**Ruby on Rails Lab Web**

In this lab, you are being asked to create a content management system that allows authenticated users to write reviews of certain restaurants they have visited and unauthenticated users to read these reviews. The requirements for the system are as follows:

1. Authenticate users to add reviews
2. Allow authenticated users to add, edit and delete reviews
3. Allow non-authenticated users to view the records

Be sure to create an appropriate database to store your restaurant reviews.

Scenario:

I have been asked to create a CMS for Welp, an up-incoming competitor to Yelp in the market of reviewing restaurants, to help their users decide who is worth their money. This company is chock full of train enthusiasts and The Legend of Zelda fans, so they have decided to create the entire system using Ruby on Rails. They are also handling all the aspects of hosting the system once it has been created.

Needs

1. CRUD
   1. Must have basic CRUD functionality to create, update, and delete restaurants and reviews
2. Authentication- Use gem Devise
   1. Users must be able to create an account, and be able to login to said account
   2. Users must only be able to create, update and delete restaurants/reviews once they are logged in
      1. Additionally, users may only edit and delete Restaurants/Reviews that they themselves have created
3. Database and Models
   1. Restaurant
      1. Restaurant\_id
      2. Title
      3. Location
      4. Description
      5. Review\_id
         1. Foreign key
         2. One-to-many
      6. User\_id
         1. Foreign key
         2. One-to-one
   2. Reviews
      1. Review\_id
      2. Rating
      3. Comment
      4. User\_id
         1. Foreign Key
         2. One-to-one
      5. Restaurant\_id
         1. Foreign Key
         2. One-to-one
   3. Users
      1. Potentially use Devise gem to handle this aspect
4. Wants
   1. I would like to implement some way to easily organize the restaurants, perhaps into categories, and be able to sort by using them
   2. Profanity Filter
      1. When Brian analyzed an early prototype, he mentioned that a profanity filter would be a nice edition