Sinatra React Project Application

Phase 3 of my software engineering course covered a new programming language, Ruby. We learnt how to use Ruby together with Active Record and Sinatra to create server-side applications. For our final project we were asked to build a web basic API with Sinatra and Active Record to support a React frontend with the following deliverables:

* Use Active Record to interact with a database
* Have at least two models with a one-to-many relationship
* At a minimum, set up the following API routes in Sinatra:
* create and read actions for both models

- full CRUD (create, read, update, delete) capability for one of the models

* Build a separate React frontend application that interacts with the API to perform CRUD actions
* Use good OO (object-orientated) design patterns
* Have separate classes for each of your models, and create instance and class methods as necessary

*Application idea*

One of the hardest parts of starting a project is coming up with an idea of what to build. Obviously, the deliverables need to be met, but I am limited in what I can build as I am still learning and the timeframes to deliver projects are quite tight. So again, I kept it simple and decided to create a ‘Workout List’ app which lists the names of well-known workouts together with brief notes on what to consider when doing the workouts, broken down in to six categories.

*Active Record*

Using Active Record I created two tables for the database, one for the categories and one for the exercise names and notes. The exercise table has a column for category ID which together with setting up Active Record associations links the two tables creating a one-to-many relationship. First two deliverables met.

*React*

I then went about setting up the React front-end. I needed to consider that I had to create components in a well thought out process that would display my data in a visually appealing way, fulfilling the read requirement of the CRUD actions. The hardest part with this was figuring out how to display the categories from the category table on the same line as the exercise name and note from the exercise table. I tried all sorts of ways to do this, but nothing worked so I did some googling and managed to find a solution, thanks to people who kindly share their code online. Such a life saver! To satisfy the create requirement I added a form so that users can add a new workout. For the delete requirement, I added a delete button to each individual line so a user can just click on this to delete a workout they don’t want in their list. For the update requirement, I added an edit button to each workout line, which allows the user to update the exercise notes as they wish. Deliverable four met.

*Sinatra*

In order for the CRUD actions, I set up in the React front-end, to work I needed to set up the Sinatra routes that would allow for these actions to be performed on the data in the database. To read the data, I wrote two separate GET routes, one for the category table and one for the exercise table. I also did the same for create and delete but instead of GET routes I wrote POST and DELETE routes. Seeing as a user can only edit the exercise notes, I only had to create one PATCH route for the exercise table. Third deliverable met.

*Bringing it all together*

In the end I was able to create the application, pictured below. Throughout the build, I made sure to structure my work using good OO design patterns, with separate classes for the models and creating instances and methods as needed. With deliverables five and six met, I was done with my project. Yay!

