3   **conference wins:** 1

nickname: Sharks   **conference:** Eastern   **division:** Southeast   **total wins:** 2   **conference wins:** 2

nickname: Eagles   **conference:** Eastern   **division:** Atlantic   **total wins:** 0   **conference wins:** 0

Action 6 and 7

Games

Please Select a Filter ⌄
Please Select a Filter
Get all games from a certain team
Get all games at a certain date

102   ☆   **Cobras**
Boston

V.S

Dragons
San Francisco