**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. This is a two part project.

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](https://github.com/stympy/faker) 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](https://gist.github.com/stungeye/afbf654810fe428832591358ecf591b1).

Two datasets / topics that are off-limits are the Board Game Geek dataset of boardgames (since that’s what was used to build [the sample project](https://board-game-freak.fly.dev/)) and anything movie-related (so as to not be too similar to Rails at the Movies demos).

# 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   
 *during the first marking week*. ⬅️

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.](https://docs.google.com/spreadsheets/d/1n5MU1NaUSkv-XEfgMjKhUqT0nT2i5km6HgvvsyedUro/edit#gid=757094693)

**NOTE:** The final section of this doc shows answers to common project problems.

# When Projects Will be Marked

There will be two separate marking periods for this project:

* **\*** Our Fall work week (no scheduled classes) October 10th to 14th
* Our Classes in the Week of October 17th to 21th

**Your project mark will be based *only* on marks you receive during one-on-one marking with your instructor.**

# 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 **30% of your grade** in this course. Time management will be an important factor in your grade for this project.

# 

# Rails Intro Project Feature Rubric

## 1 - Gathering and Storing Data

**🌟** 🕯️ **1.1 - Describe Datasets -** 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 - Database ERD -** Create an 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 - AR Models -** 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 t1.2 d and will be populated with data in 1.7.

**🌟**🕯️ **1.4** **- At least 1 model Association -** Add an association to your ActiveRecord models to define at least one one-to-many or many-to-many relationship. (*Associations must be used in 3.x or 4.x features.*)

**1.5 - Model Associations -** Add multiple ActiveRecord associations to your models. (*These associations must be used in 3.x or 4.x features.*)

🕯️ **1.5** Two of more one-to-many associations.

💡 **1.5** At least one one-to-many and one many-to-many association.

**🔥 1.5** Two or more many-to-many associations.

💡 **1.6** **- Validations -** 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** **- Data Sources -** 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 CSV files and multiple endpoints from the same API will be considered separate data sources.

🕯️ **1.7** Data is pulled from a single data source. (Faker is 1 source)

💡 **1.7** Data is pulled from two data sources.

🔥 **1.7** Data is pulled from three or more data sources.

**For feature 1.7 the minimum number of table rows populated by your seed script is 200 rows. By this I mean 200 including rows from all tables.**

## 2 - Web Site Navigation

🕯️ **2.1** **- About Page -** 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 - Menu -** 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** - **Collection Navigation** - 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** **- Member Pages -** There exists a page (show?) for each of the individual entries of the data you have collected to display all the attributes of the collected data point.

💡 **3.3** **- Multi-model Data on Member Pages -** On the individual entry pages (3.2) data pulled from associated models is also present.

**🔥 3.4** **- Hierarchical Collection Navigation -** 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 - Pagination -** Large collections of data are presented using pagination. (You can use the [kaminari gem](https://github.com/kaminari/kaminari) or a javascript library to implement this feature.)

**🔥 3.6** **- Mapping -** Location data is presented on a map (either one map for the entire collection or a separate map for each data point). This feature will only be possible for students who have imported datasets that includes addresses or GPS coordinates. In other words, you can’t just add a random map to your project to get these marks. The map can be provided using the [Google Static Map API](https://developers.google.com/maps/documentation/static-maps/) or [Google Maps Javascript](https://www.driftingruby.com/episodes/plotting-points-on-a-map-from-scratch). (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](https://docs.mapbox.com/api/maps/#static).)

## 4 - Searching and Filtering

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

💡 **4.2** **- Hierarchical Search -** 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** **- Valid HTML -** All HTML generated by your app validates as HTML5 with no errors.

🕯️ **5.2 - ERB Conditional -** At least one of your ERB views includes a conditional (if or unless). *Don’t use the Demo example, use your own!*

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

## 

## 6 - Source Control

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

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

# 

# **Appendix A**

# Common Problems and Troubleshooting Guide

Here are some common problems students run into with this project along with some suggested solutions.

* [Seed Script Runs Without Error, But No Records Are Created](#_scajqe373nln)
* [Dataset Has Its Own Existing Primary and Foreign Keys](#_4l6oxl45tkay)
* [ActiveRecord Is Acting Weird When Importing / Querying Data](#_x55p8io3tlfq)
* [How to Add a New Column or a Foreign Key to a Table](#_gv2zusmo5ihx)
* [CSV Parsing Fails Even Though CSV Data Looks Correct](#_ty1vqb6giamb)

## 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

## 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.

## ActiveRecord Is Acting Weird When Importing / Querying Data

There are certain reserved words that will cause problems if used as table names or column names in your database. Unfortunately Rails will not prevent you from using these names, but if used they will cause ActiveRecord to break in strange ways.

**Some words you should avoid:** action, asset, desc, key, params, test, type

This is not a complete list. Check this app for more ActiveRecord reserved words:

[reservedwords.herokuapp.com](http://reservedwords.herokuapp.com)

## How to Add a New Column or a Foreign Key to a Table

If you need to add a new column or a new foreign key to an existing table/model you can do this using a migration. Here are two examples.

**Example 1:** Imagine you need to add a stock\_quantity integer column to an existing Product model (and products table):

rails g migration AddStockQuantityToProducts stock\_quantity:integer

**Example 2:** Imagine you need to add a FK reference to a category (one-to-many) to an existing Product model (and products table):

rails g migration AddCategoryRefToProducts category:references

The migration file created by those commands would then have to be execute:

rails db:migrate

## CSV Parsing Fails Even Though CSV Data Looks Correct

Problems parsing CSV data are sometimes due to the character encoding used when the file was saved and/or if the file starts with a hidden [BOM](https://en.wikipedia.org/wiki/Byte_order_mark).

Opening the CSV in Excel and saving it again as a UTF-8 CSV will often solve the problem.