Lauren Becker – CareDash Developer Take Home Test

**Used**

Language: Ruby on Rails and ActiveRecord

Database: MySQL

**Write-up**

Because Ruby, Rails, and MySQL are all self-taught languages/frameworks for me, I thought it would be easier to conceptualize a simple RESTful API in the form of a real application. My application is a VERY simplified version (with no CSS to make it look fancy) of a group-fitness instructor review app. In addition to requesting the objects in JSON format, I have also made the app so that you can view it normally in HTML to see its features as well. Users can add instructors to the database by inputting their name, class format, and description in a form. These parameters make up the parent-model. Later they can add reviews for each constructor with an integer rating (1-10) of their class experience and a comment explaining their rating. This is the child model. The instructor model has a has\_many relationship with the review model because one instructor can have many reviews. When a parent is requested in JSON, the parent objects’ children are nested as a JSON array. When a child object is requested, it includes its parent. The app has all of the CRUD functions (Create, Retrieve, Update, Delete) and accepts POST in both html and JSON form. Children are dependent, so deleting the parent of a child will also delete the child. GET endpoints for JSON format include viewing all instructors (with their reviews), viewing one instructor by id (with that instructor’s reviews), and viewing a review by id (with instructor id). The GET endpoints can also be accessed in html format by clicking the “Show” button by each instructor when viewing all instructors from localhost:3000/instructors

Some extra design features include validation to make sure instructor name entries are entered and are at least two characters long, and descriptions are less than 300 characters. Review ratings must be between 1-10 (and be entered) and have a comment less than 300 characters as well. If in HTML format, the user will be notified if and how they have failed to meet these requirements. There is also a dialogue box that comes up before the user can successfully delete an object, checking to make sure that they really want to delete the object.

An obvious design issue in this app is the fact that there is no authentication, so anyone can add/remove instructors and reviews as they please. The app should also check for duplicate entries before posting to the database. A scalability issue can arise if the instructor and review databases become huge and querying is slow as a result. The database will need to expand to accommodate larger quantities of data. In general Ruby on Rails and the RESTful API has good scalability compared to other methods of creating web apps.

**File summary**

**/app/models/instructor.rb** parent-model

**/app/models/review.rb**  child-model

**/app/controllers/instructors\_controller** CRUD methods for parent model

**/app/controllers/reviews\_controller** methods to create, show, and destroy for child model

see **/app/views/instructors** and **/app/views/reviews** for html format