Rapid Recipes

Anisha Nakagawa, Annabel Consilvio, Maggie Jakus
Olin College of Engineering, Spring 2015

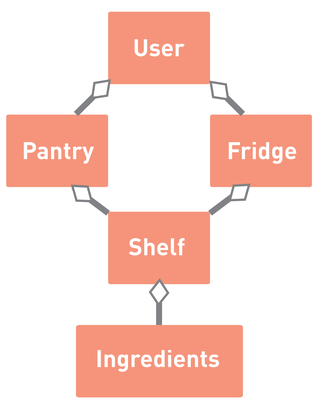
Background

College students often have a limited number of ingredients that may not appear to be easily used together. We created a program that lets the user search for recipes based on the ingredients they currently have.

Features

- a Create a user account
- b Save a list of ingredients specific to a user in a "pantry"
- c Enter lists of ingredients to search through in a "fridge"
- d Ability to search by max cook time
- e Ability to store user's pantries
- f Access through an interactive web application

Class Structure



Each user objects has a pantry and a fridge as attributes. The pantry is stored between uses so the user can store commonly used ingredients, while the fridge is re-entered each time. The pantry and fridge as both extend the shelf class, which has a list of ingredients as an attribute.

GUI
(webpage)

Presents visuals for users to login, add ingredients to search for recipes, and displays images and information about the relevant recipes. Uses HTML and CSS to generate the webpage and Javascript to interact with the Python backend.

Flask
(integration)

Flask sends information to the back-end recipe program and receives information from the HTML template. It routes the user's information to back-end to create a user object and with ingredients, and then runs the recipe search.

Heroku
(server)

Constantly runs the flask program on a separate machine and installs the necessary file and library dependencies. Through the server, the HTML GUI can be accessed from www.rapid-recipes.herokuapp.com.

Python
(back-end)

Yummly API returns recipes that include available ingredients. Uses our search algorithm to find usable recipes that do not exceed the user's max cook time.

Yummly API

The recipe search uses the Yummly API as recipe database, which is searchable by ingredient through a URL call. The back-end python functions format the correct API call for each ingredient, and stores the list of ingredients in a list.

MongoDB
(database)

Information about the users and recipes is saved in Mongo DB. The users' pantry is stored between successive runs, and the list of recipes containing common ingredients are memoized to reduce the runtime of the program. The database can be accessed from the server or any computer.

Program Structure

The program searches for recipes in a back-end python file, which is connected to the web-based interface through Flask. The back-end stores information about the users and recipes using Mongo DB. The entire application is hosted on a server with Heroku so that it can be accessed through a URL.

An example of returned, relevant recipes on the webpage. If you hover over an image, it shows the title of the recipe and how long it takes to cook. Each image is a link to a seperate page of instructions.

Search Algorithm

This program searches through the user's available ingredients. A list of all potential recipes is built up calling the Yummly API (with a url call) for each ingredient. It creates a list of recipes that contain at least one available ingredient.This list is then filtered to remove items where the user is missing one or more ingredients. These recipes for each ingredient are memoized to reduce calls to the Yummly API, and decrease the program runtime.

Future Work

The class structure is designed so that the search program can be further customized, so future iterations could include dietary restrictions, meal type, food ethnicity, or allergen information. In addition, the Flask Login Manager would provide support for accessing the web application with multiple users simultaneously, which is currently not supported.

Attributions

1. Flask 0.10 and documentation
2. MongoLab and documentation
3. Heroku and documentation
4. Python ast library, for string to list conversions
5. GUI framework from http://flask.pocoo.org/docs/0.10/patterns/jquery/
6. Cook-time slider code from JSFiddle