Specifications for Phase 2

As part of phase 2, you will have to create models for `Employee`, `Store`, and `Assignment` and write unit tests for these models. To make sure you are able to build these models and tests, below are some specs for each of these models.

Stores must:

1. have all proper relationships specified
2. have values which are the proper data type and within proper ranges. Remembering the 2nd Rule of Software Development, that means:
    - zip code must be present and should be legitimate five digit zip code, validated with a regex
    - phone must be present and a 10 digit number, but the user may input values with dashes, dots or other common formats – e.g., 999-999-9999; 999.999.9999; (999) 999-9999 are all acceptable
3. have a store name and it must be unique (case-insensitive)
4. have a street and city (no other validation required)
5. have a state that is (*for now*) either PA, OH, or WV
6. have phone values that are saved in the system as a string of 10 digits only, regardless of the format the user inputs.
7. have the following scopes:
    - 'active' -- returns only active stores
    - 'inactive' -- returns all inactive stores
    - 'alphabetical' -- orders results alphabetically by store name
8. have the following methods:
    - 'make_active' -- which flips an active boolean from inactive to active and updates the record in the database
    - 'make_inactive' -- which flips an active boolean from active to inactive and updates the record in the database

---

Employees must:

1. have all proper relationships specified
2. have a first and last name (no other validation required)
3. have values which are the proper data type and within proper ranges. Remembering the 2nd Rule of Software Development, that means:
    - phone must be present and a 10 digit number, but the user may input values with dashes, dots or other common formats – e.g., 999-999-9999; 999.999.9999; (999) 999-9999 are all acceptable
    - social security number must be present and a 9 digit number, but the user may input values with dashes, spaces or other common formats – e.g., 999-99-9999; 999 99 9999; 999999999 are all acceptable
    - have a date of birth that is a proper date and at least 14 years in the past, as there are regulations against hiring anyone under the age of 14.

4. have social security numbers which are unique in the system.
5. have phone values that are saved in the system as a string of 10 digits only, regardless of the format the user inputs.
6. have social security number values that are saved in the system as a string of 10 digits only, regardless of the format the user inputs.
7. have a role that is either an employee, a manager, or an admin
8. have boolean methods associated with each role (e.g., `manager_role?`)
9. have the following scopes:
    - 'active' -- returns only active employees
    - 'inactive' -- returns all inactive employees
    - 'alphabetical' -- orders results alphabetically by last name, first name
    - 'is_18_or_older' -- returns all employees 18 years old or older
    - 'younger_than_18' -- returns all employees under 18 years old
    - 'regulars' -- returns all employees who have the role 'employee'
    - 'managers' -- returns all employees who have the role 'manager'
    - 'admins' -- returns all employees who have the role 'admin'

10. have the following methods:
    - 'name' -- which returns the employee name as a string "last_name, first_name" in that order
    - 'proper_name' -- which returns the employee name as a string "first_name last_name" in that order with a space between them
    - 'current_assignment' -- which returns the employee's current assignment or nil if the employee does not have a current assignment
    - 'over_18?' -- which returns a boolean indicating whether this employee is over 18 or not

- 'make_active' -- which flips an active boolean from inactive to active and updates the record in the database
- 'make_inactive' -- which flips an active boolean from active to inactive and updates the record in the database

---

Assignments must:

1. have all proper relationships specified
2. have start and end dates that are proper dates, if given. In addition:
   - every assignment must have a start date and it must be on or before the present date
   - an assignment's end date (if it exists) must be after its start date
3. have a store_id and it must be restricted to stores which exist and are active in the system
4. have a employee_id and it must be restricted to employees who exist and are active in the system
5. have the following scopes:
   - 'current' -- which returns all the assignments that are considered current
   - 'past' -- which returns all the assignments that have terminated
   - 'by_store' -- which orders assignments by store
   - 'by_employee' -- which orders assignments by employee name (last, first)
   - 'chronological' -- which orders assignments chronologically with the most recent assignments listed first
   - 'for_store' -- which returns all assignments that are associated with a given store (parameter: store object)
   - 'for_employee' -- which returns all assignments that are associated with a given employee (parameter: employee object)
   - 'for_role' -- which returns all assignments that are associated with employees of a given role (parameter: role)
   - 'for_date' –- which returns all assignments that were/are active on that particular date (parameter: date)
6. have a callback that will automatically end the employee's current assignment if a new assignment is created for that employee.

NOTE: In this phase we will *not* validate that any `active` field is actually a boolean.