Model Specifications

Teams should:

1. Have all proper relationships specified.
2. Have a name and description.
3. Have values that are the proper data type.
4. Only be deleted if it has no users associated with it; otherwise, it should be made inactive.
5. Only be edited/deleted by admins.
6. Have a method to calculate the number of total points accrued.
7. Have a method to find the top-X scorers within the team.
8. Have make_active and make_inactive methods.
9. Have the following scopes:
    a. 'active' – returns all active teams
    b. 'inactive' – returns all inactive teams

Users should:

1. Have all proper relationships specified.
2. Have a first_name, last_name, username, password, and email.
3. Have a required connection to team.
4. Have values that are the proper data type; role can be 'admin', 'regular', or blank.
5. Have unique usernames.
6. Have a callback that automatically sets role to 'regular' if not set.
7. Not be allowed to join inactive teams.
8. Validate that email is unique.
9. Never be deleted, only made inactive by the user that created it.
10. Have a method to calculate the number of total points accrued.
11. Have a method to list all challenges completed.
12. Have make_active and make_inactive methods.
13. Have the following scopes:
    a. 'for_team' – returns all users on a particular team (Parameter: team)
    b. 'for_challenge' – returns all users that have a submission associated with a particular Challenge (Parameter: challenge)
    c. 'for_role' – returns all users that have a particular role (Parameter: role)
    d. 'by_last_name' – orders users alphabetically by last name
    e. 'by_first_name' – orders users alphabetically by first name
    f. 'active' – returns all active users
    g. 'inactive' – returns all inactive users

Challenges should:

1. Have all proper relationships specified.
2. Have a name and num_points.
3. Have values that are the proper data type, and positive integer point values.
4. Only be deleted if it has no submissions associated with it.
5. Only be edited/deleted by admins.
6. Have the following scopes:
    a. 'alphabetical' – orders challenges in alphabetical order
    b. 'for_user_completed' – returns all challenges that have a submission associated with a particular user (Parameter: user)

   c. ‘for_user_incomplete’ -returns all challenges that a particular user does not have a submission for (Parameter: user)
   d. ‘for_category’ – returns all challenges that are part of a particular category (Parameter: category)

Submissions should:

1. Have all proper relationships specified.
2. Have a required connection to both challenge and user.
3. Have values that are the proper data type.
4. Have unique user-challenge pairs.
5. Only be edited/deleted by the user that submitted it.
6. Have a callback that automatically sets date_completed to the current date.

Photos should:

1. Have all proper relationships specified.
2. Have a required connection to submission.
3. Have values that are the proper data type.
4. Only be edited/deleted by the user that submitted it.
5. Have the following scopes:
   a. ‘chronological’ – orders photos by submission date (most recent first)
   b. ‘by_challenge’ – orders photos alphabetically by challenge name
   c. ‘by_user’ – orders photos alphabetically by user (last_name, first_name)
   d. ‘for_team’ – returns all photos submitted by a particular team (Parameter: team)
   e. ‘for_user’ – returns all photos submitted by a particular user (Parameter: user)
   f. ‘for_date’ – returns all photos submitted on a particular date (Parameter: date)
   g. ‘for_challenge’ – returns all photos submitted for a particular challenge (Parameter: challenge)
   h. ‘for_category’ – returns all photos submitted for a challenge of a particular category (Parameter: category)
   i. ‘for_past_days’ – returns all photos submitted in the past X days (Parameter: X)