Terminal App Documentation

Software Development Plan

Application Function:

With the recent and rapid spread of viruses like COVID, Ebola, and Swine Flu. Businesses need to be able to adapt quickly and easily to the new world. The Ruby Terminal Ordering App will allow restaurants and bars to open their doors to customers and let them order without the need for human interaction. The reasons for developing this app are two-fold, firstly to learn more about how the Ruby programming language works, secondly to become a millionaire after graduating by selling this amazingly sophisticated application to every restaurant in the (possibly) post-apocalyptic world.

My target audience will be for the cautious patron who loves going out for a meal and drinks, but hates getting Covid or other highly infectious diseases.

The Ruby Terminal Ordering App will allow customers in restaurants and bars to order their food and drink without requiring waitstaff or a QR code. Once the Ordering App has started, the customer is greeted with an artistic 'welcome' message to make them feel comfortable from the start. Below the welcome message, a highly-customisable menu is displayed with the items available to order on the left, and the prices on the right. The customer is then prompted to input their order and quantity, in the result of a larger group, the Ordering App can handle multiple orders without having to restart each time. Once the table has ordered everything they would like, a simple input of ‘done’ is all that is required to send their order to the kitchen. They are then rewarded for using the Ordering App by receiving a confirmation of their order together with the total price owed, an artistic ‘goodbye’ message, and a request for a quick survey of their experience. The restaurant or bar management also receive the total amount of each order for cash-up at the end of the day, and the results of the survey, in a separate text file.

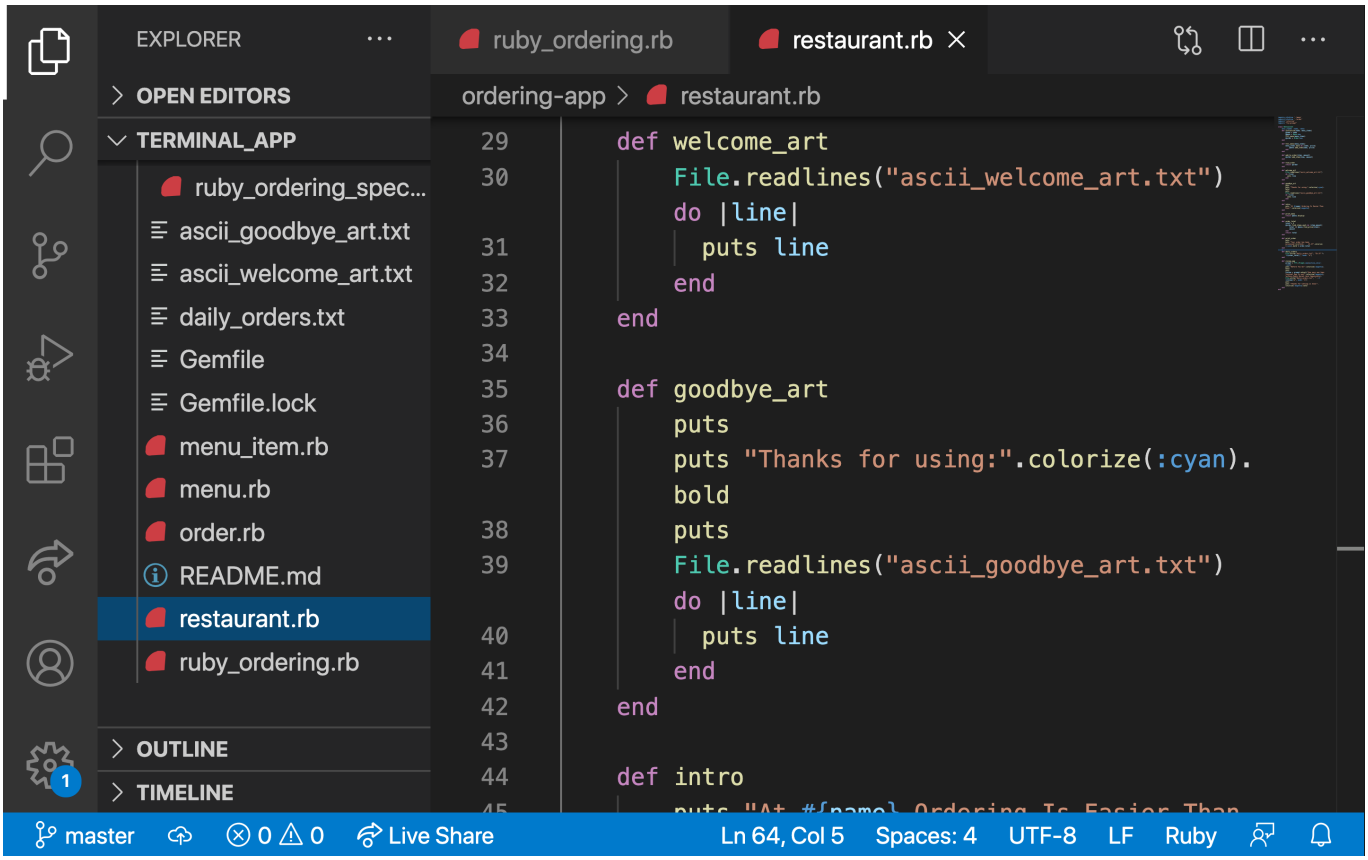
To clarify, the features of the app will involve:

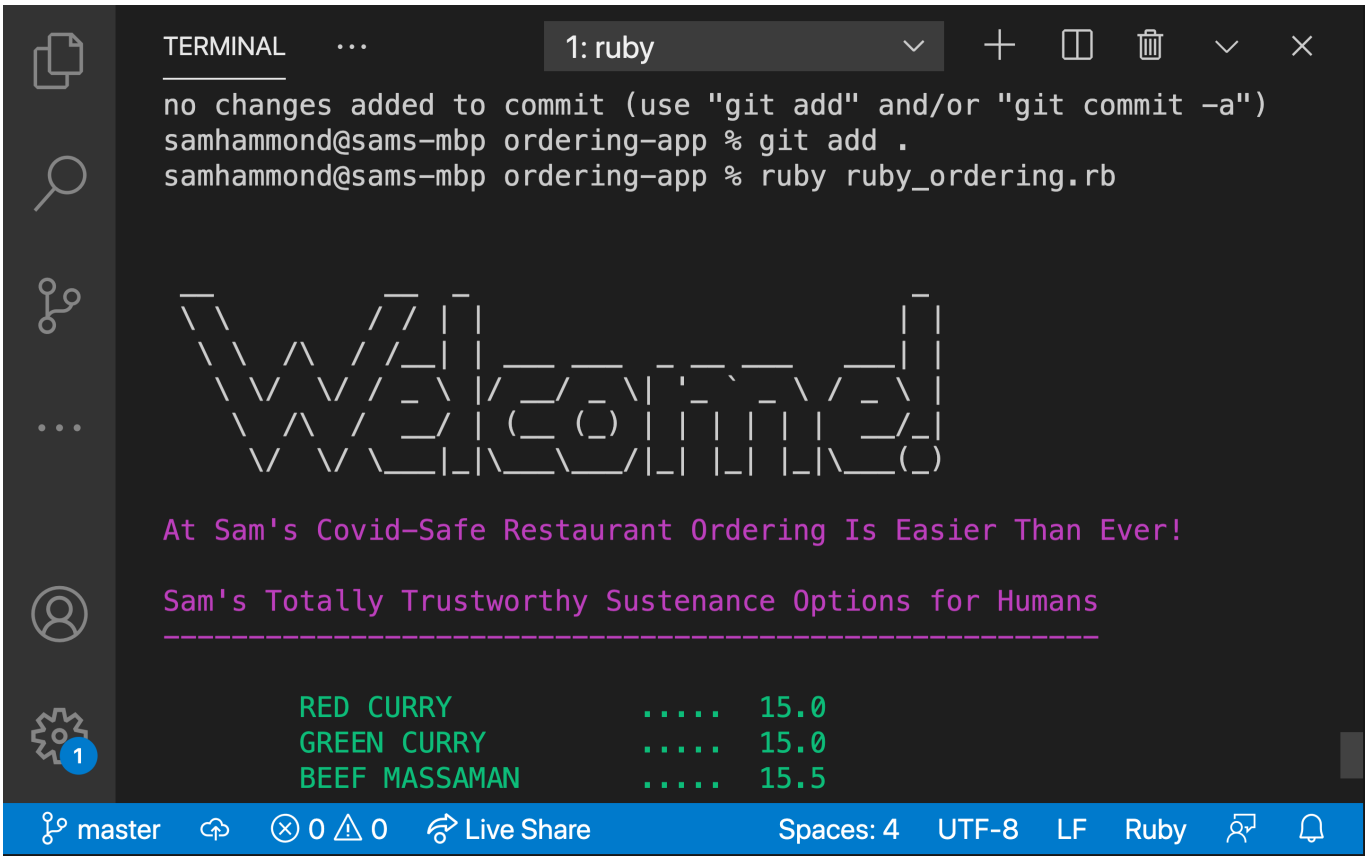
- A ‘welcome’ and ‘goodbye’ message using ASCII art,
- A loop to continue taking orders until all customers have finished including quantities,
- A highly-customisable menu and code to allow quick implementation for a large number of businesses,
- A receipt with the total calculated and printed,
- A survey that uses TTY prompt to simplify the experience (making it more likely for the customer to leave a review)
- Creating txt files for kitchen orders, total amounts, and survey results.

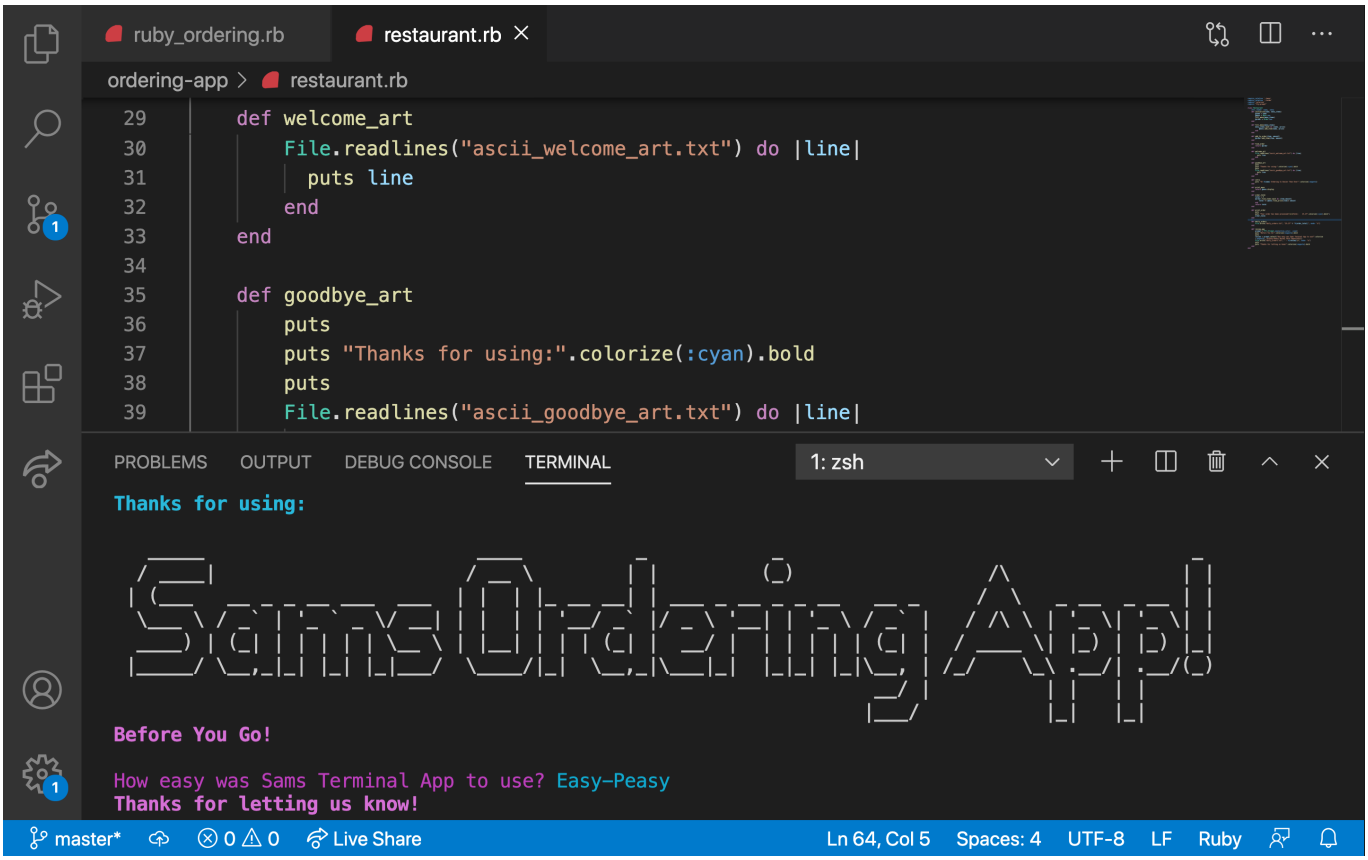
Features

Welcome & Goodbye Art:

The Ordering App will display Ascii art which will be developed using the Ruby Gem ‘Artii’. Initially I was planning on using a simple ‘puts’ string however, I didn’t like how messy it made my code look so I will add the art to a .txt file and call it using a block inside a method.

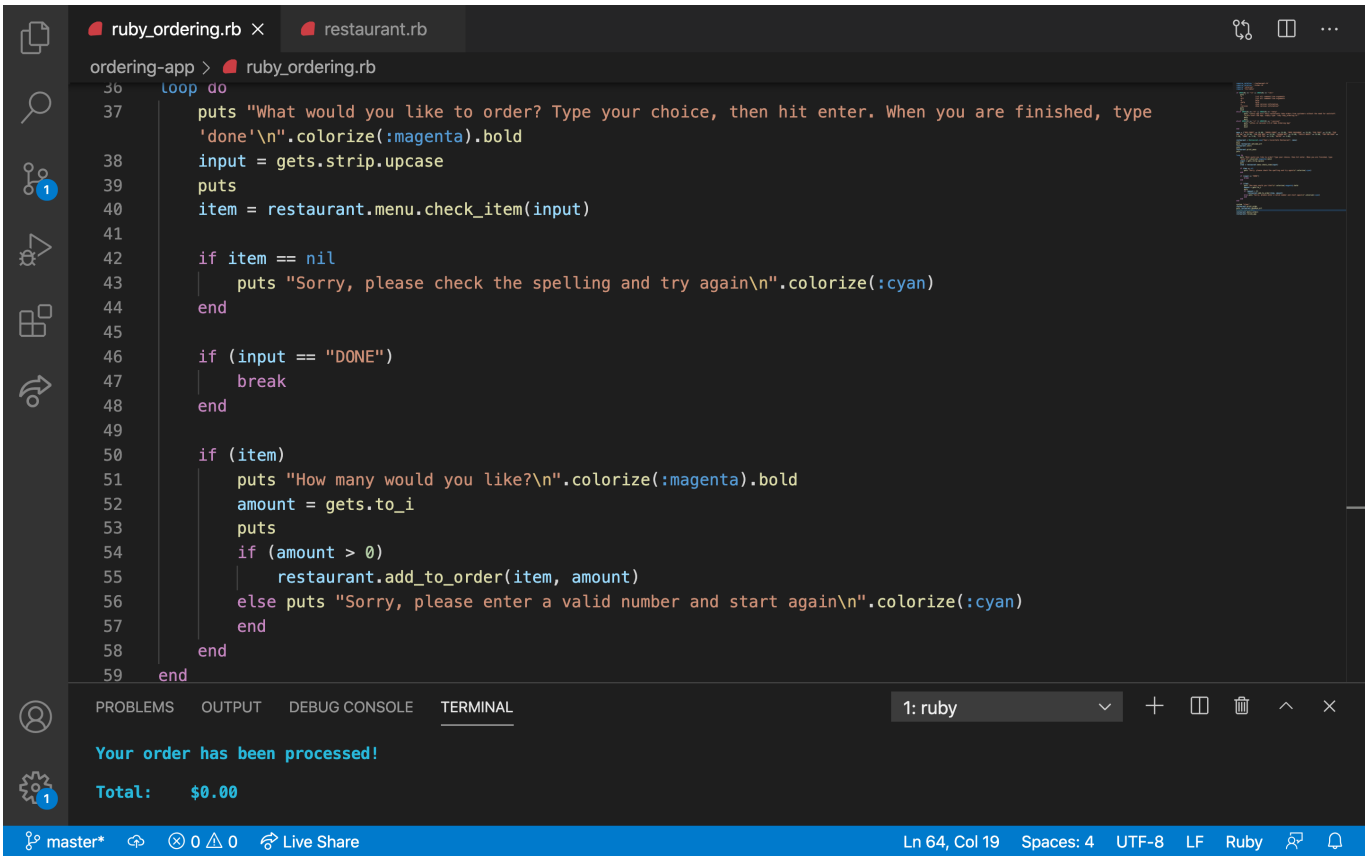

EXPLORER
ruby_ordering.rb
restaurant.rb
OPEN EDITORS
TERMINAL_APP
ordering-app > restaurant.rb
ruby_ordering_spec...
ascii_goodbye_art.txt
ascii_welcome_art.txt
daily_orders.txt
Gemfile
Gemfile.lock
menu_item.rb
menu.rb
order.rb
README.md
restaurant.rb
ruby_ordering.rb
OUTLINE
TIMELINE
29 def welcome_art
30 File.readlines("ascii_welcome_art.txt")
do |line|
31 puts line
32 end
33 end
34
35 def goodbye_art
36 puts
37 puts "Thanks for using:".colorize(:cyan).
bold
38 puts
39 File.readlines("ascii_goodbye_art.txt")
do |line|
40 puts line
41 end
42 end
43
44 def intro
master
0 0
Live Share
Ln 64, Col 5
Spaces: 4
UTF-8
LF
Ruby

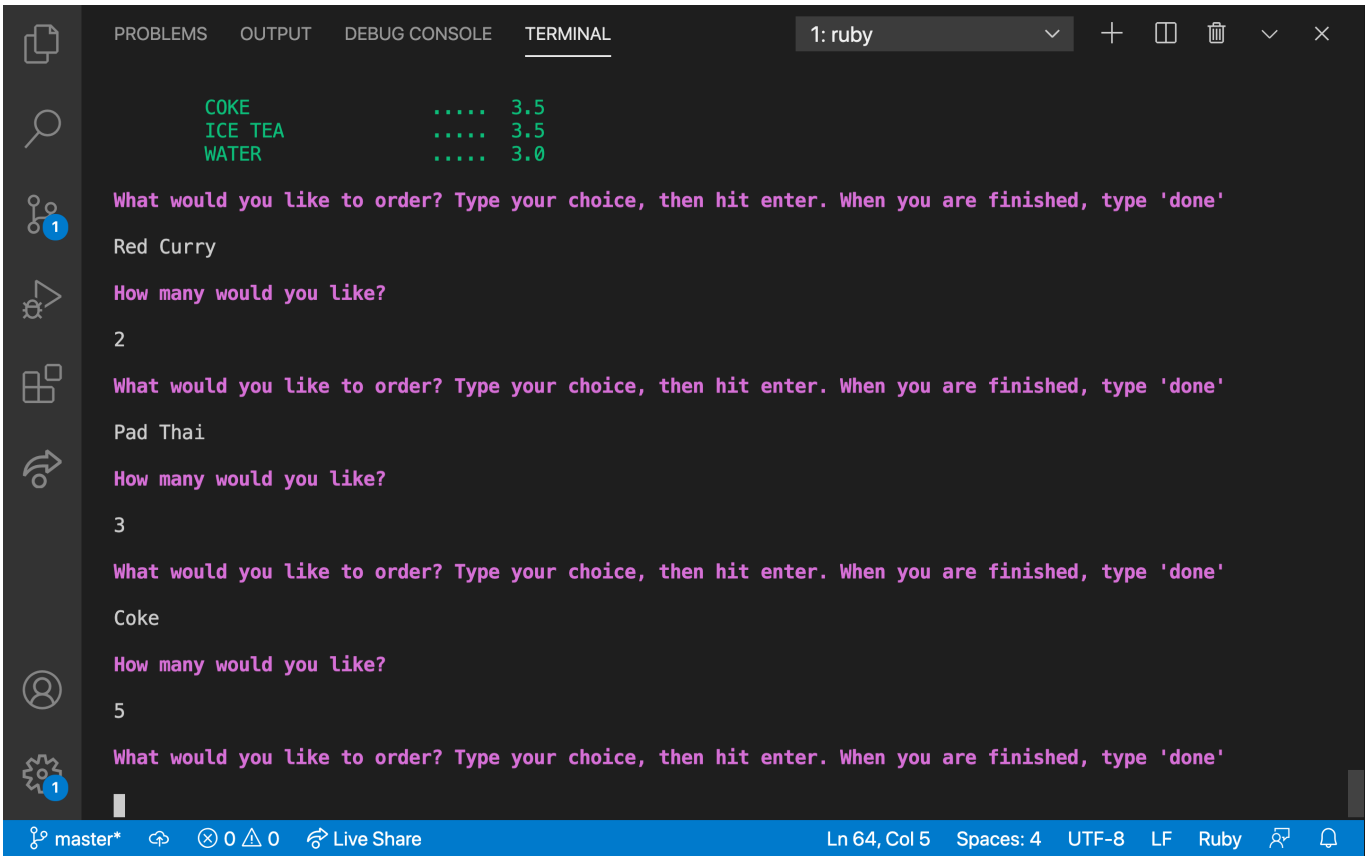

TERMINAL
1: ruby
no changes added to commit (use "git add" and/or "git commit -a")
samhammond@sams-mbp ordering-app % git add .
samhammond@sams-mbp ordering-app % ruby ruby_ordering.rb
At Sam's Covid-Safe Restaurant Ordering Is Easier Than Ever!
Sam's Totally Trustworthy Sustenance Options for Humans
RED CURRY 15.0
GREEN CURRY 15.0
BEEF MASSAMAN 15.5
master
0 0
Live Share
Spaces: 4
UTF-8
LF
Ruby



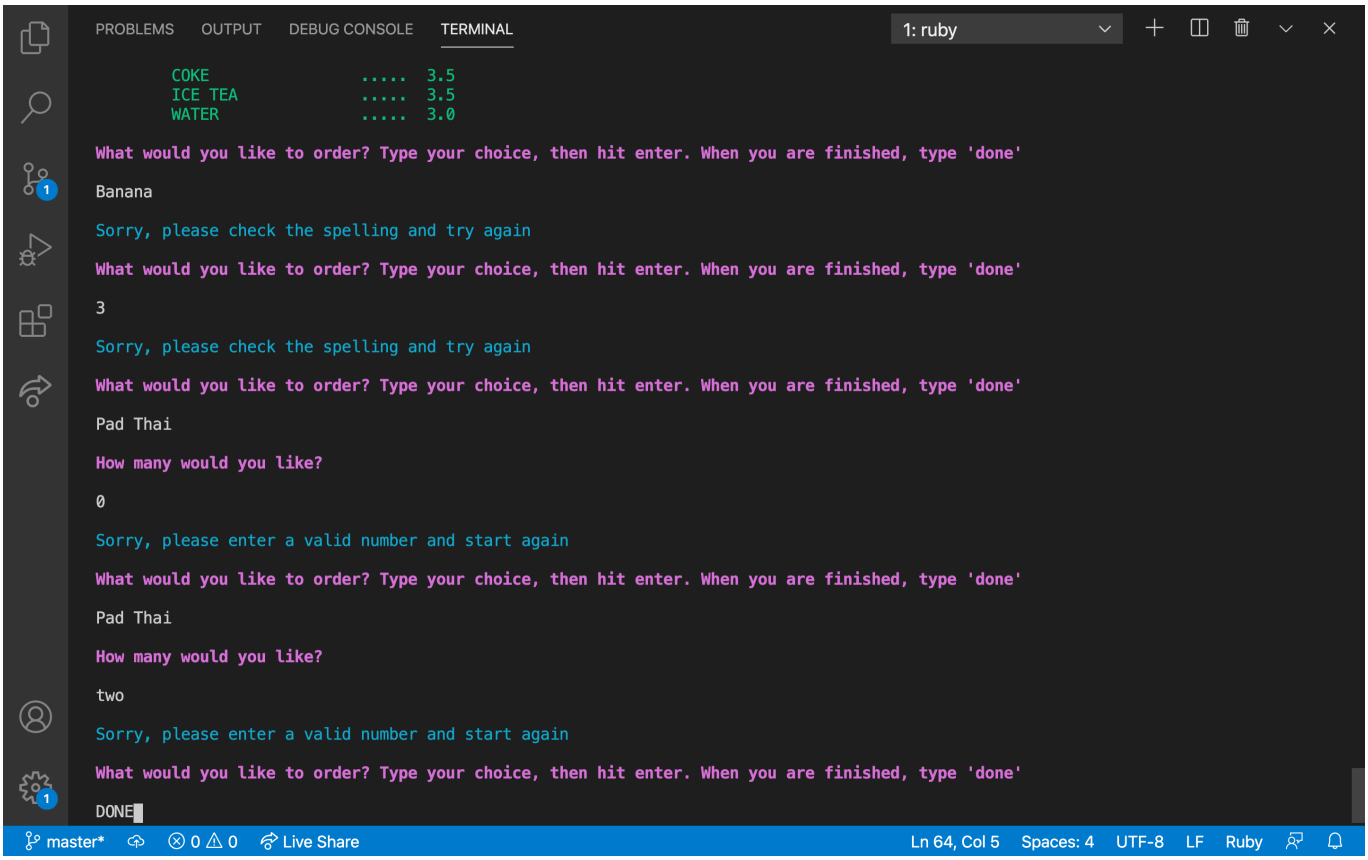
Continue Taking Orders & Quantities Until Everyone Has Ordered:

The Ordering App will continue asking for orders until the customer has typed 'done'. The app will first ask for you to enter what you would like to order, the output has specific instructions on how to do this. It will then verify that the restaurant has that item on the menu before asking how many they would like to order. If the input is a valid number it will add the item with its quantity to the order before asking you for the next item. Once you type 'done' the loop will break.



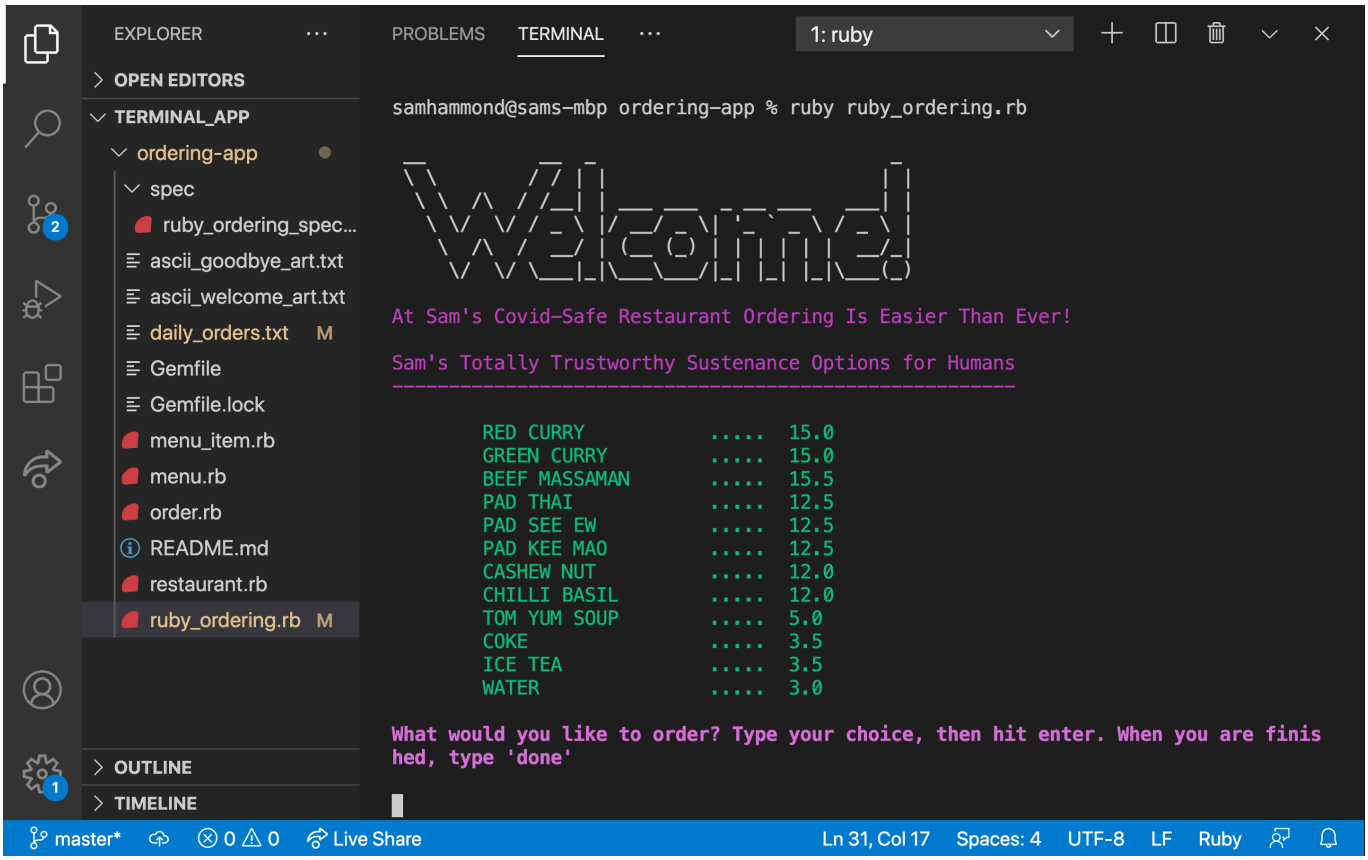


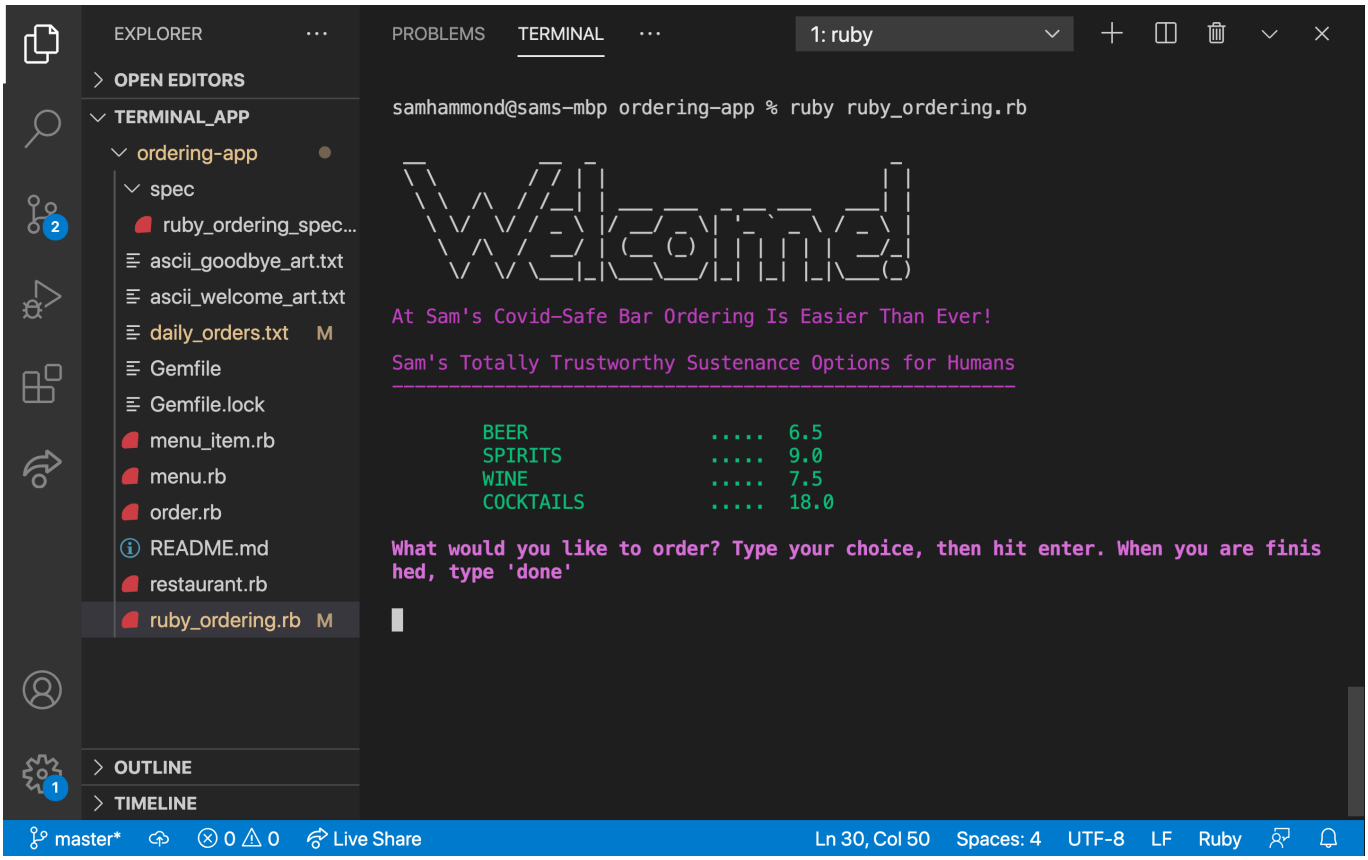
To address error handling, friendly instructions are given when the wrong input is received and the loop will start again.

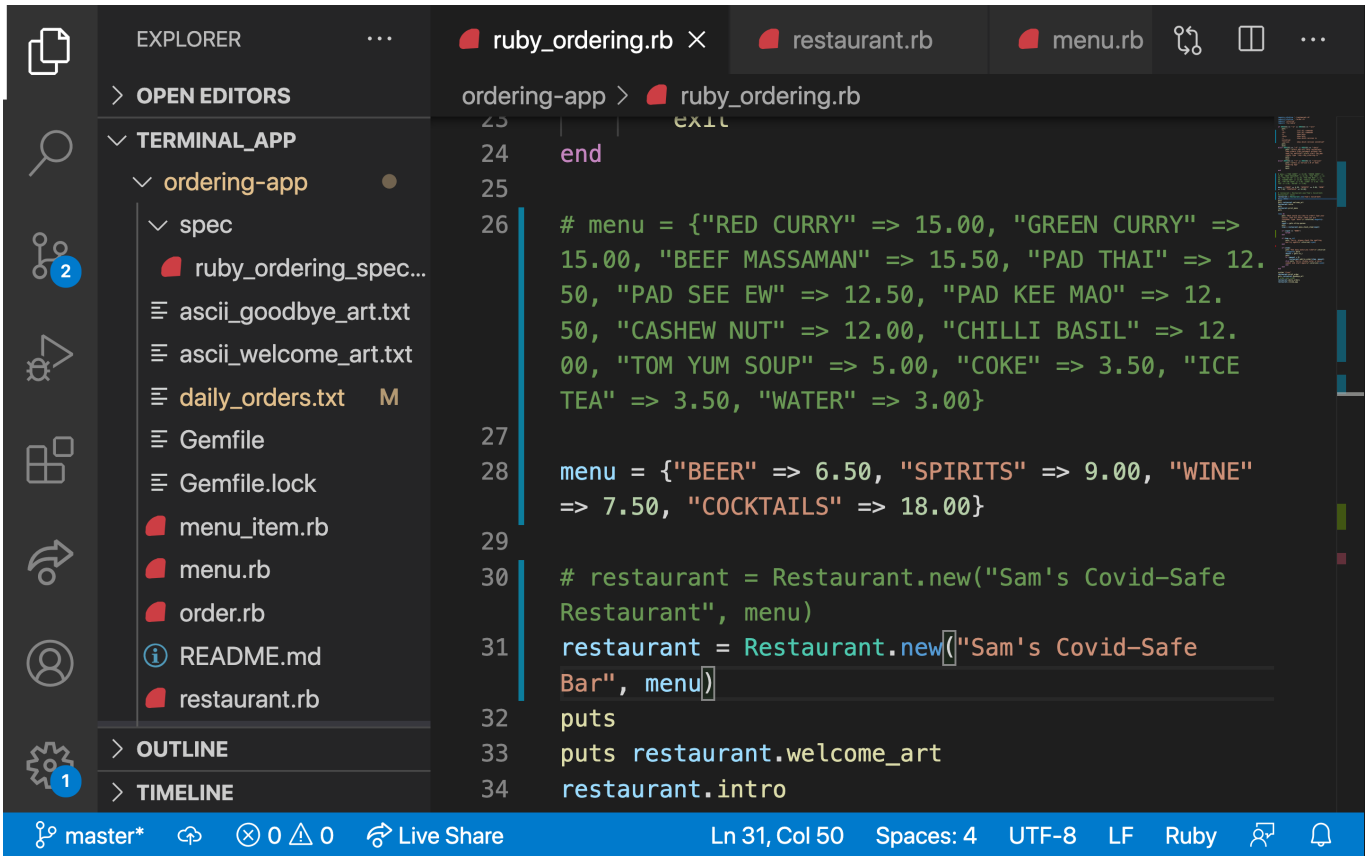


A highly-customisable menu and code to allow quick implementation for a large number of businesses:

I will separate the code into classes to help quick installation over a variety of businesses. Menu, Restaurant, Order & MenuItem won't require any change to the code. When installing the Ordering App in a new restaurant it will only take a minute or two to set-up by simply opening the main.rb file and changing the name of the restaurant, and replacing the items and prices in the 'menu'. Your new restaurant is now ready to go!

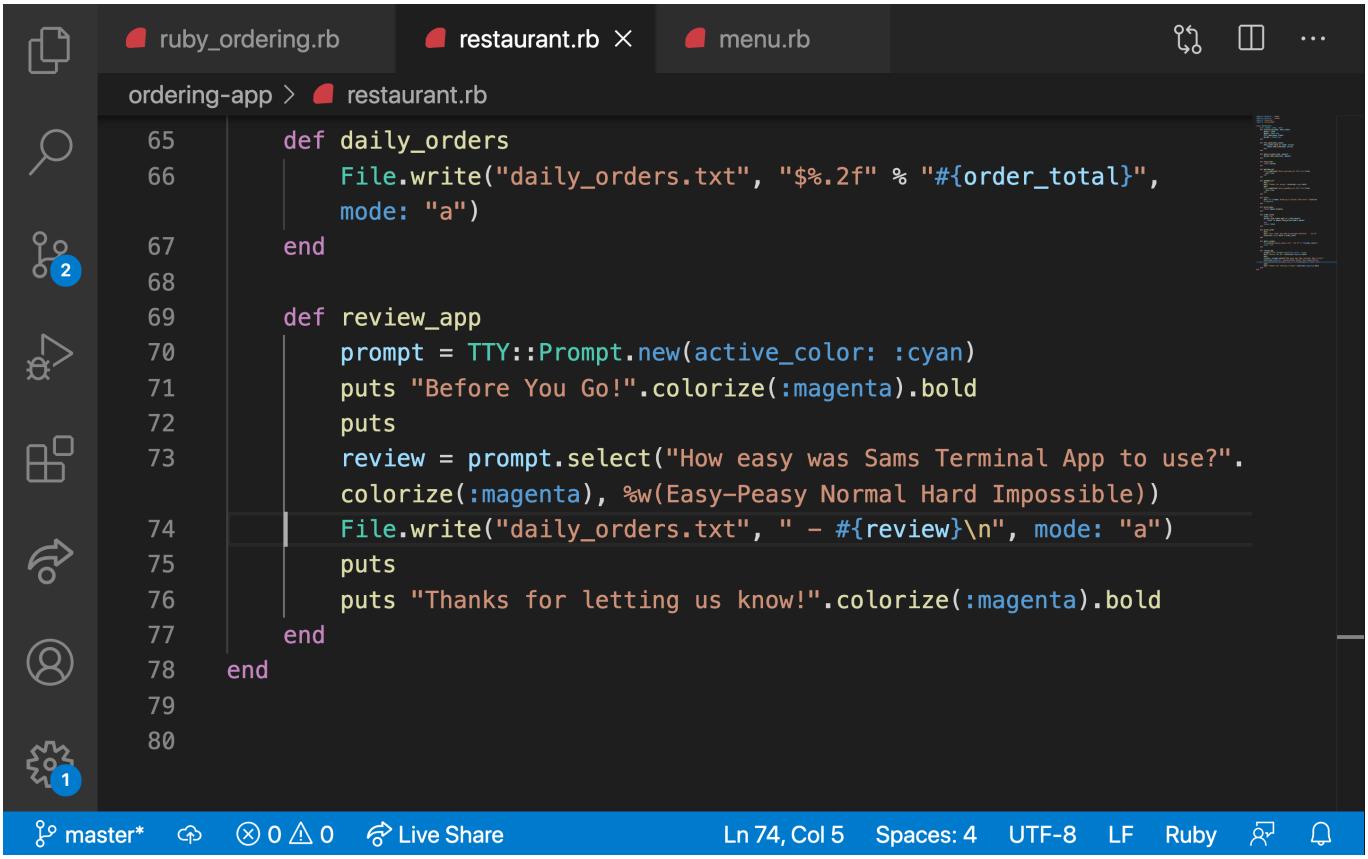

EXPLORER
OPEN EDITORS
TERMINAL_APP
ordering-app
spec
ruby_ordering_spec...
ascii_goodbye_art.txt
ascii_welcome_art.txt
daily_orders.txt M
Gemfile
Gemfile.lock
menu_item.rb
menu.rb
order.rb
README.md
restaurant.rb
ruby_ordering.rb M
OUTLINE
TIMELINE
PROBLEMS
TERMINAL
1: ruby
samhammond@sams-mbp ordering-app % ruby ruby_ordering.rb
At Sam's Covid-Safe Restaurant Ordering Is Easier Than Ever!
Sam's Totally Trustworthy Sustenance Options for Humans
RED CURRY 15.0
GREEN CURRY 15.0
BEEF MASSAMAN 15.5
PAD THAI 12.5
PAD SEE EW 12.5
PAD KEE MAO 12.5
CASHEW NUT 12.0
CHILLI BASIL 12.0
TOM YUM SOUP 5.0
COKE 3.5
ICE TEA 3.5
WATER 3.0
What would you like to order? Type your choice, then hit enter. When you are finis
hed, type 'done'
master*
Live Share
Ln 31, Col 17
Spaces: 4
UTF-8
LF
Ruby

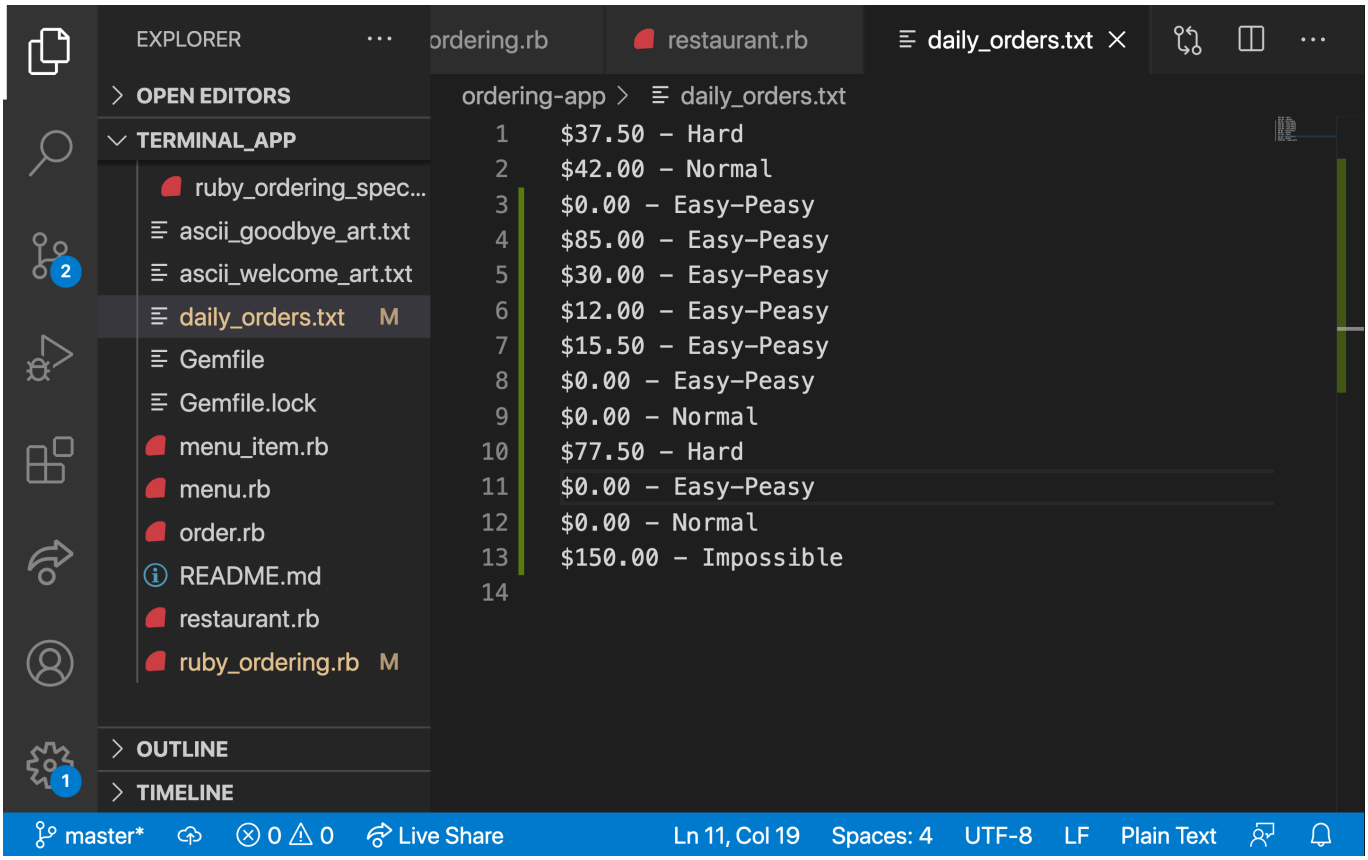

EXPLORER
OPEN EDITORS
TERMINAL_APP
ordering-app
spec
ruby_ordering_spec...
ascii_goodbye_art.txt
ascii_welcome_art.txt
daily_orders.txt M
Gemfile
Gemfile.lock
menu_item.rb
menu.rb
order.rb
README.md
restaurant.rb
ruby_ordering.rb M
OUTLINE
TIMELINE
PROBLEMS
TERMINAL
1: ruby
samhammond@sams-mbp ordering-app % ruby ruby_ordering.rb
At Sam's Covid-Safe Bar Ordering Is Easier Than Ever!
Sam's Totally Trustworthy Sustenance Options for Humans
BEER 6.5
SPIRITS 9.0
WINE 7.5
COCKTAILS 18.0
What would you like to order? Type your choice, then hit enter. When you are finis
hed, type 'done'
master*
Live Share
Ln 30, Col 50
Spaces: 4
UTF-8
LF
Ruby

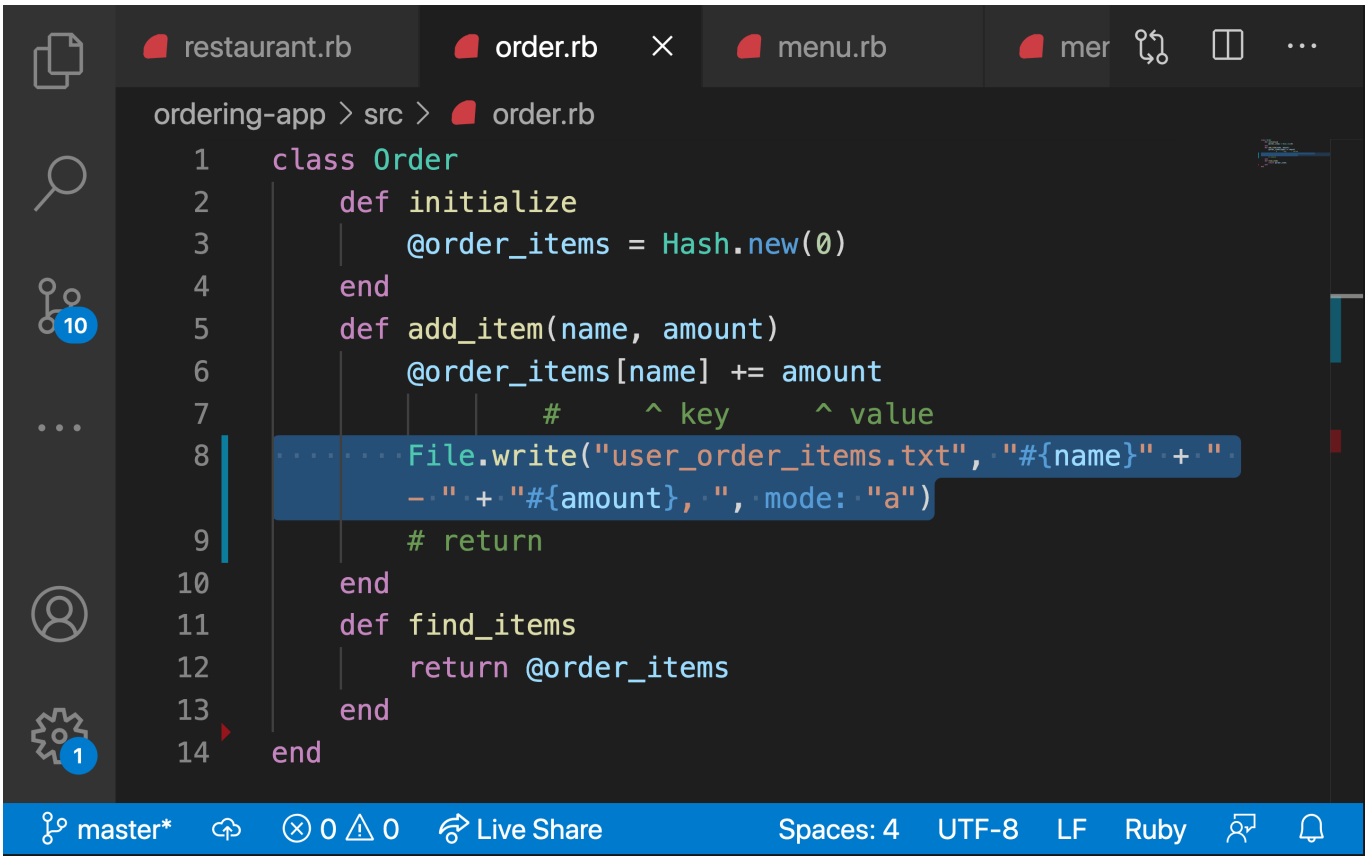


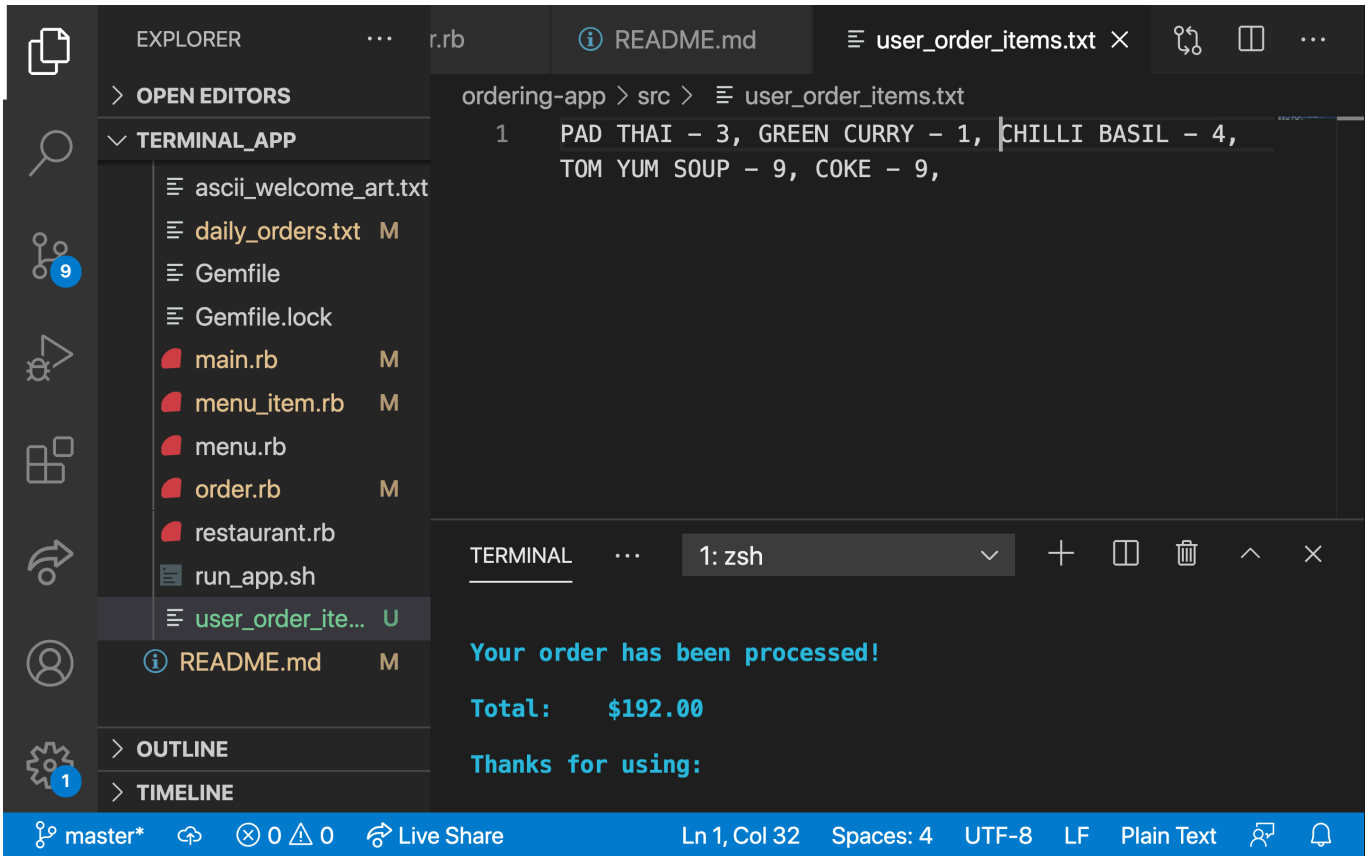
Creating txt files for kitchen orders, total amounts, and survey results:

.txt files will be created for the orders, cash sales, and survey. This will help management keep track of the daily sales and help me refine later versions with the feedback given in the survey. To do this for the total amount & survey I will define a method that utilises the 'File' class in Ruby and call it towards the end of the program. Orders and amount will go to the kitchen as soon as the customer orders the item.



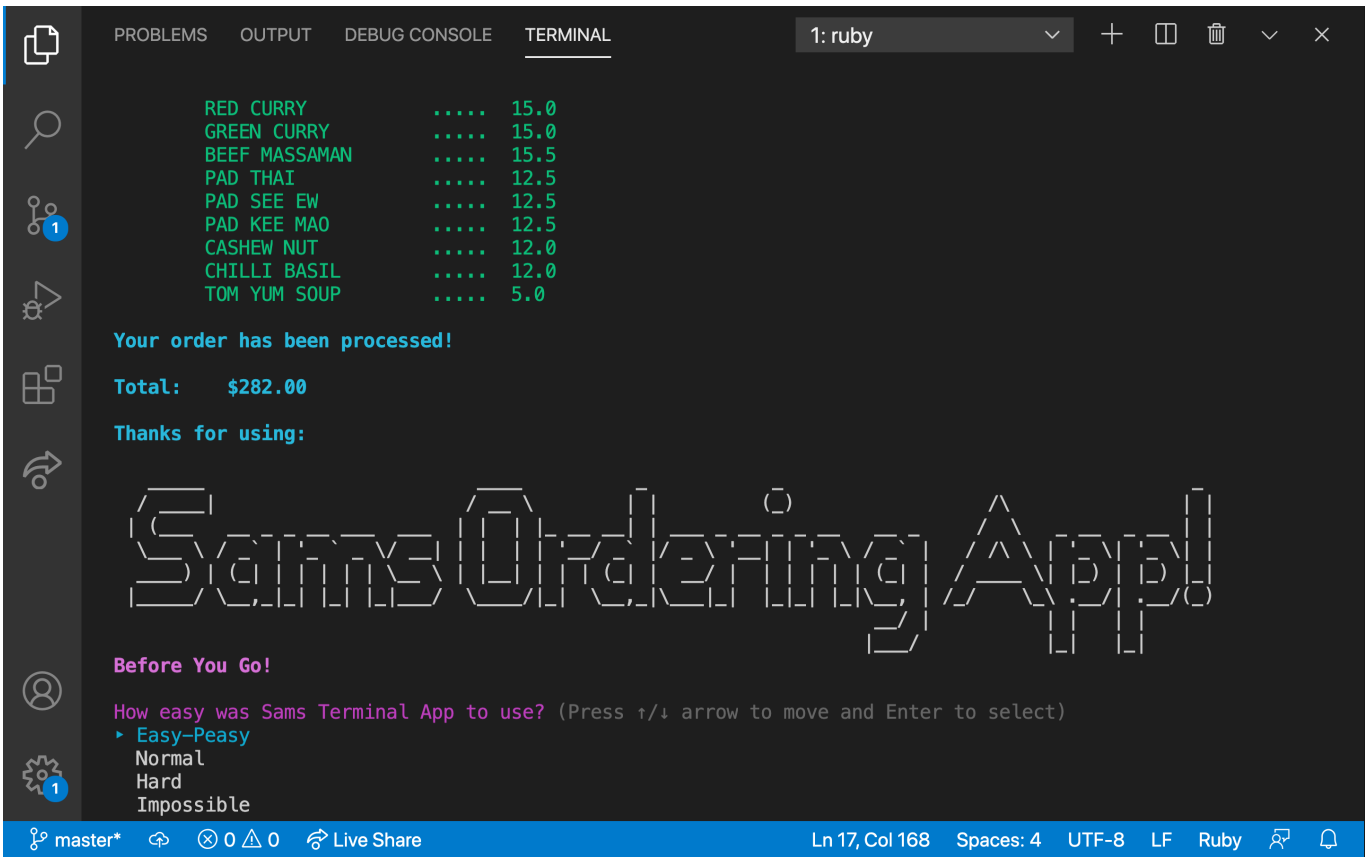

EXPLORER
ordering.rb
restaurant.rb
daily_orders.txt
OPEN EDITORS
TERMINAL_APP
ruby_ordering_spec...
ascii_goodbye_art.txt
ascii_welcome_art.txt
daily_orders.txt M
Gemfile
Gemfile.lock
menu_item.rb
menu.rb
order.rb
README.md
restaurant.rb
ruby_ordering.rb M
OUTLINE
TIMELINE
ordering-app > daily_orders.txt
$37.50 - Hard
$42.00 - Normal
$0.00 - Easy-Peasy
$85.00 - Easy-Peasy
$30.00 - Easy-Peasy
$12.00 - Easy-Peasy
$15.50 - Easy-Peasy
$0.00 - Easy-Peasy
$0.00 - Normal
$77.50 - Hard
$0.00 - Easy-Peasy
$0.00 - Normal
$150.00 - Impossible
master* 0 0 Live Share Ln 11, Col 19 Spaces: 4 UTF-8 LF Plain Text

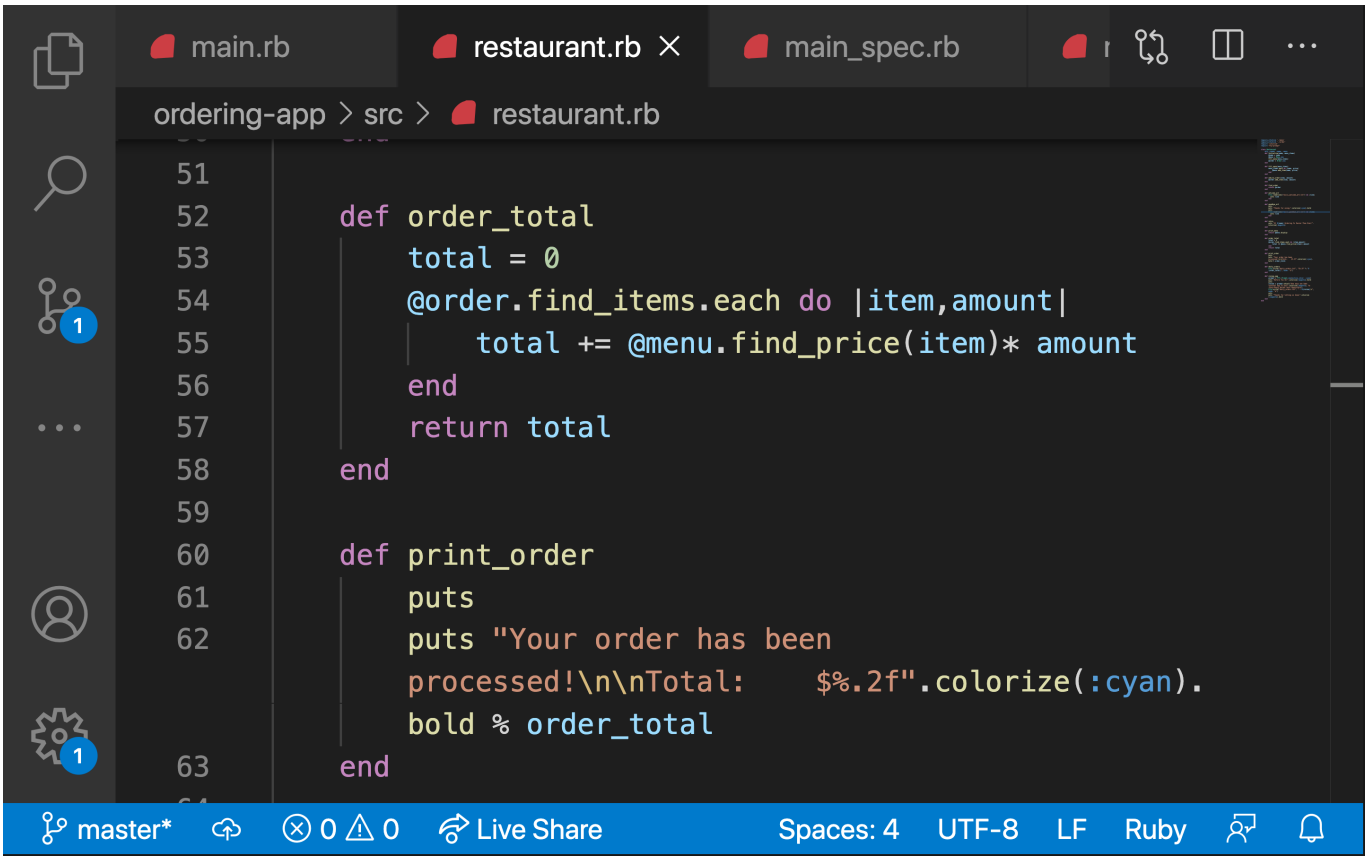

restaurant.rb
order.rb
menu.rb
mer
ordering-app > src > order.rb
class Order
def initialize
@order_items = Hash.new(0)
end
def add_item(name, amount)
@order_items[name] += amount
# ^ key ^ value
File.write("user_order_items.txt", "#{name}" + " - " + "#{amount}, ", mode: "a")
# return
end
def find_items
return @order_items
end
end
master* 0 0 Live Share Spaces: 4 UTF-8 LF Ruby



Total calculated and printed on receipt:

The item price will be multiplied by the quantity and then added to the total. The bill will then be printed-out once the user types '*done*'.





Outline of User Interaction & Experience:

There will be two different experiences which depends on whether the user is a customer or an employee.

Customer User:

The customer will be greeted with a welcome message that is more than just a title, followed by a print-out of the menu available to them. They will find out how to interact with the app by following the simplified instructions that prints-out underneath the menu. If the user follows the first instruction correctly, they will input their order into the command line and push 'enter'. The same method is then used to get how many of that item they would like. The customer will keep interacting with the loop feature until they complete their order, once complete, the simple instructions make it clear to type 'done'. This will clear the screen and print the receipt, with some art created with a Ruby gem. The customer will then be asked to answer a one-question survey with a list of 4 options they can use the arrow keys to cycle through if they have the time.

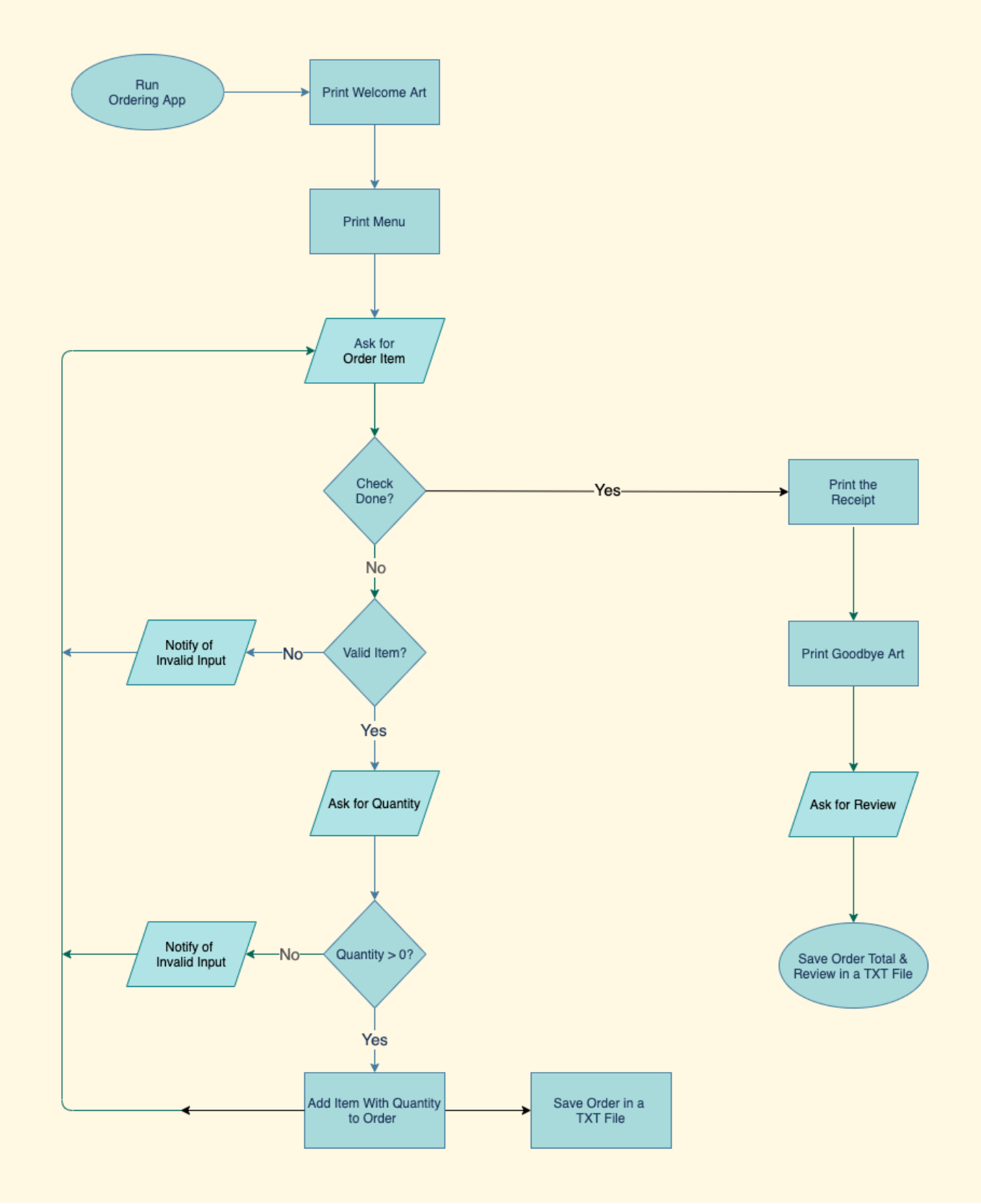
If the customer ever types an invalid input they will receive a friendly message asking them to please try again. This includes ordering '0' of something.

The customer can then leave the restaurant and head to the bar on the corner, which will be interacted with in the same way as the restaurant. Only the items to order and name showcased will be different.

Employee User:

An employee will be able to interact with customer orders, amount sold in the day so far, and the survey answer by opening the .txt files in the app.

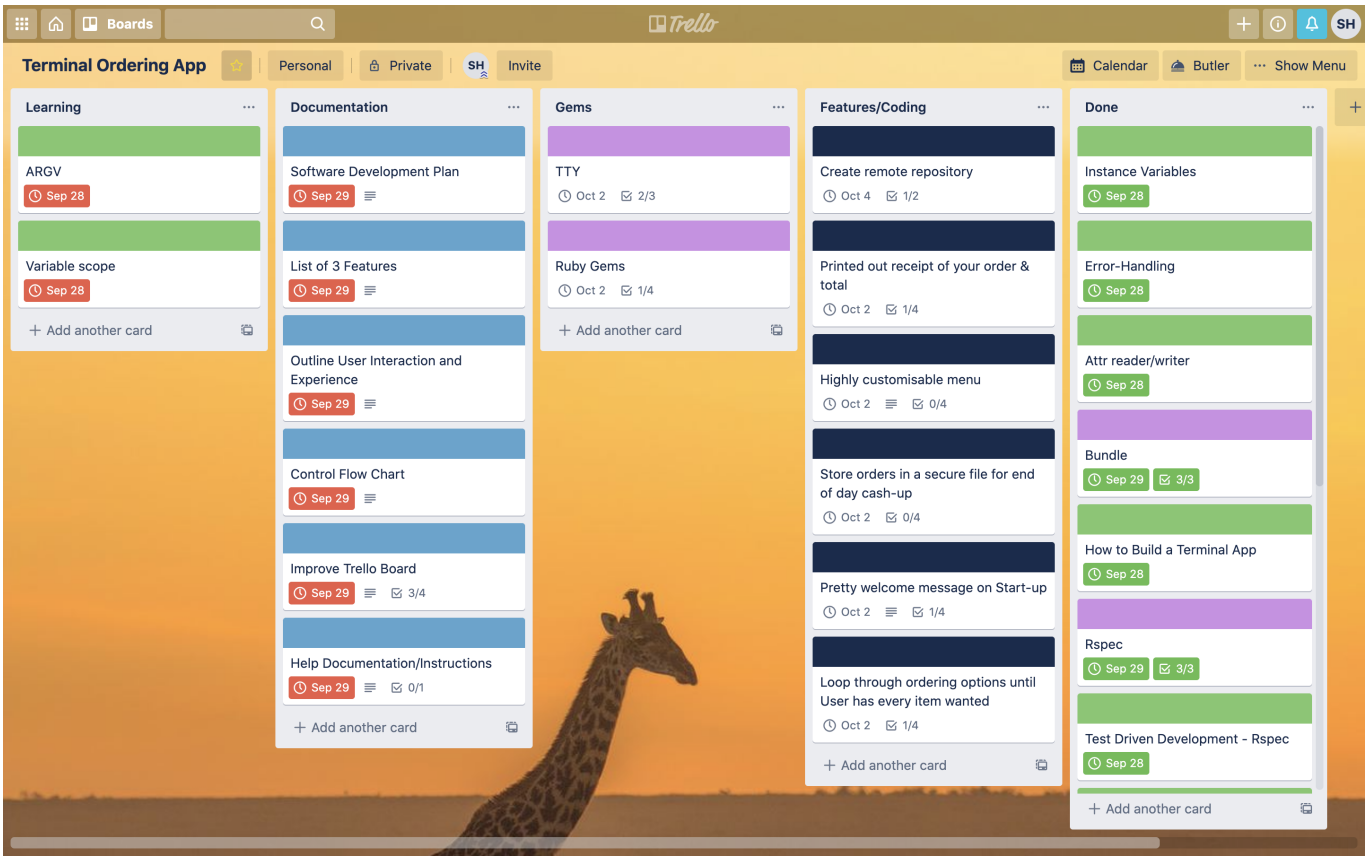
Control Flow Chart:



Link to My Implementation Plan

Trello link

I regret not taking a screenshot when I first created my plan, but below is a screenshot of where I was up to with 5 days left until my deadline:



Instructions for Installation and Use

- Once published, you can clone or download the Ordering-App from my Github page
- You will need ruby installed on your computer. You can download it by clicking this link
- You will also need the Bundler gem installed. It should come installed with Ruby but you can also install it with the following input into your terminal: *gem install bundler*
- To install the gems, go to the src location of the app on your terminal and use the command: *bundle install*
- Run the app from the same location with the bash script: *./run_app.sh*
- The app should run on all 21st Century computers
- For more information about the app before running it type *ruby main.rb* followed by:
    - -a or -all ... list all commands
    - -h or -help ... show help
    - -v or -version ... show which version is installed

Link to my slide deck for the W.I.P presentation as the videos won't work in a pdf:

LINK


COVID-19 PANDEMIC
Conspiracy theorists burn 5G towers claiming link to virus
Global
NATIONAL