Rails Intro Project

Goal

The goal of this project is to create a Ruby on Rails application using data collected from one or more data sources.

The first part involves creating the required database tables and pulling data from various data sources. The second part involves using Ruby on Rails to create reports, allow searching, and/or display visualizations based on this data.

The data you use can be pulled from open datasets or public and private APIs. Fake data generated by the Faker gem can also be used, as can data that you scrape from web pages. You may also need to research how best to import data from JSON, XML and CSV sources.

A list of potential sources for data can be found here.

The one dataset that is off-limits so the Board Game Geek dataset of boardgames, since that's what I used to build the sample project.

How You Obtain Marks For This Project

This document lists the possible features that can be included in the web application you are building. You do not have to implement all of these features.

Feature marks are defined according these badges:

🌟	**-3 marks if not implemented.**
🕯️	**3 marks for successful implementation.**
💡	**5 marks for successful implementation.**
🔥	**8 marks for successful implementation.**
BOOST	A 1.2 multiplier will be applied to marks received *before* midterms. ⬅

Your project will be graded out of a total of **100** marks.

You can make a copy of this marking spreadsheet to keep track of your marks.

When Projects Will be Marked

There will be four separate marking periods for this project:

- Our One Hour Class in the Week of October 12th to 16th
- Our Last Two Hour Class in the Week of October 12th to 16th
- Our One Hour Class in the Week of October 26th to 30th
- Our Last Two Hour Class in the Week of October 26th to 30th

Your project mark will be based *only* on marks you receive during in-class marking.

Marks received during the week of October 12th to 16th will receive a 1.2x bonus multiplier.

How Your Project Will be Marked

During an in-class marking session you will demonstrate your project's features to your instructor. For each feature demonstrated your instructor will determine if that feature will be marked as completed or not. It is your responsibility to come prepared to a marking session with a list of the features you wished to have marked. This list should include the feature number and the feature text from the list below. In order for a feature to be considered complete, you must have spent sufficient time and effort on its implementation. When in doubt, check with your instructor. You can rework features determined by your instructor to be incomplete and have them marked again if there are remaining marking periods.

With this marking process you are accumulating marks throughout the project, or in video game terms you are levelling up your mark. After any of the in-class marking sessions you will know your current project mark.

This project is worth 40% of your grade in this course. Time management will be an important factor in your grade for this project.

Remember: A minimum mark of 50% is required on this project in order to pass this course. This requirement is listed in the course outline.

Rails Intro Project Feature Rubric

1 - Gathering and Storing Data

🌟 🕯️ **1.1** - Select one or more data sources that you wish to pull data from. Write a short description of the data you are planning to use, how the data is structured, and the required database tables and columns you will require. This description should also explain how data you pull from different sources are related or how you plan to integrate the different datasets.

💡 **1.2** - Create a ERD diagram to document the schema of the database you plan on building for this project. This ERD should be based on your research from 1.1.

1.3 - Use Rails to generate the Active Record models and tables required to store the data from the data sources described above. This might include any required join tables, as well as tables for related fake data generated using the Faker gem.

- 🕯️ **1.3** Two tables are created and will be populated with data in 1.7.
- 💡 **1.3** Three tables are created and will be populated with data in 1.7.
- 🔥 **1.3** Four or more tables are created and will be populated with data in 1.7.

🌟💡 **1.4** - Add an association to your ActiveRecord models to define a one-to-many relationship. (This association must be used in 3.x or 4.x features.)

🔥 **1.5** Add an association to your models to define a many-to-many relationship through a join table. (This association must be used in 3.x or 4.x features.)

💡 **1.6** - Add two or more appropriate validations to all of your ActiveRecord models to ensure that the data you import into your tables will be valid.

1.7 - Write the seeds.rb script to pull in the data from your various sources into your database. You can use Faker as one of your data sources. Multiple endpoints from the same API are considered a single source. **(Minimum number of rows across all tables: 200)**

- 🕯️ **1.7** Data is pulled from a single data source.
- 💡 **1.7** Data is pulled from a two data sources.
- 🔥 **1.7** Data is pulled from from three or more data sources.

2 - Web Site Navigation

🕯️ **2.1** - Your web application has an about page that includes the details about your data sources from 1.1 and optionally your ERD from 1.2.

💡 **2.2 -** There is a menu present on all pages that includes links to two or more locations. (For example, a menu with a link to your homepage and your about page.)

3 - Data Navigation

🌟 🕯️ **3.1** - There exists a way for the user to navigate through the data you've collected at a high level. (For example, table or list of the entire data collection.)

💡 **3.2** - There exists a page for each of the individual entries of the data you have collected to display all the attributes of the collected data point.

💡 **3.3** - On the individual entry pages (3.2) data pulled from associated models is also present.

🔥 **3.4** - Data can be navigated by category or hierarchically based on a one-to-many or many-to-many relationship. (This means being able to navigate via a link back and forth between the "show" pages of your associations. For example, category "show" pages that link to the "show" page of all items in that category, with the "show" pages of each item linking back to the "show" page of the item's category.)

💡 **3.5** - Large collections of data are presented using pagination. (You can use the kaminari gem or a javascript library to implement this feature.)

🔥 **3.6** - Location data is present on a map (either one map for the entire collection or a separate map for each data point). The map can be provided using the Google Static Map API or Google Maps Javascript. (Google Maps API now requires a credit card to activate your API key. For low usage projects the API is still free. An alternative would be the Map Box Static Map API.)

4 - Searching and Filtering

🌟 🕯 **4.1** - Users can use a simple form to perform a text search through the available data.

💡 **4.2** - A user's search can be restricted to a specific category (or other one-to-many / many-to-many relationship) using a dropdown within the search form.

5 - Markup and Design

🌟 🕯 **5.1** - All HTML generated by your app validates as HTML5 with no errors.

🕯 **5.2** - At least one of your ERB views includes a conditional (if or unless).

💡 **5.3** - You built your markup and styling around the Bootstrap (or another similar) CSS framework . (At a minimum your layout should be built around your framework's grid system.)

6 - Source Control

🌟 🕯 **6.1** - You have configured git and github to keep your source under control with the ability to push from master to origin.

🕯 **6.2** - You've used git to keep your source under control *with a minimum of 20 commits.* Commits must be accompanied by reasonable commit messages.

Please see the next page for a list of common problems encountered by students on this project.

Appendix A

Common Problems and Troubleshooting Guide

Here is a list of common problems students run into with this project.

Seed Script Runs Without Error, But No Records Are Created

The first thing you should do is check to see if your model validations or associations are preventing your records from being saved.

Remember that you can check a model object for validation errors like this:

```
puts some_model_object.errors.message.inspect
```

My Dataset Has Its Own Existing Primary and Foreign Keys

It's not recommended to try to maintain the same primary and foreign keys as your dataset when importing your data. When looping through your dataset you can use the existing keys to find related data, but use Active Record to create new primary and foreign keys. If you find this confusing have a brainstorming chat with your instructor.