Statement of Purpose

The software is intended to help an individual user answer the following questions:

1. Which recipes in my library can I create, given the ingredients in my pantry?
2. Given the need to feed # people, which ingredients must I buy to create certain recipes, less the amounts on hand?

Concept of Operation

1. The user can POST a new Recipe
    a. a Recipe consists of a `name`, a list of one or more Ingredients, a number of `servings`, a list of zero or more `tags`, a unique identifier `recipeID`
        i. a `name` is a string of at least 1 character, should be short enough to be displayed on the intended output device, and should be unique among Recipes
        ii. an Ingredient consists of a `name` and a Amount
            1. the `name` is a string of at least 1 character, should be short enough to be displayed on the intended output, and should be unique among Ingredients
            2. the Amount consists of a `quantity` and a `unit` of measure
                a. `unit` is a string containing standard abbreviation for a SI unit of mass, unit of volume, or the word ‘each’
                b. `quantity` is an integer
                    i. If the user enters a floating point value e.g. [1.5, “L”] the software will silently convert to the nearest whole-number unit [1500, “mL”]
            3. When adding an Ingredient, the user should be able to select Ingredients the software already knows about from other Recipes, or enter the `name` of a new Ingredient
        iii. `servings` is a whole number equal to or greater than 1 representing the number of people intended to be served by the proportions in this Recipe
        iv. a `tag` is a user-defined descriptive string which can aid the user in searching and filtering Recipes
        v. the `recipeID` is an alphanumeric, randomly generated string
    b. All defined Recipes are added to a single instance of RecipeBook which consists of zero or more Recipes

2. The user can GET the RecipeMenu
    a. a RecipeMenu consists of zero or more Recipes
    b. The software will generate() RecipeMenu based on one or more conditions:
        i. all Recipe in RecipeBook (default)
        ii. only Recipe whose `name` contains a given string
        iii. only Recipe which matches one or more `tag`
        iv. only Recipe which contains a particular Ingredient
        v. only Recipe which serve at least # portions
        vi. only Recipe whose ingredients are contained in the Pantry
    c. The user can GET a Recipe from RecipeMenu to view its details
    d. The user can mass DELETE all currently displayed Recipe from RecipeBook
        i. the software will ask the user to confirm the decision to mass delete()
3. The user can GET a Recipe from any view containing one or more Recipe i.e. RecipeMenu, MealPlan
    a. The user can see which Ingredient are contained in Pantry in sufficient Amount
    b. The user can choose to edit() the Recipe details
        i. The software will present the same view as defining a new Recipe, prepopulated with existing Recipe components
        ii. While editing, the user can choose to save() or discard() changes
    c. The user can choose to delete() the Recipe from the RecipeBook
        i. the software will ask the user to confirm the decision before committing changes
    d. The user can choose to create() the Recipe (i.e. prepare it as a meal)
        i. creating a Recipe will deduct the specified Amount of each Ingredient from the Pantry
            1. if the Recipe Ingredient and Pantry Ingredient have Amount with different `unit` of measure, the software will convert between them
    e. The user can choose to plan() the Recipe
        i. the software will ask the user how many servings they wish to plan for
        ii. the software will create a new Meal
            1. a Meal consists of a Recipe and a `number` of servings
        iii. the software will add the new Meal to the MealPlan
            1. the MealPlan consists of zero or more Meals

4. The user can GET the MealPlan
    a. the MealPlan is displayed as a list of Meals
        i. the user can choose to edit() the number of servings or remove a Meal from the plan by setting the serving count to zero
    b. the user can see which planned Meals are missing sufficient Ingredients in Pantry
        i. When calculating Ingredient shortages, the user should have the option to ‘fuzz’ the calculation i.e. don’t report Ingredients within ##% of the required Amount
    c. the user can choose to generate() a ShoppingList from the current MealPlan
        i. a ShoppingList consists of a list of one or more Ingredients
        ii. a ShoppingList is generated by adding all Ingredients from each Recipe from each Meal in MealPlan
            1. if Ingredient already present, increase Amount
            2. decrease Amount of each Ingredient on ShoppingList by the Amount of same Ingredient in Pantry
        iii. the user can choose to discard() the ShoppingList
        iv. the user can choose to stock() the ShoppingList
            1. the software will ask the user for the Amount of each Ingredient to stock
            2. the software will provide the Amount from the ShoppingList as default and allow the user to change the number and unit of measure, since the user will not likely be able to purchase the exact amount on the list
            3. The software will update the Amount of each item in Pantry and discard the ShoppingList
5. The user can GET the contents of the Pantry
    a. the Pantry consists of a list of zero or more Ingredient
    b. the user can choose to add() a new Ingredient to Pantry
        i. the software will ask user for name and Amount
            1. the software will suggest names of Ingredients from already known Recipe
    c. the user can choose to update() an existing Ingredient by changing the name and/or Amount
        i. behavior: altering an Ingredient in Pantry will not affect any Recipe which depend on said Ingredient
6. Backup/restore
    a. The user can export() the RecipeBook, an individual Recipe, or the Pantry contents
    b. Exported objects are serialized into JSON and the resulting text is base64 encoded
    c. When choosing to import() an object, the user can choose to overwrite any existing Recipe or Ingredient with the same name
        i. If the option to overwrite is not chosen, attempting to import an object that already exists should fail with error

Architecture

1. The user interacts with the software using a web browser
2. The software run on a MERN stack or equivalent technologies
3. The software can be run on a user's local computer, mobile device, or a remote host
    a. If hosted in the cloud, the site should be secured behind an appropriate authentication service
4. The software uses MVC pattern to decouple business logic from presentation and data stores.

Document Conventions

1. Capitalized nouns: class name
2. `Fixed width font:` member name, primitive or string
3. verb() - method
4. VERB - software behavior reflected in HTTP request method