



The Recipe Bot

Ruby Terminal Application





Why this app?

- This app helps users quickly find recipes based on what ingredients they already have at home and reduces the amount of ingredients they have to go out and buy.
- This prevents users from wasting time finding recipes that involve ingredients they don't have at home which is a very common and very frustrating problem.
- It also helps prevent them from doubling up on ingredients or not buying enough when shopping for multiple recipes as it tells the user exactly how much of each ingredient they need for each recipe.

Outline

Overview & Features

Application Logic

Challenges, Ethical Issues & Favourite Parts

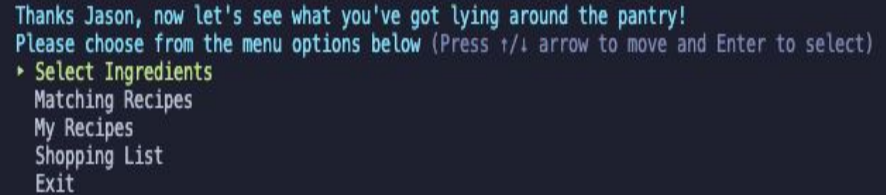
Questions

Overview & Features



Navigation

- The app can be navigated by using the “up”, “down” arrows and the enter key.
- Most of my user input is taken using TTY-Prompt and i have done this to limit the amount of errors that a user could make while using my application
- Every menu has a prompt instructing the user how to navigate through the menu items and make selections.



```
Thanks Jason, now let's see what you've got lying around the pantry!  
Please choose from the menu options below (Press ↑/↓ arrow to move and Enter to select)  
▶ Select Ingredients  
  Matching Recipes  
  My Recipes  
  Shopping List  
  Exit
```



'Select Ingredients' Feature

- Users will be able to view a list of ingredients populated from all available recipes in the app.
- This feature gives the user the ability to select ingredients from that list that they currently have at home.
- Once they select an ingredient it is added to their list.
- If they make a mistake they can also remove the ingredients from their list.

```
What ingredients do you currently have at home? You can add as many ingredients as you like. Scroll down for more options. Once you are done adding ingredients you can select "Finished Adding Ingredients to go back. (Press t/l +/- arrow to move and Enter to select)
```

```
▶ 1.5kg whole chicken  
  ½ cup melted butter  
  3 fresh whole rosemary sprigs  
  400g can mixed beans drained  
  1 cup grated cheddar cheese  
  ½ cup chopped shallots
```



'Matching Recipes' Feature

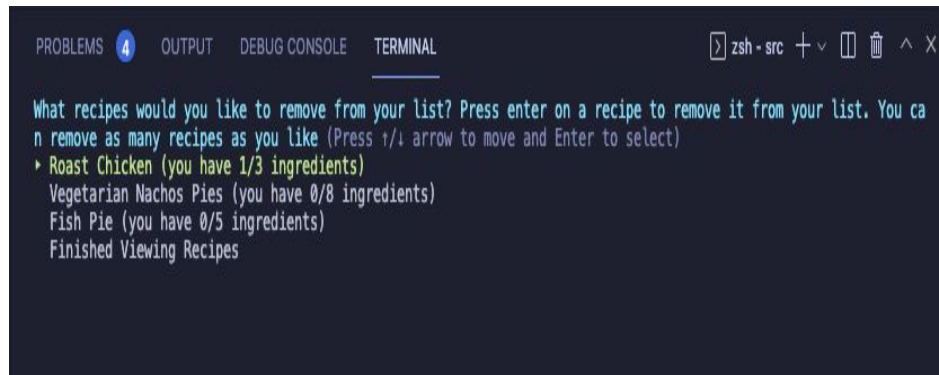
- Once users have selected at least 1 ingredient, they will be able to view their matching recipes.
- Each recipe will display the total matching ingredients next to it so the user can choose recipes that have the most matching ingredients.
- The user can select as many recipes as they want and add them to their list of recipes.
- Users will also be able to remove recipes that they no longer wish to have on their list.

```
What recipes would you like to add to your list? Press enter on a recipe to add it to your list. You can add as m
any recipes as you like (Press ↑/↓ arrow to move and Enter to select)
> Roast Chicken (you have 1/3 ingredients)
  Vegetarian Nachos Pies (you have 0/8 ingredients)
  Fish Pie (you have 0/5 ingredients)
  Italian Meatballs (you have 0/4 ingredients)
  Mediterranean Rosemary Skewers (you have 0/4 ingredients)
Finished Adding Recipes
```



'My Recipes' Feature

- This is where users can view their selected recipes.
- They will be able to select a recipe to open up the full recipe which includes the cooking instructions.




```
PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL
zsh - src + v [ ] [ ] ^ X

What recipes would you like to remove from your list? Press enter on a recipe to remove it from your list. You can remove as many recipes as you like (Press ↑/↓ arrow to move and Enter to select)
▶ Roast Chicken (you have 1/3 ingredients)
  Vegetarian Nachos Pies (you have 0/8 ingredients)
  Fish Pie (you have 0/5 ingredients)
  Finished Viewing Recipes
```




'Shopping List' Feature

- Once users have selected their recipes from the matching recipes, they will be able to view a shopping list of all of the ingredients they are missing from their selected recipes.

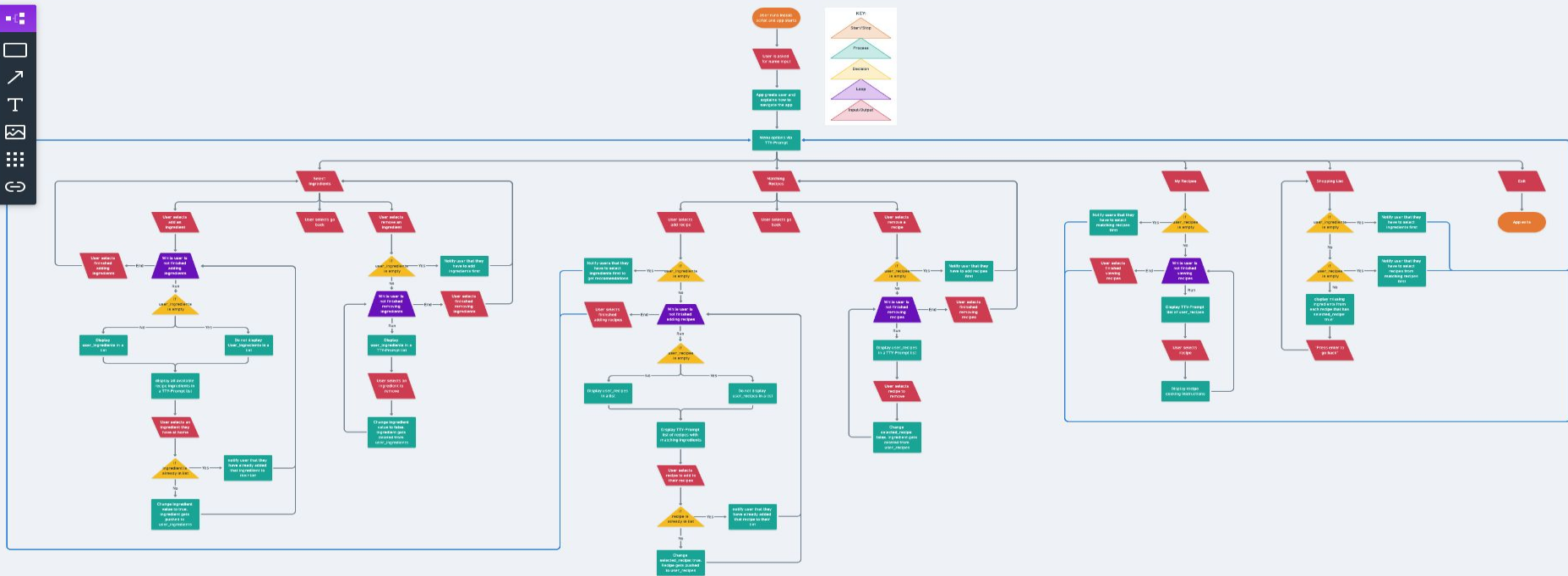


```
PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL
> zsh - src + v [ ] [ ] ^ X

Here are the ingredients you need to buy
1/2 cup melted butter
3 fresh whole rosemary sprigs
400g can mixed beans drained
1 cup grated cheddar cheese
1/2 cup chopped shallots
1 medium red capsicum finely chopped
2 sheets puff pastry
1 handful tortilla chips
2 tbsp chopped coriander
8 jalapeños chopped
800g frozen mixed vegetables thawed
2 tbsp plain flour
400g skinless smoked cod, trout or salmon 3cm pieces
1 1/2 cups fish stock
3 sheets puff pastry
press 'Enter' to go back
```

Application Logic

Control Flow Diagram





Main.rb

- This stores all of the main feature functions
- It also has all of the main menu options and all submenus using TTY-Prompt
- Importing recipes from the seed file

```
#
# WELCOME SCREEN AND MAIN MENU LOGIC
#
system 'clear'
puts artii.asciify('The Recipe Bot').colorize(:green)
name = ''
if name != ''
  puts "Welcome to your personalised recipe suggestion bot #{name}".colorize(:cyan)
else
  puts "Welcome to your personalised recipe suggestion bot! What is your Name?".colorize(:cyan)
  name = gets.chomp
end
system 'clear'
puts "Thanks #{name}, now let's see what you've got lying around the pantry!".colorize(:cyan)
option = ''
user_ingredients = []
user_recipes = []
while option != 'Exit'
  option = main_menu
  case option
  when 'Select Ingredients'
    system 'clear'
    add_remove_option = ''
    while add_remove_option != 'Go Back'
      add_remove_option = ingredient_sub_menu
      case add_remove_option
      when 'Add Ingredients'
        system 'clear'
        added_ingredient = ''
        while added_ingredient != 'Finished Adding Ingredients'
          puts user_ingredients
          added_ingredient = add_ingredients_sub_menu
          add_ingredients(added_ingredient, user_ingredients)
        end
      when 'Remove Ingredients'
        system 'clear'
        if user_ingredients != []
          removed_ingredient = ''
          while removed_ingredient != 'Finished Removing Ingredients'
            puts user_ingredients
            removed_ingredient = remove_ingredients_sub_menu(user_ingredients)
            remove_ingredients(removed_ingredient, user_ingredients)
          end
        else
          system 'clear'
          puts "You need to add some ingredients first".colorize(:red)
        end
      end
    end
  next
end
system 'clear'
when 'Matching Recipes'
  system 'clear'
  if user_ingredients != []
    add_remove_option = ''
    while add_remove_option != 'Go Back'
      add_remove_option = recipe_sub_menu
      case add_remove_option
      when 'Add Recipe'
        system 'clear'
      end
    end
  end
end
end
```



Seed.rb

- This stores all of the recipes
- Originally i wanted to use an API but i ran out of time so it's hard coded for now.

```
1 require_relative './recipes'
2 require_relative './individual_recipe'
3
4 def seed
5   recipes = Recipes.new(
6     [IndividualRecipe.new(
7       'Roast Chicken',
8       {
9         '1.5kg whole chicken' => false,
10        '¼ cup melted butter' => false,
11        '3 fresh whole rosemary sprigs' => false
12      },
13      [
14        'Preheat air fryer to 180C, for 4 minutes.',
15        'Pat the chicken dry with paper towel.',
16        'Brush completely with butter.',
17        'Season with sea salt and cracked pepper and lots of lovely fresh rosemary.',
18        'Place chicken into air fryer basket.',
19        'Cook at 180°C, for 50-60 minutes.',
20        'Let the roast chicken rest for a few minutes before removing carefully with tongs.'
21      ]
22    ),
23    IndividualRecipe.new(
24      'Vegetarian Nachos Pies',
25      {
26        '400g can mixed beans drained' => false,
27        '1 cup grated cheddar cheese' => false,
28        '½ cup chopped shallots' => false,
29        '1 medium red capsicum finely chopped' => false,
30        '2 sheets puff pastry' => false,
31        '1 handful tortilla chips' => false,
32        '2 tbsp chopped coriander' => false,
33        '8 jalapeños chopped' => false
34      },
35      [
36        'Mix beans, cheese, shallots and capsicum in a bowl.',
37        'Set aside.',
38        'Pre-heat Pie Maker.',
39        'Cut out 12 pie bases.',
40        'Line the pie holes with pastry.',
41        'Crush a few tortilla chips in each base then fill with the beans mixture.',
42        'Sprinkle with coriander and a few jalapeños; use more if you like it spicy.',
43        'Close and cook for 15minutes.',
44        'Remove pies and cool slightly.',
45        'Serve with more crushed tortillas and a dollop of salsa and sour cream or your Guacamole.'
46      ]
47    ),
48    IndividualRecipe.new(
49      'Fish Pie',
50      {
51        '800g frozen mixed vegetables thawed' => false,
52        '2 tbsp plain flour' => false,
53        '400g skinless smoked cod, trout or salmon 3cm pieces' => false,
54        '1½ cups fish stock' => false,
55        '3 sheets puff pastry' => false
56      },
57      [
58
```

Recipe.rb

- This stores all of the recipe class objects.
- I also have the following class functions:
- Display_ingredients - gives a list of all available ingredients.
- Selected_recipes - finds recipes that user has selected and returns those associated ingredients.
- Display_missing_ingredients - returns only the false ingredients from selected ingredients for the shopping list.

```
require_relative './individual_recipe'

2
3
4 class Recipes
5   attr_reader :individual_recipe
6
7   def initialize(individual_recipe)
8     @individual_recipe = individual_recipe
9   end
10
11   # MANUALLY TESTED - displays all ingredients from all available recipes
12   def display_ingredients
13     available_ingredients = []
14     @individual_recipe.each do |item|
15       available_ingredients.push(item.ingredients.keys)
16     end
17     available_ingredients
18   end
19
20   # TESTED - finds the recipes that the user has selected based on the selected_recipe attribute being
21   # set to true and returns an array of recipe ingredients
22   def selected_recipes
23     selected_recipes = []
24     @individual_recipe.each do |i|
25       selected_recipes.push(i.ingredients.each { |k, _v| k }) if i.selected_recipe == true
26     end
27     selected_recipes
28   end
29
30   # TESTED - displays all of the ingredients that have their values set to false (meaning the user has
31   # not selected those ingredients)
32   def display_missing_ingredients(ingredients)
33     unselected_ingredients = []
34     missing_ingredients = []
35     ingredients.each do |item|
36       unselected_ingredients.push(item.select { |_k, v| v == false })
37     end
38     missing_ingredients.push(unselected_ingredients.each { |i| puts i.keys })
39     missing_ingredients
40   end
41 end
```

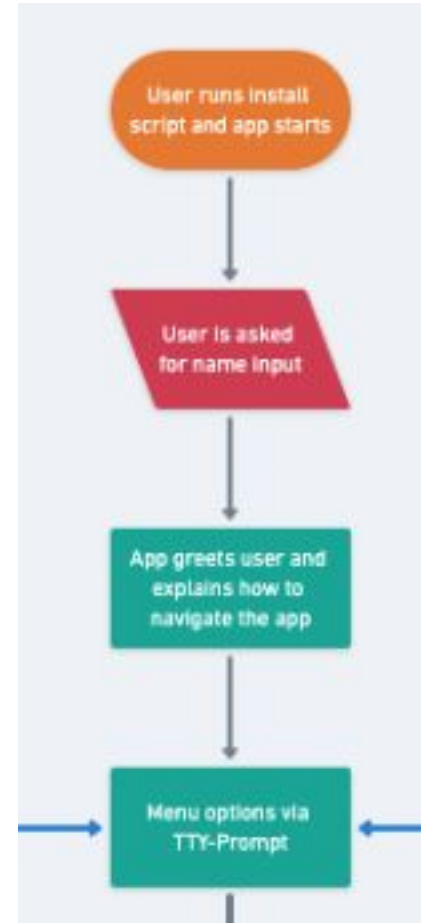
Individual_recipe.rb

- This stores all of the individual_recipe class objects.
- I also have the following class functions:
- To_s - which overrides the default string display so i can show matching ingredients.
- Selected_ingredients - increments every time a user selects an ingredient.
- Print_full_recipe - prints the full recipe method to the screen

```
1 require 'colorize'
2
3 class IndividualRecipe
4   attr_reader :name, :instructions, :serves
5   attr_accessor :ingredients, :selected_recipe, :selected_ingredients
6
7   def initialize(name, ingredients, instructions, serves)
8     @name = name
9     @instructions = instructions
10    @ingredients = ingredients
11    @serves = serves
12    @selected_ingredients = 0
13    @selected_recipe = false
14    @total_ingredients = 0
15  end
16
17  def to_s
18    "#{@name} (you have #{@selected_ingredients}/#{@ingredients.length} ingredients)"
19  end
20
21  # TESTED - increments @selected_ingredients for each ingredient that has a value of true
22  def selected_ingredients
23    @ingredients.each do |_k, v|
24      # if item.has_value?(true)
25      @selected_ingredients += 1 if v == true
26    end
27    # used below return for test case
28    @selected_ingredients
29  end
30
31  # MANUALLY TESTED - prints the full recipe method
32  def print_full_recipe
33    @name.to_s
34    puts 'Ingredients:'.colorize(:cyan)
35    (@ingredients.each { |k, _v| puts k }).to_s
36    puts 'Instructions:'.colorize(:cyan)
37    (@instructions.each { |i| puts i }).to_s
38    "Serves: #{@serves}"
39  end
40 end
```

'Navigation' Logic

- The navigation logic will be taken care of by TTY-Prompt
- There will be a main menu with case options
- Inside each main menu item there will also be TTY-Prompt sub-menus that the user can interact with to add/remove/view ingredients/recipes/shopping list



- Firstly i initialised a TTY-Prompt
- Then i set up each menu with applicable options
- Then i added the TTY-Prompt menus into the appropriate case options in my main menu.

```

$prompt = TTY::Prompt.new
artii = Artii::Base.new

# -----
# MENUS
# -----

def main_menu
  $prompt.select('Please choose from the menu options below'.colorize(:cyan),
    ['Select Ingredients', 'Matching Recipes', 'My Recipes', 'Shopping List', 'Exit'])
end

def ingredient_sub_menu
  $prompt.select('Would you like to add or remove an ingredient from your list?'.colorize(:cyan),
    ['Add Ingredients', 'Remove Ingredients', 'Go Back'])
end

def recipe_sub_menu
  $prompt.select('Would you like to add or remove a recipe from your list?'.colorize(:cyan),
    ['Add Recipe', 'Remove Recipe', 'Go Back'])
end

def add_ingredients_sub_menu
  $prompt.select(
    'What ingredients do you currently have at home? You can add as many ingredients as you like.
    Scroll down for more options. Once you are done adding ingredients you can select "Finished Adding
    Ingredients to go back.'.colorize(:cyan), [
      $recipes.display_ingredients, 'Finished Adding Ingredients'
    ]
  )
end

def remove_ingredients_sub_menu(user_ingredients)
  $prompt.select(
    'What ingredients would you like to remove from your list? You can remove as many ingredients as
    you like'.colorize(:cyan), [
      user_ingredients, 'Finished Removing Ingredients'
    ]
  )
end

def add_recipes_sub_menu
  answer = $prompt.select(
    'What recipes would you like to add to your list? Press enter on a recipe to add it to your list.
    You can add as many recipes as you like'.colorize(:cyan), [
      $recipes.individual_recipe, 'Finished Adding Recipes'
    ]
  )
  answer.to_s
end

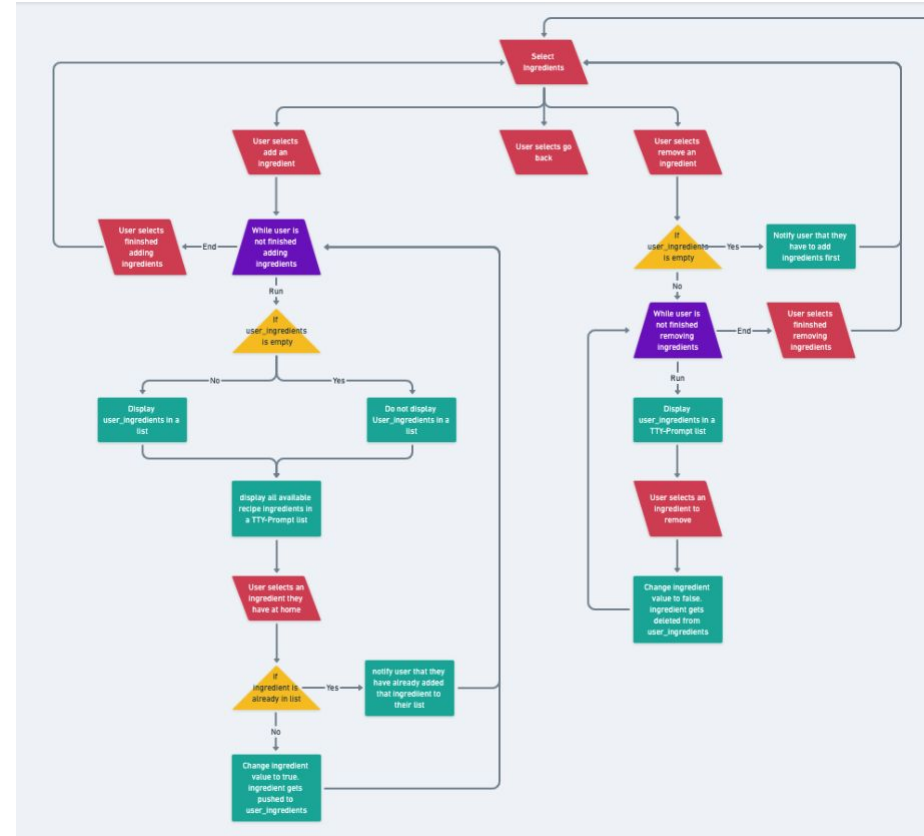
def remove_recipes_sub_menu(user_recipes)
  $prompt.select(
    'What recipes would you like to remove from your list? Press enter on a recipe to remove it from
    your list. You can remove as many recipes as you like'.colorize(:cyan), [
      user_recipes, 'Finished Removing Recipes'
    ]
  )
end

def user_recipes_sub_menu(user_recipes)
  $prompt.select(
    'What recipes would you like to remove from your list? Press enter on a recipe to remove it from
    your list. You can remove as many recipes as you like'.colorize(:cyan), [
      user_recipes, 'Finished Viewing Recipes'
    ]
  )
end

```

'Select Ingredients' Logic

- Users will be able to view all available ingredients from all available recipes.
- Once they select an ingredient it gets pushed to a new array called `user_ingredients` which is displayed above the TTY-Prompt and the `selected_ingredients` attribute gets changed to `true` in the class object.
- If a user deletes an ingredient it will get deleted from the new array and the value will be changed to `false` in the class object.
- If a user tries to add the same ingredient twice they will receive an error message stating that they have already added that ingredient.





'Select Ingredients' Code

- First i created a class function inside Recipe class to iterate through each ingredient in each recipe and print the ingredient keys to a new array
- Next i created a function to push ingredients selected to a new array. I also created the delete function as well.
- Then i added added the logic in the same function to update the value of the selected ingredient in the hash to => true by using has_key? To locate the same ingredient in the correct individual_recipe object

```
# MANUALLY TESTED - displays all ingredients from all available recipes
def display_ingredients
  available_ingredients = []
  @individual_recipe.each do |item|
    available_ingredients.push(item.ingredients.keys)
  end
  available_ingredients
end
```

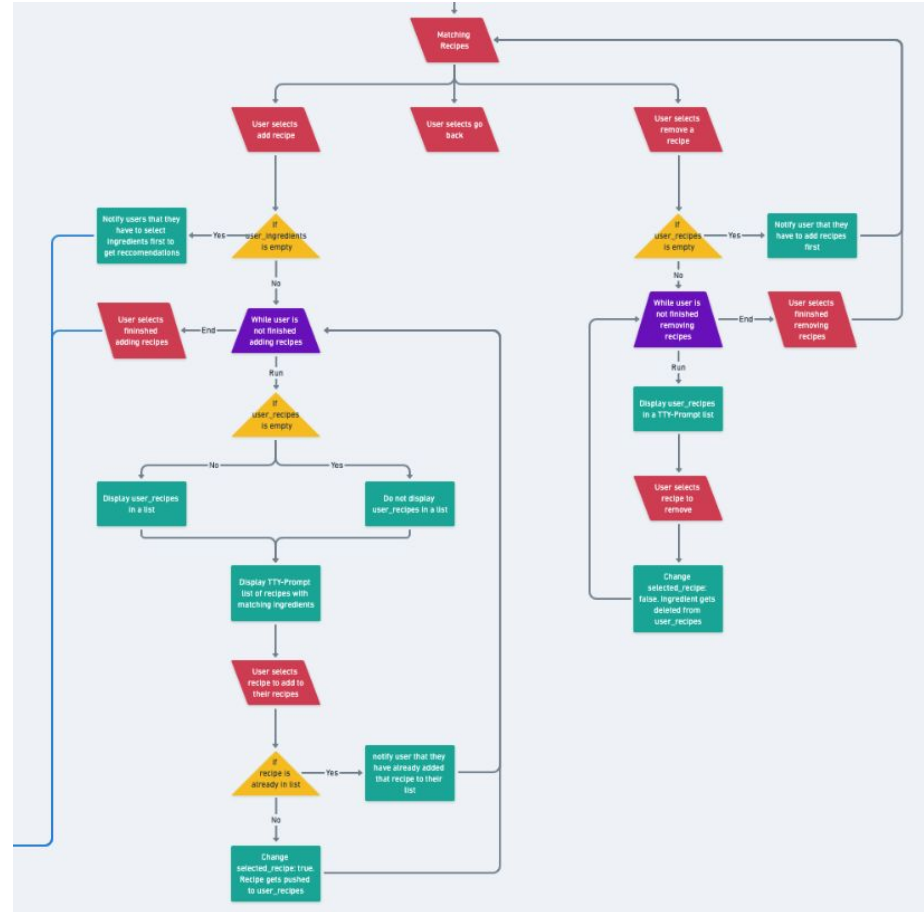
```
# -----
# SELECT INGREDIENTS FEATURE
# -----

# TESTED - adds ingredients to an array once the user selects them and updates the ingredient value to true in the individual_recipe object class
def add_ingredients(ingredient, user_ingredients)
  system 'clear'
  if ingredient != 'Finished Adding Ingredients'
    if user_ingredients.find { |item| item == ingredient }
      puts 'You have already added that ingredient, please choose a different ingredient'.colorize(:red)
    else
      # this piece of code is causing bugs need to investigate
      user_ingredients.push(ingredient)
      $recipes.individual_recipe.each_with_index do |item, _index|
        item.ingredients[ingredient] = true if item.ingredients.has_key?(ingredient)
      end
    end
  end
end

# TESTED - removes ingredients from the array once the user selects them and updates the ingredient value to false in the individual_recipe object class
def remove_ingredients(ingredient, user_ingredients)
  system 'clear'
  if ingredient != 'Finished Removing Ingredients'
    user_ingredients.delete(ingredient)
    $recipes.individual_recipe.each_with_index do |item, _index|
      item.ingredients[ingredient] = false if item.ingredients.has_key?(ingredient)
    end
  end
end
```

'Matching Recipes' Logic

- Once users have selected ingredients they will be able view a list of recipes that have the number of matching ingredients next to each recipe
- Once a recipe has been selected it gets pushed to a new recipe array and displayed above the TTY-Prompt and the selected_recipe attribute is set to true in the individual recipe class.
- Users will also be able to delete recipes from their list which will delete the recipe from the users recipes array and change the selected_recipe attribute to false.





'Matching Recipes' Code

- First i wrote a to_s function inside the individual recipe class to display the recipe name and the number of selected ingredients.
- Then i added the individual_recipe class selected_ingredients function for incrementing the selected_ingredients attribute in the same class.
- Then i wrote a function to push the recipes that the user selects into a new recipes array and set the selected_recipe attribute in the individual_recipe class object to true.
- Then i wrote the remove function which deletes the item from the array and sets the selected_recipe attribute in the class object to false.

```
def to_s
  "#{@name} (you have #{@selected_ingredients}/#{ingredients.length} ingredients)"
end

# TESTED - increments @selected_ingredients for each ingredient that has a value of true
def selected_ingredients
  @ingredients.each do |_k, v|
    # if item.has_value?(true)
    @selected_ingredients += 1 if v == true
  end
  # used below return for test case
  @selected_ingredients
end
```

```
#
# MATCHING RECIPES FEATURE
#

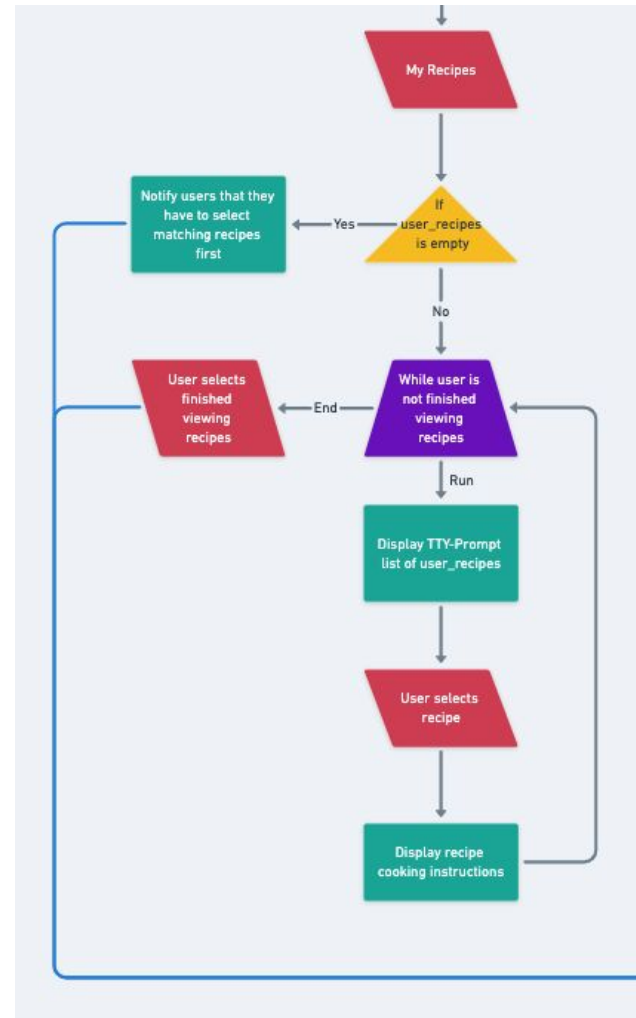
# this is not working properly as it is only incrementing over 1 recipe. now it seems to be working??
what did i change.. no idea
def matching_recipes
  $recipes.individual_recipe.each do |item|
    item.selected_ingredients
  end
end

# TESTED - adds recipes to an array once the user selects them and sets the selected_recipe attribute
to true in the individual_recipe class object
def add_recipes(recipe, user_recipes)
  system 'clear'
  if recipe != 'Finished Adding Recipes'
    if user_recipes.find { |item| item == recipe }
      puts 'You have already added that recipe, please choose a different recipe'.colorize(:red)
    else
      user_recipes.push(recipe)
      $recipes.individual_recipe.each do |item|
        item.selected_recipe = true if recipe.include? item.name
      end
    end
  end
end

# TESTED - removes recipes from an array once the user selects them and sets the selected_recipe
attribute to false in the individual_recipe class object
def remove_recipes(recipe, user_recipes)
  system 'clear'
  if recipe != 'Finished Removing Recipes'
    user_recipes.delete(recipe)
    # NEED TO TEST
    $recipes.individual_recipe.each do |item|
      item.selected_recipe = false if recipe.include? item.name
    end
  end
end
```

'My Recipes' Logic

- Once the user has added at least 1 recipe they will be able to view this list in the menu option "My Recipes".
- This list will be populated using recipes that have selected_recipe attribute set to true.
- Users will be able to select from this TTY-Prompt list of recipes to display the entire recipe and cooking instructions.





'My Recipes' Code

- First i displayed a TTY-Prompt list of the users recipes array.
- Then i wrote the individual_recipe class function print_full_recipe to print the entire recipe including cooking instructions.
- Then i wrote the function to call print_full_recipe by iterating over each individual_recipe class object using the selected user_recipe and checking if it has .include? in the individual_recipe attribute name.

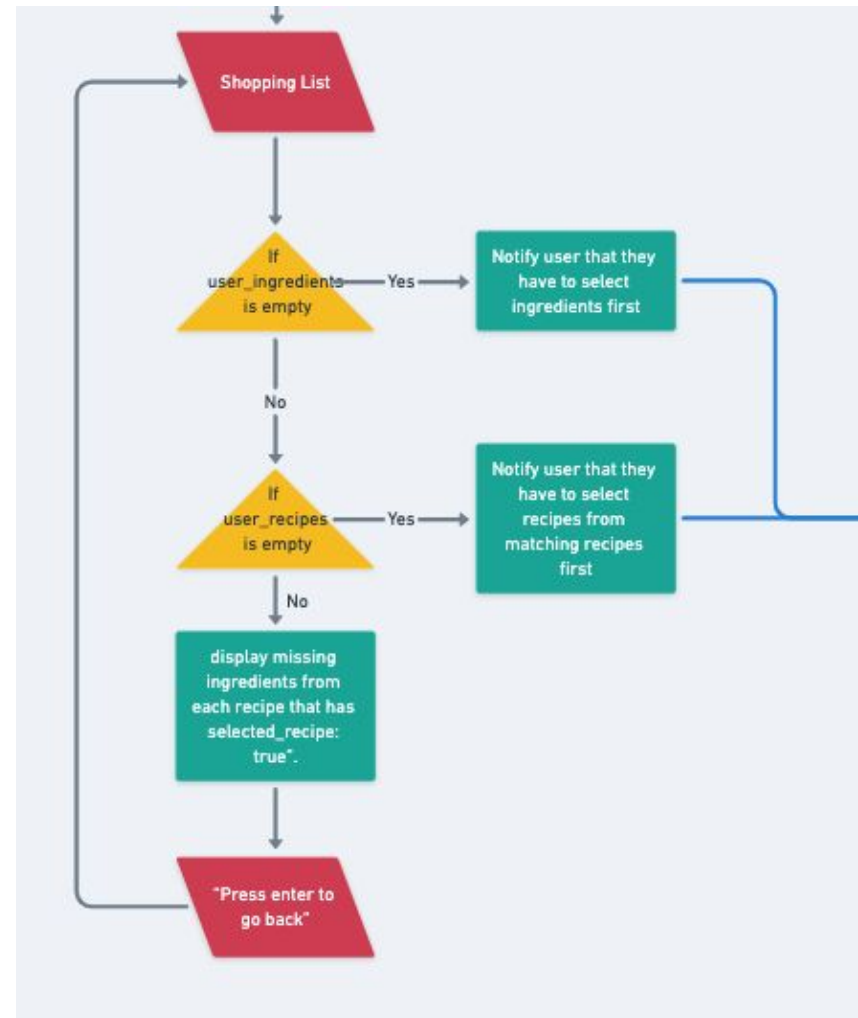
```
# MANUALLY TESTED - prints the full recipe method
def print_full_recipe
  @name.to_s
  puts 'Ingredients:'.colorize(:cyan)
  (@ingredients.each { |k, _v| puts k }).to_s
  puts 'Instructions:'.colorize(:cyan)
  (@instructions.each { |i| puts i }).to_s
  "Serves: #{@serves}"
end
end
```


```
#
# MY RECIPES FEATURE
#

# TESTED - prints full recipe method in a list to the screen
def display_entire_recipe(recipe, user_recipes)
  system 'clear'
  if recipe != 'Finished Viewing Recipes' && user_recipes.find { |item| item == recipe }
    $recipes.individual_recipe.each do |item|
      item.print_full_recipe if recipe.include? item.name
    end
  end
end
end
```


'Shopping List' Logic

- If user has added ingredients AND recipes then they can view a shopping list of ingredients that they need to buy for the recipes that they selected.
- This is done only showing ingredients that have not been selected (ingredient value is false) from the recipes that have been selected (selected_recipe is true) by iterating over a list of user recipes.





'Shopping List' Code

- First i added the class function selected recipes to return a new array of all recipe ingredients that had selected_recipe set to true.
- Then i passed those new ingredients into the function display missing recipes.
- Then i wrote the class function display missing ingredients to only display the ingredients that had a value of false.

```
# TESTED - finds the recipes that the user has selected based on the selected_recipe attribute being
set to true and returns an array of recipe ingredients
def selected_recipes
  selected_recipes = []
  @individual_recipe.each do |i|
    selected_recipes.push(i.ingredients.each { |k, _v| k }) if i.selected_recipe == true
  end
  selected_recipes
end

# TESTED - displays all of the ingredients that have their values set to false (meaning the user has
not selected those ingredients)
def display_missing_ingredients(ingredients)
  unselected_ingredients = []
  missing_ingredients = []
  ingredients.each do |item|
    unselected_ingredients.push(item.select { |_k, v| v == false })
  end
  missing_ingredients.push(unselected_ingredients.each { |i| puts i.keys })
  missing_ingredients
end
```

Challenges, Ethical Issues & Favourite Parts



Challenges

- Juggling work and family commitments.
- Fell behind in learning activities due to the workbook and spent a fair bit of time googling.
- Visualising the correct logic from the start so that i knew exactly how to code it. I spent a lot of time writing functions only to realise a day later i wouldn't even need the function as i could achieve the same thing another way(which was often simpler). If i had my control flow and logic correct from the start it would have saved me a lot of time and pain.
- This lead to me spending a lot of time stuck in a loop jumping from one function to the next and it felt like every time i fixed one problem it broke something else which was alarmingly stressful.
- I had trouble accessing my nested arrays and hashes inside my nested class objects.



Ethical Issues

- I made sure that any gems or other material that i used would be linked in my attribution section in my blog post and in my README file.
- This slide deck was created using the balsamic template in the template gallery in google slides



Favourite Parts

- Writing rspec tests that actually worked made it a lot easier implement the rest of the code. Testing saved this project in the end.
- When a new function worked on the first go (very rare).
- Towards the end when the app was almost complete and i realised that i might actually get it done in time.
- Running Rubocop autocorrect on my code taught me a few tricks.
- Closing 30 odd stack overflow tabs after a week of troubleshooting.

Questions?
