Term 1 Assessment 2 - Terminal Application

https://github.com/philrussel21/raid-my-kitchen-terminal-app

App Concept

We’ve all heard about amazing chefs all over the world with their mouth watering recipes and their out of the box approach. What you did not hear about is a chef that can come up with a delicious recipe from little to no input at all.

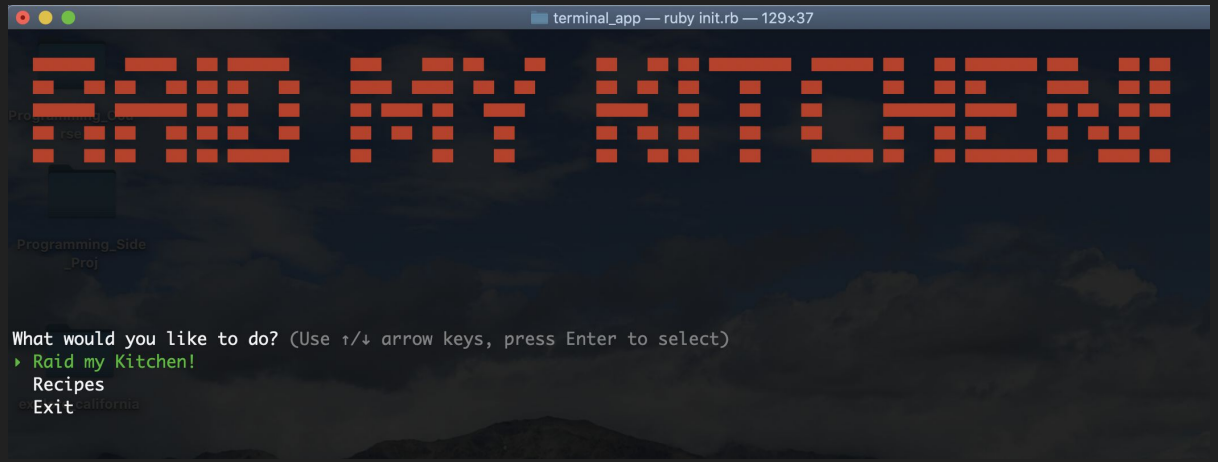
This app is to address that unspoken need. The need to bring up meal ideas by simply providing how much time you can spare cooking or by simply providing the base meat of your choice!

The Features & Flow of the App

Key Features

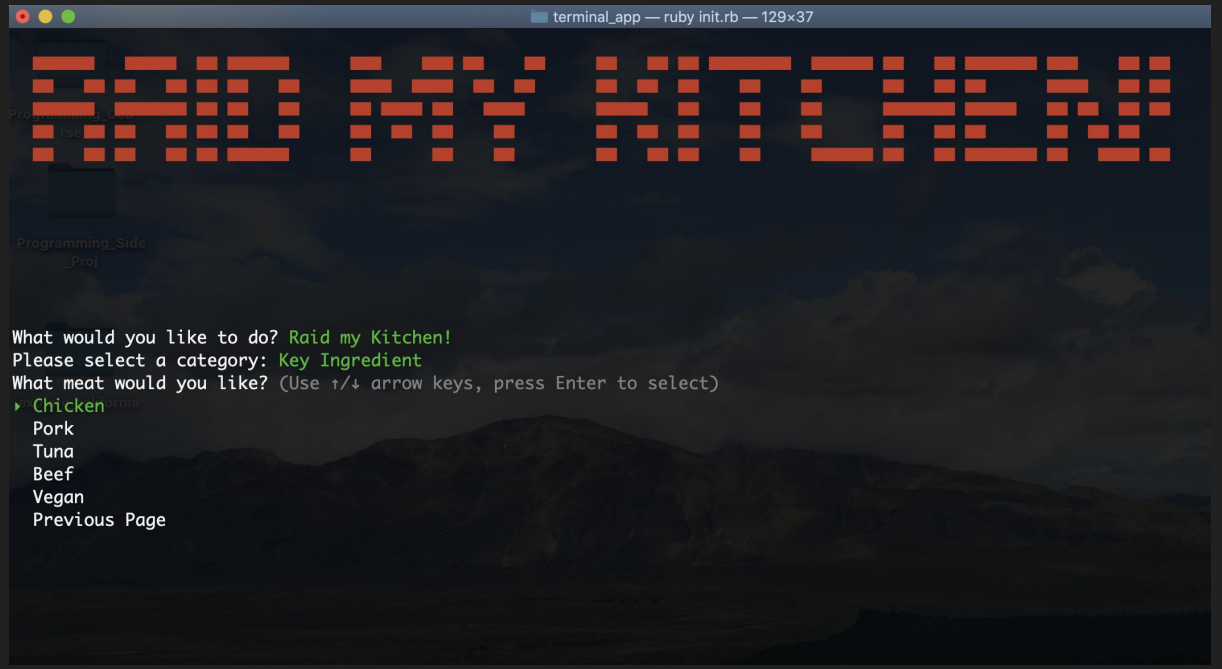
The key features of this app is divided into two sub categories.

- Raid My Kitchen! Features
- Recipes Features



Raid My Kitchen!

- Key Ingredient Option
  - Lists the key ingredient(meat base) of all the dish from the recipe database.
  - When the user makes a selection, the app would then show all the available recipes from the database that matches the choice.
  - The app then prints the ingredients and each method for the selected dish.

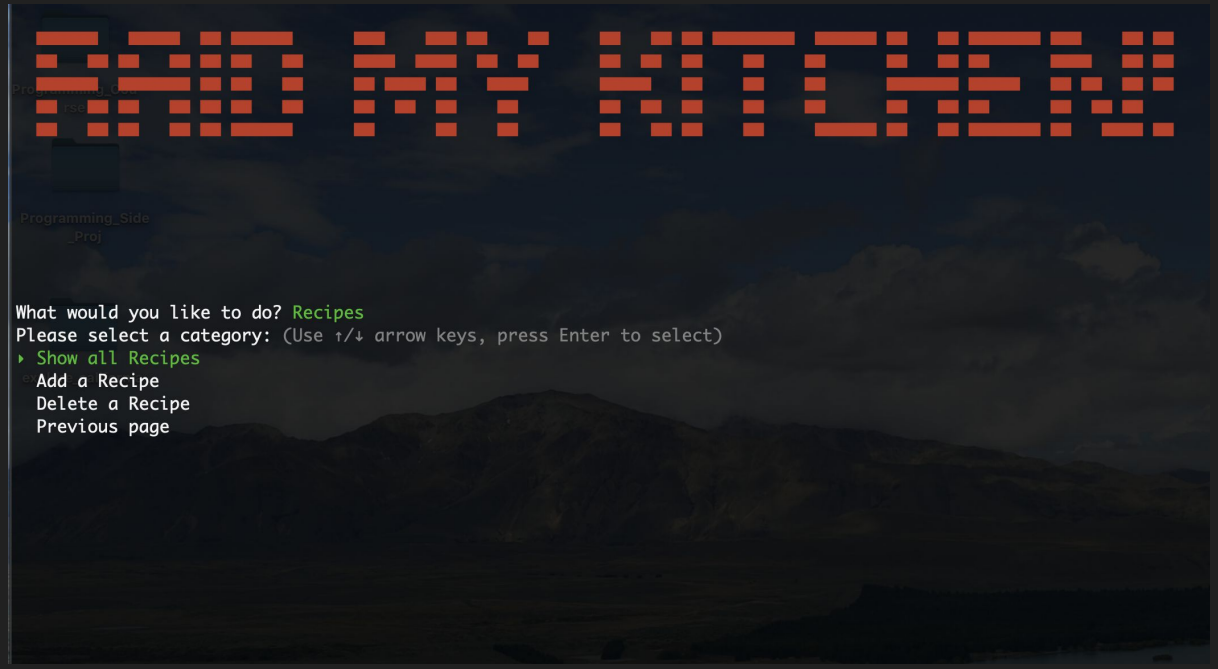


Raid My Kitchen!

- Cooking Time
  - Asks user for the desired cooking and prepping time in minutes then lists all the recipes that can be done within the given timeframe.
  - The app then prints the ingredients and each method for the selected dish.
  - Custom Exception Classes are in place to catch invalid inputs.

Recipes

- Show All Recipes
  - Lists all recipes available in the database. When user makes a selection, prints the Ingredients then each method of the selected dish.



Recipes

- Add a Recipe
  - Lets the user add a recipe to the database.
  - Custom Exception Classes were put in place to catch invalid inputs or duplicate recipe.

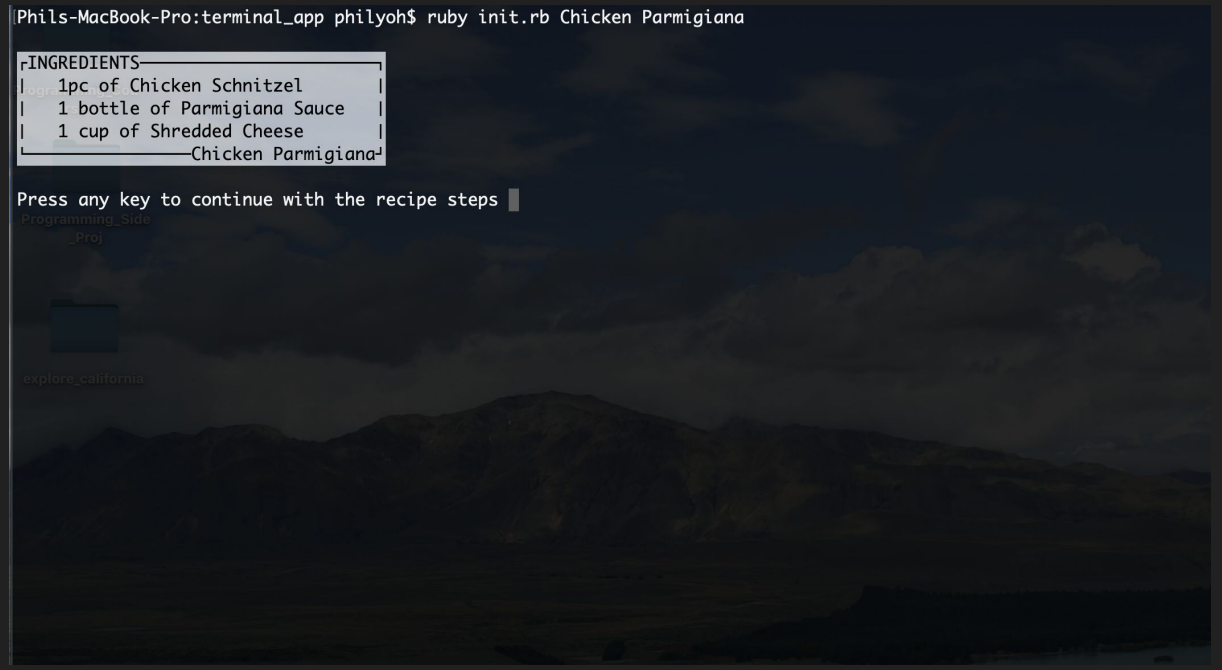
Recipes

- Delete a Recipe
  - Lets the user remove a recipe from the database.
  - Confirmation prompt was put in place to confirm action.

Optional Feature

The app has an optional feature that accepts command line arguments. Simply put the name of the dish desired and it would output its ingredients and methods if it matches anything in the recipe database.

If the name of the dish cannot be found in the recipe database, the app would run its normal course.



Roadblocks & Challenges

- Creating a class object from user input.
  - ➢ Solution found on stackoverflow.
- gets() method failing when command line args are passed.
  - ➢ ARGV.clear after the iteration that used the args.
- Tty-box gem issue
  - ➢ Custom Exception Class to catch Argument Error originating from gem source code.

```ruby
#creates a new object from DishFormat::NewDish from user input
new_dish_object = 'DishFormat::NewDish'.split('::').inject(Object){|object,clazz|object.const_get clazz}
new_dish = new_dish_object.new(new_dish_name,new_dish_meat,new_prep_and_cook_time)
new_dish.ingredients = new_dish_ingredients
new_dish.method = new_dish_methods
$default_recipe.add_dish(new_dish) #Adds the dish to the local database
```

```ruby
def main
  if !ARGF.argv.empty?
    arg = ARGF.argv.join(' ')
    display_ing_and_method(arg) if $default_recipe.all_recipes.keys.include?(arg)
    ARGV.clear
    clear
    app_intro
    return raid_my_kitchen
```

```ruby
    #Potential to throw an error if the characters in the ingredients array is not
    # box Title and Dish name
    box = TTY::Box.frame(style: {
      fg: :black,
      bg: :white,
      border: {
        fg: :black,
        bg: :white
      }
      }, title: {top_left: "INGREDIENTS", bottom_right: "#{choice}"}) do
      dish_ing
    end
    puts box
  #rescues the ArgumentError from above.
  rescue ArgumentError
    $default_recipe.all_recipes[choice].ingredients.each{|ing| puts ing}
  end
```

That’s all folks!