Menu Management Backend

Instructions

- Use Rails or another web app framework you're familiar with.
- Place in a Github Repo and send us a link when you're done.
- The project is due 1 week after receiving this document, but you can work on it as much as you'd like.
- Write extensible code, make commits, and test as you would a production application. We'd rather see a well-constructed and well-tested app with fewer features.
- Complete as many levels as you wish, but complete them one at a time. Resist making decisions for future levels. Your commits should reflect this approach.
- Do not hesitate to ask us any questions you've got.

Requirements:

- Level 1: Basics
  - Create an object model for `Menu` and `MenuItem`s classes
  - `Menu` has many `MenuItem`s
  - `Menu` and `MenuItem` have typical data associated with restaurants
  - Illustrate behavior via unit tests
- Level 2: Multiple Menus
  - Introduce a `Restaurant` model, and allow `Restaurant`s to have multiple `Menu`s
  - `MenuItem` names should not be duplicated in the database
  - `MenuItem` can be on multiple `Menu`s of a `Restaurant`
  - Illustrate behavior via unit tests
- Level 3: Complex Menu Items
  - Extend the object model to enable the following behavior:

        - A diner can order a dinner salad one of two ways:
            - As a standalone dish, with selection of dressing
            - As a side of an entree, with selection of dressing
        - A diner can order a side of any dressing with any appetizer or entree
    - Illustrate this behavior with unit tests
- Level 4: Fred and Fran like Fish on Fridays
    - This is a bonus! We'd love to talk about your solution if you think it's a fun problem
    - Design and implement a prediction engine that predicts:
        - The likelihood of a customer dining on a particular day
        - What dish they are most likely to order

Evaluation

To ensure consistency, our Engineering team uses a rubric to score your submission according to the following criteria. Please show us your strengths for coding, testing, technical design, and attention to detail.

Changeability:

- We expect an extensible project. If requirements change, your code should be adaptable.

Iterative approach:

- We expect an iterative approach to complete each level. We appreciate it when you make us aware of the assumptions you used and decisions you made along the way.

Verification:

- We expect a low-bug or bug-free implementation of the project.

Validation with tests:

- We expect application behavior to be validated with unit tests.