The Digital Cookbook

❖ Introduction

The task is to develop a digital cookbook – a culinary recipes repository with simple search capabilities with an HTML interface. The application must be developed in Ruby on Rails and make use of SQLite3 database

❖ Models

The application has two models: `Recipe` and `Ingredient`. There is a one-to-many association between them.

The `Recipe` model has at least the following attributes: `name` and `instructions` store textual data and `cooking_time` stores numeric data. The `Ingredient` model has a `name` attribute which stores textual data. Both models may have other columns as well, e.g. when they're required by associations.

The following model validations are required:

- The recipe's `name` must be present and no longer than 100 characters.
- The recipe's `instructions` must be present and no longer than 1000 characters.
- The recipe's `cooking_time` must be present and be larger than 0.
- The ingredient's `name` must be present and no longer than 100 characters.

❖ Searching

When a user visits a root URL (GET request on `/` path), a search form is displayed. The form consists of:

- input of type text labeled "Cuisine"
- input of type text labeled "Ingredient"
- button labeled "Search".

Clicking on "Search" executes a search request. As the result, the user is navigated to a page which lists all the matched recipes.

If the "Cuisine" has been provided in the search form, then the result set is limited to recipes whose cuisine equals that provided value.

If the "Ingredient" has been provided in the search form, then the result set is limited to recipes which have an ingredient with the name that equals to that provided value.

If both fields have been left out blank, the result set includes all the recipes in the database. In case no recipes have been matched, then "No recipes found" should be displayed.

Every matched recipe should be displayed in a way that includes its name, instructions, cooking time (followed with "min" suffix), and names of the ingredients.

Matching recipes and ingredients may be performed either in a case-sensitive or a case-insensitive manner, both will pass the tests. Neither the order of displayed recipes matters.

See integration tests for more details. In order to simulate user's interactions, [Capybara](http://jnicklas.github.io/capybara/) has been employed. That said, knowledge of this gem is not necessary to understand the test suite.

- **Routes, controllers, and views**

  There are no restrictions on controllers, routes, and views, except for these which can be concluded from the search use case. Any HTML version and DOM structure work as long as it provides the functionality described.