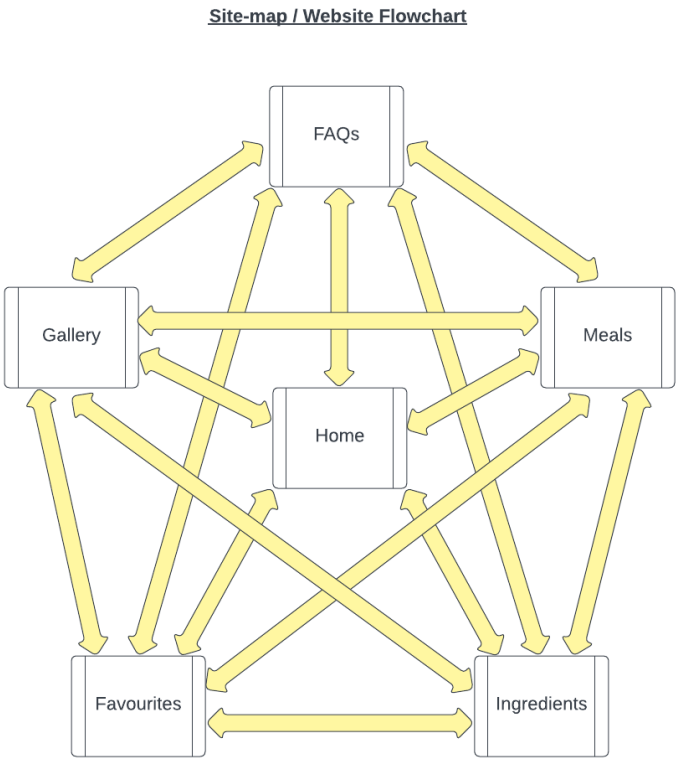
Website Planning Document

Introduction:

The link to the website can be found here: https://pythonmodule.onrender.com/home

The link to the github for the project can be found here: https://github.com/djonskanlyn/PythonModule

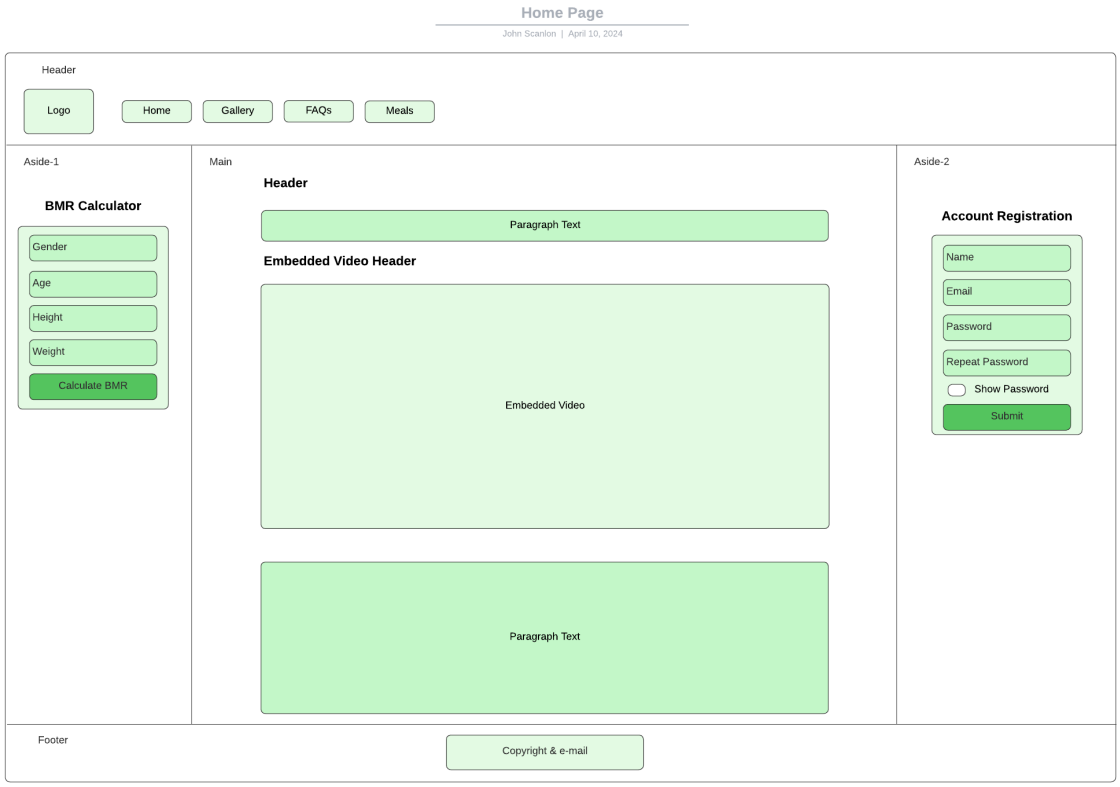
The website has a health and fitness theme and has 6 web pages.



The general page structure is the same for all 6 pages. Each page is split into 5 sections (header, main, aside-1, aside-2, and footer) using flex-box styling. The footer on each page has a copyright marker and the authors e-mail.

- There is a common background image for all pages.
- There is a common css file (style.css) styling all pages loaded through the header of the base.html file.
- There is also a common javascript file (script.js) that loads a common logo in the header of all the webpages using appendChild().
- Additionally, the navigation links are added to the header using the script.js file using appendChild().

Home Page:



The Home page has:

- A short paragraph about the website.
- An embedded Youtube video explaining what the BMR is to users.
- A paragraph of text on the BMR.
- A **BMR calculator** that asks the user to select their gender, and enter their age, height and weight.

  There are validations on age (between 16yo & 90yo); height (between 1.25m & 2.5m); and weight (between 40kg & 160kg).

  Hitting the calculate button triggers the validations.

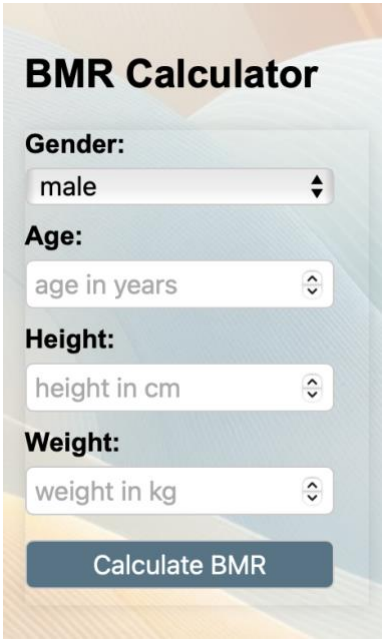
  If there are validation issues the issue will appear in red below the calculator.

  If the validations are passed the BMR will appear in green below the calculator.

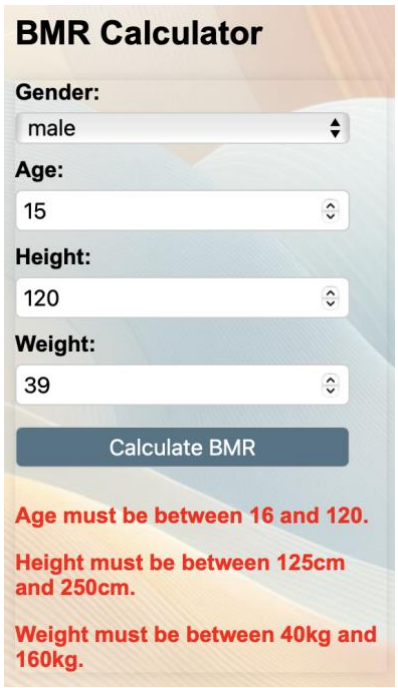
  All the interactive functionality is facilitated with javascript code. The “calculator.js” script is the source of the code for the BMR calculator.

Any data entered in the calculator is monitored and saved in the local storage of the webpage and will remain in the case of a page refresh.

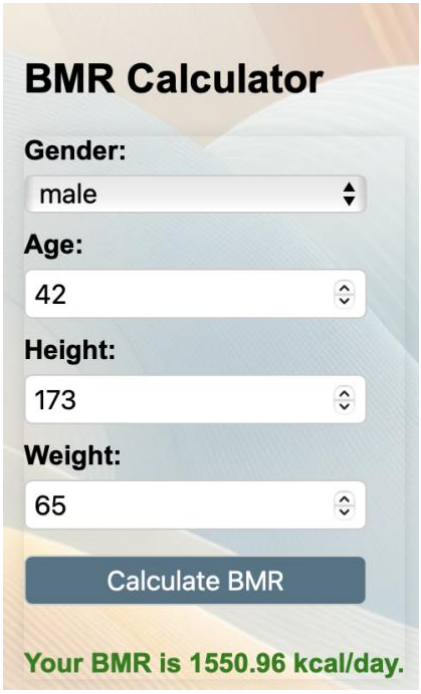
- Calculator with no data entered (on load)



- When all validations fail (comments added via document.getElementById().InnerText)



- When all validations pass (comments added via document.getElementById().InnerText)



- The **Registration Form** asks the user for their name; e-mail address; and asks them to create a password, which must be repeated.

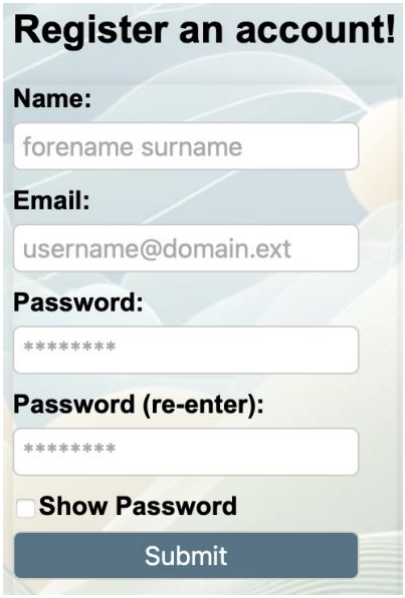
  There is a check box that will show the password characters if checked.

  The validations for the form include:

  - checking that all 4 inputs are populated;
  - checking that a valid e-mail is submitted;
  - checking that the password has at least 8 characters and contains at least 1 lowercase character, 1 uppercase character, 1 number and 1 special character.
  - checking that both password submissions match.
  - If there are validation issues the issue will appear in red below the registration form. If the validations are passed a success message will appear in green below the calculator.

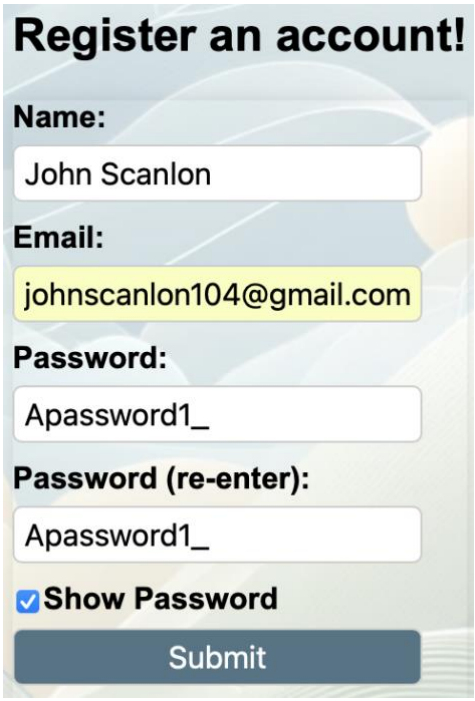
  Any data entered in the registration form is monitored and saved in the local storage of the webpage and will remain in the case of a page refresh.

- On page load

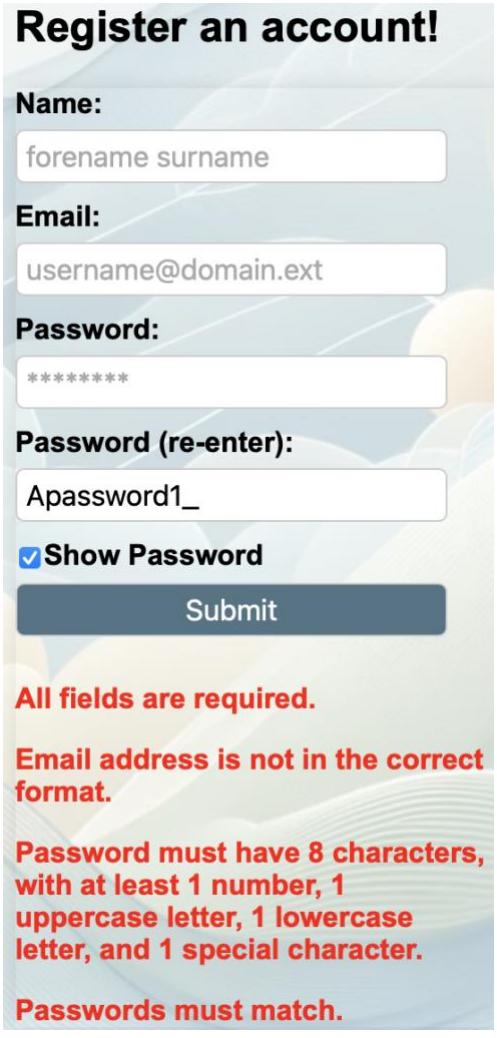


- when populated (show password box not checked)

- when populated (show password box checked)

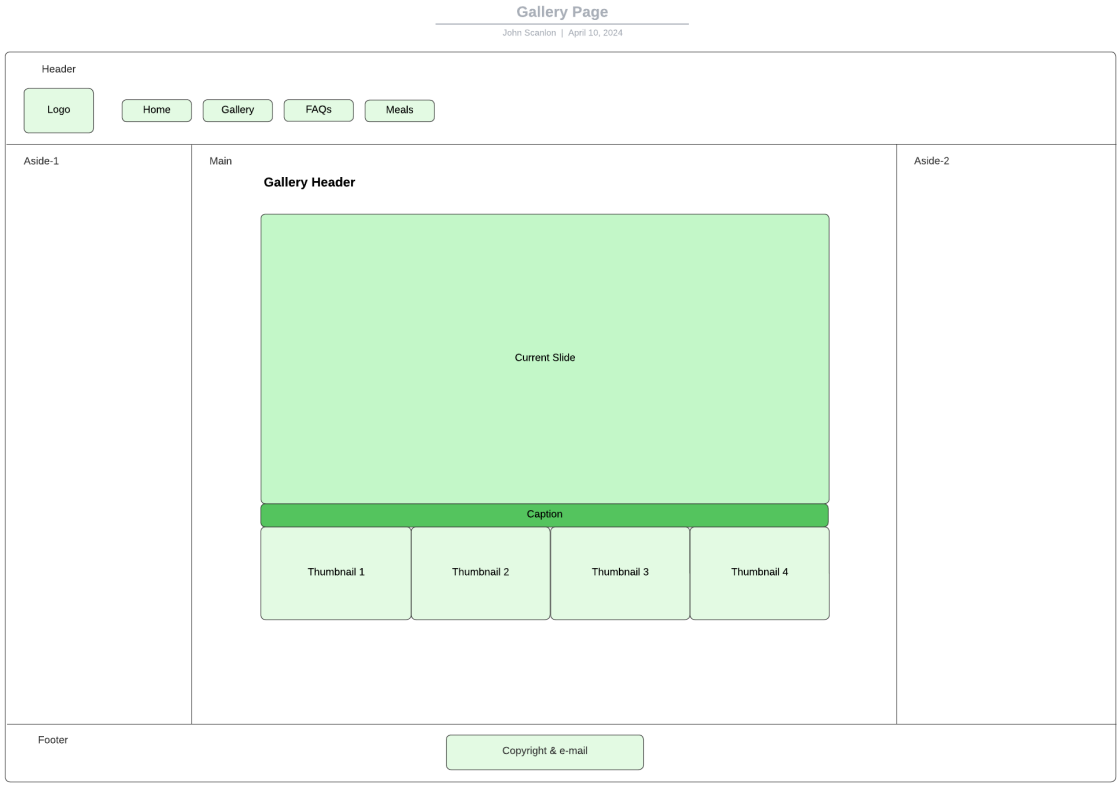


- failed submission (with all possible error messages)

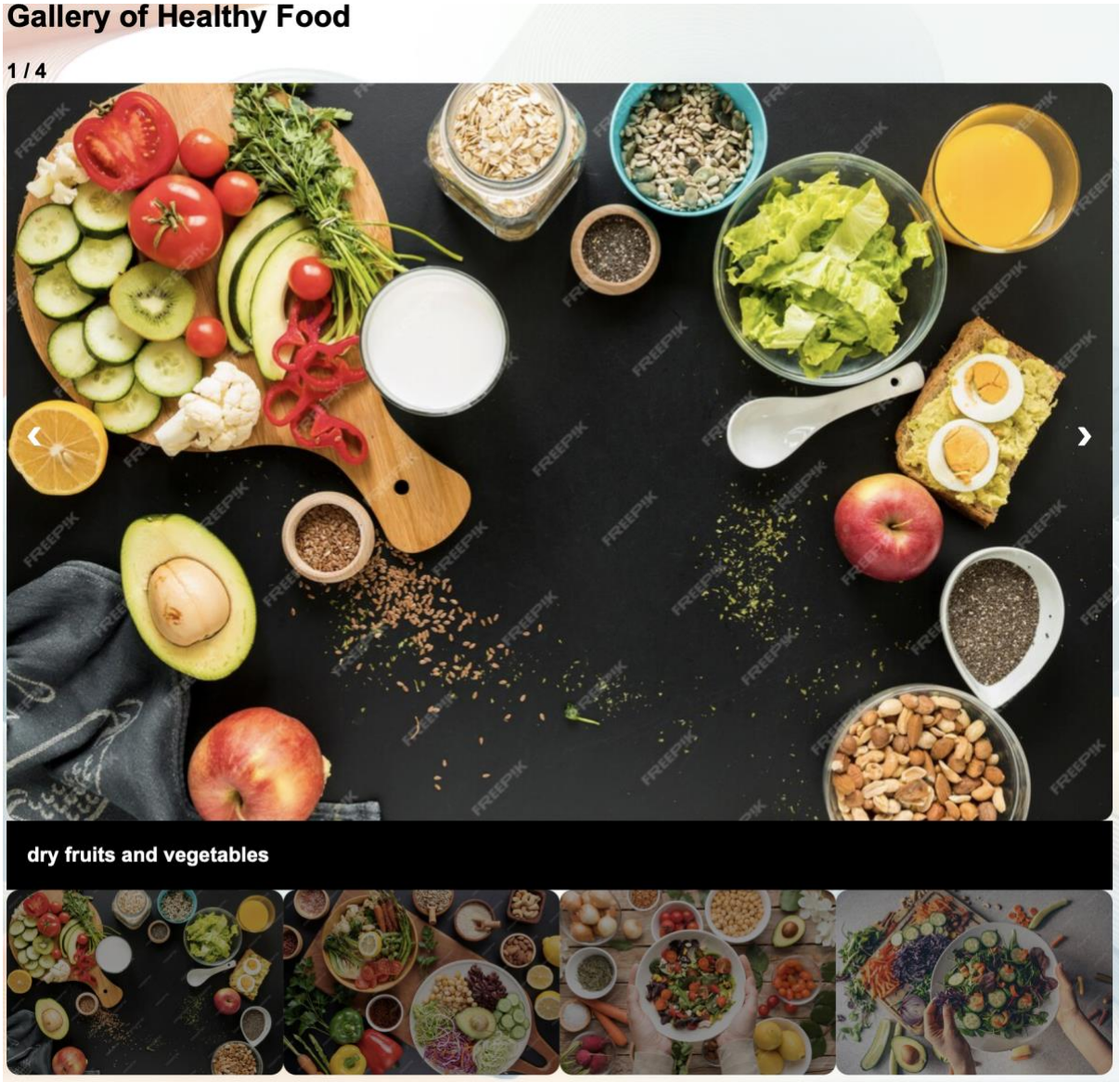


- successful submission

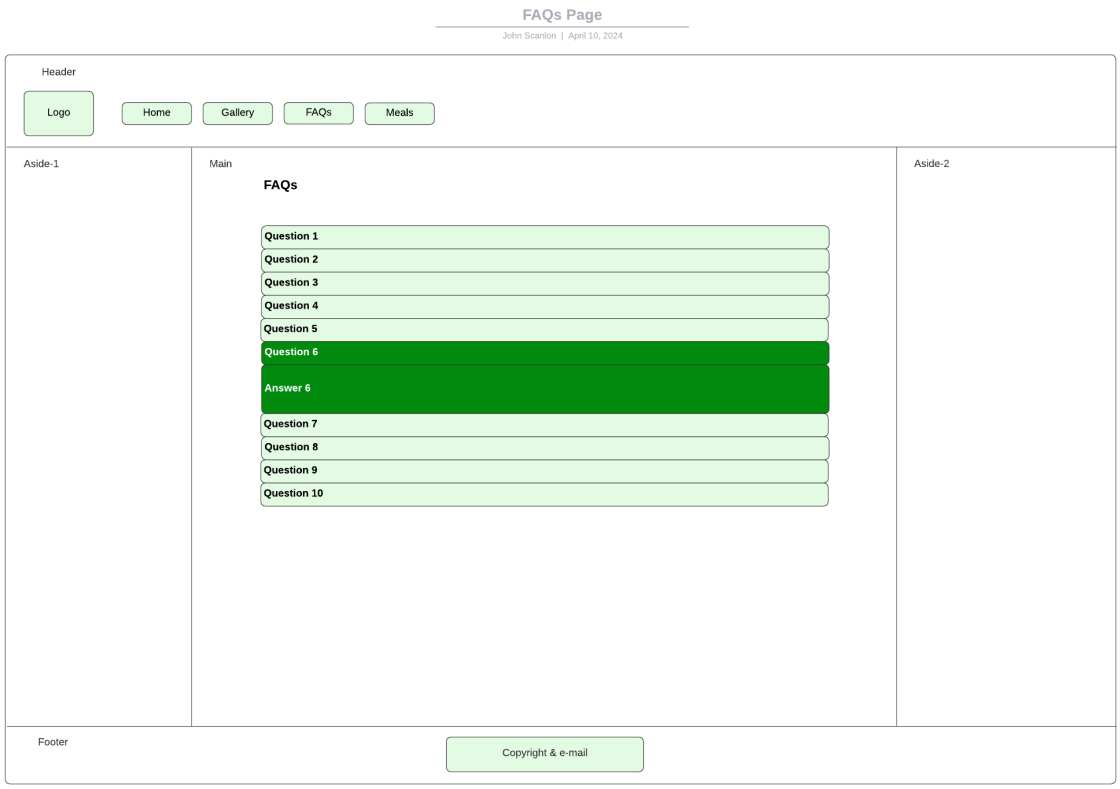
Gallery Page:



- The Gallery page consists of a gallery with a slideshow format of 4 images of health food.
- There is a main image of the selected thumbnail.
- Underneath is a caption of the selected image.
- Beneath the caption are thumbs of all the available images.
- The main image can be selected by navigating to the thumb or using the previous/next buttons at the edges of the main image.
- The thumbs lighten when the cursor hoovers over them.
- There is a gallery.js file that facilitates the interactive functionality of the webpage.



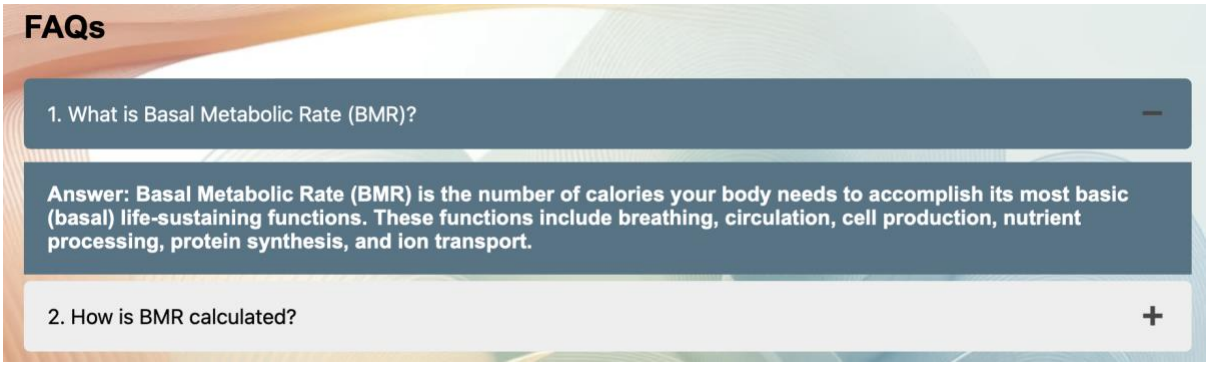
FAQs Page:



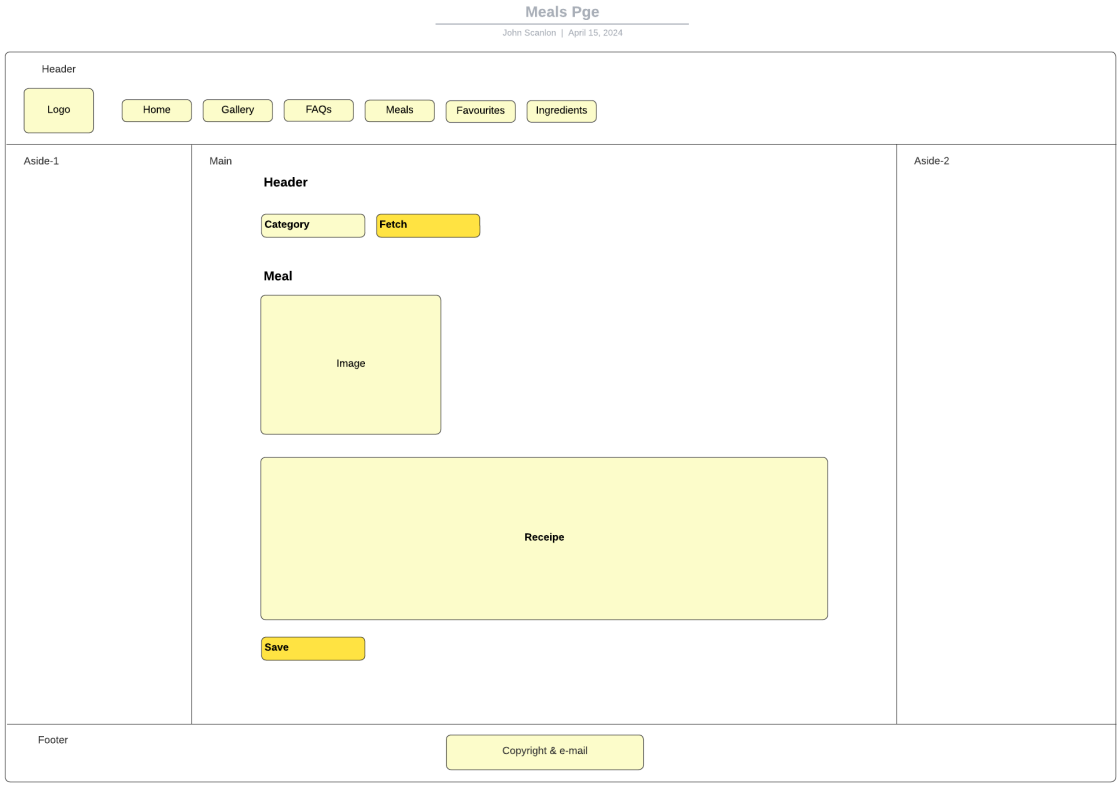
- The FAQs page consists of 10 questions and answers on the BMR in an accordion format (expand/collapse answer format)
- Each question loads collapsed; where there is a “+” symbol to expand the answer.
- When the answer is expanded the colour of the question darkens and a “minus” button appears to collapse the answer.
- There is an accordion.js file that facilitates the interactive functionality of the webpage.

    - Collapsed answers

- Expanded question 1



Meals Page:



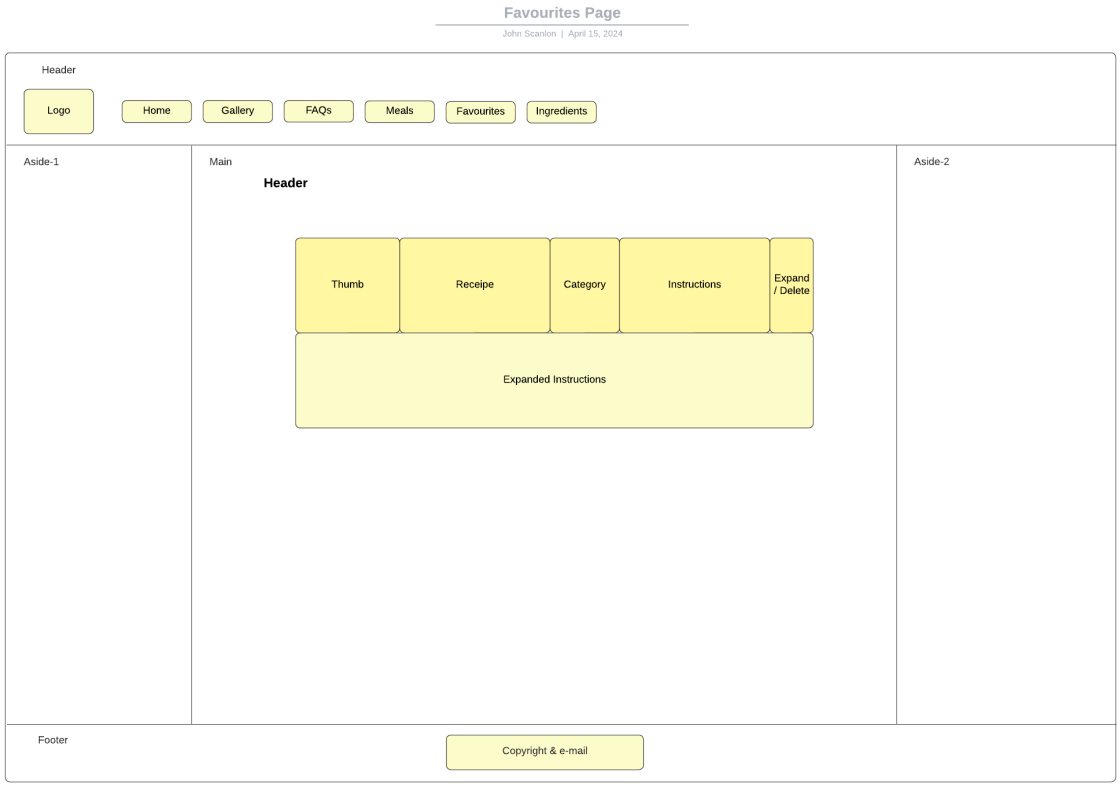
- The Meals page is linked to a meals database API the details of which can be found at this link: https://www.themealdb.com/api.php
- The "generate a random meal" button fetches data from the API using the meals_api.js file to facilitate the data pull.
- The user selects a meal category (but the default is beef) and clicks the fetch button.
- The script selects a random recipe for that category of meal and populates the page with the recipe name, an image of the meal and the cooking instructions in bullet point format.
- Clicking the fetch button, enables the "save a favourite" button.

- Clicking the “save as favourite” sends the data for the recipe (stored as string data in a hidden form on the webpage) via a POST call to the server (via Python).
- Clicking the save favourite button will take the user to the “favourites” web page where they will see the recipe they just saved in a table of recipes.

- Meals page on loading (note default category and disabled “Save as favourite” button)

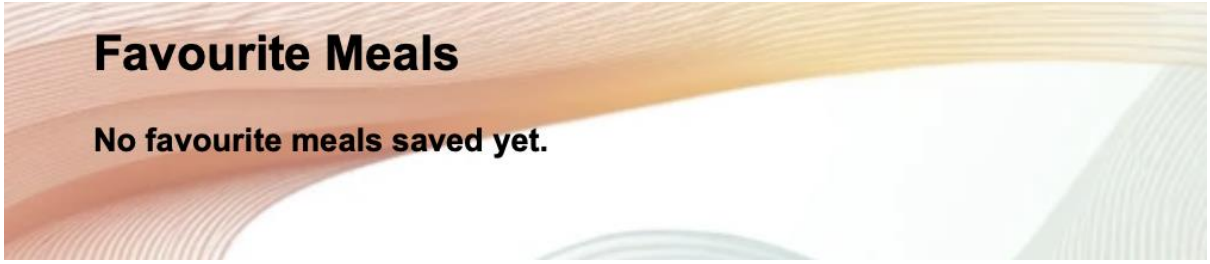
- After clicking “generate a recipe” (note the enabled “Save as favourite” button

Favourites Page:

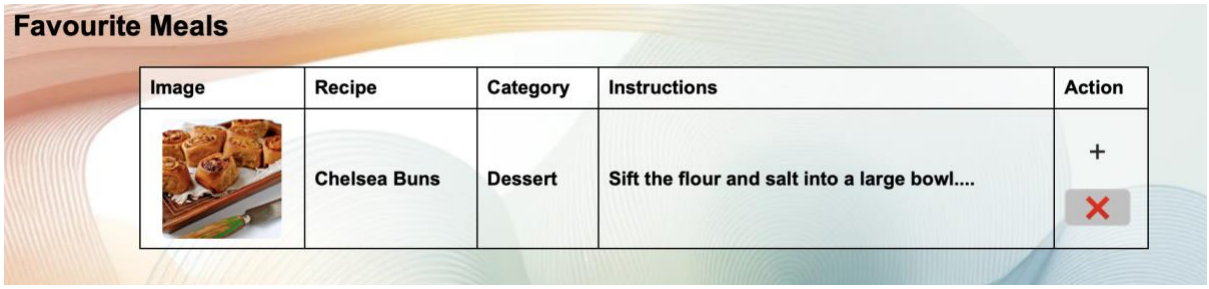


- The Favourites page is a table that contains all the recipes that were randomly generated and saved as favourites.
- The page is serviced by the favourites.js script. The purpose of which is to expand and collapse the full cooking instructions for each recipe.
- There is also a link in the table that deletes a favourite by calling a POST route to remove a favourite from the python dict that populates the table in the webpage.
- The dict uses the API's unique meal id as a key, preventing duplicates from appearing in the table.
- The table contains an image of the meal, the recipe name, the meal category and a snippet string of the cooking instructions.
- The final column of the table holds the delete icon to remove the recipe and a "+" icon to expand the full cooking instructions or "-" icon to collapse the full cooking instructions.

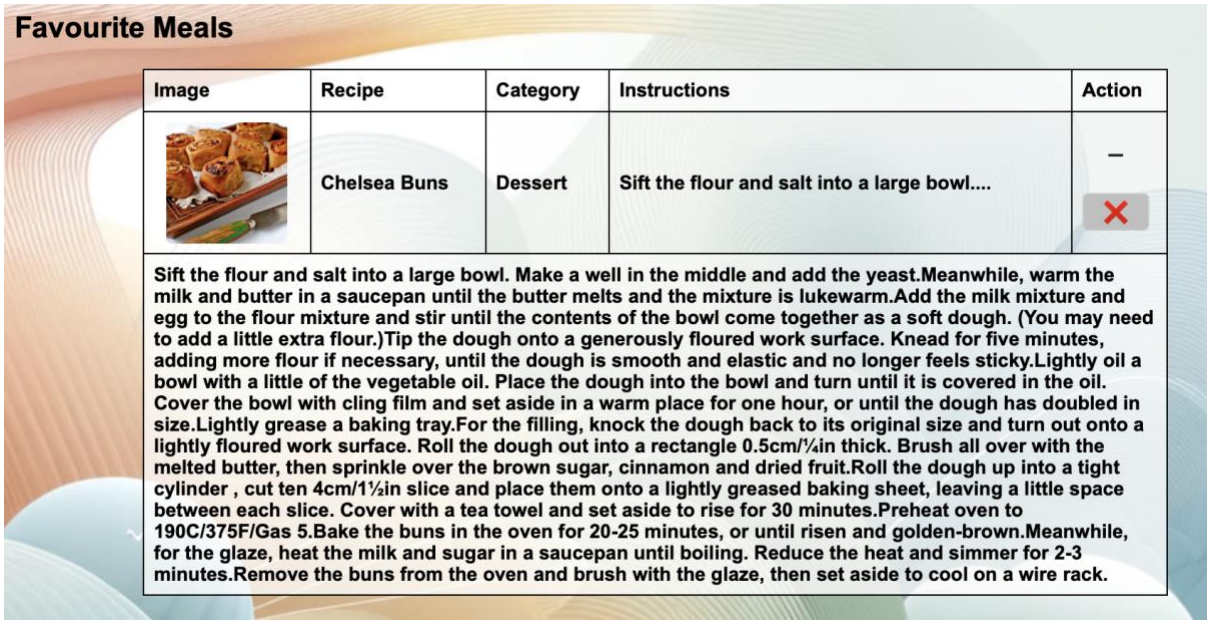
    - On load (and assuming no recipes saved as favourite yet)



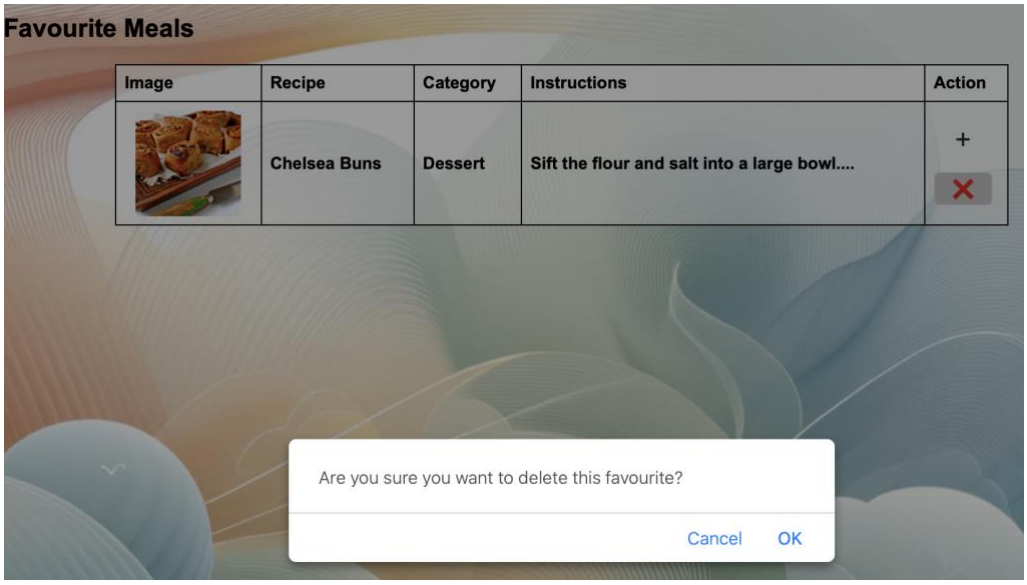
- On Load or redirect (having saved a favourite recipe)



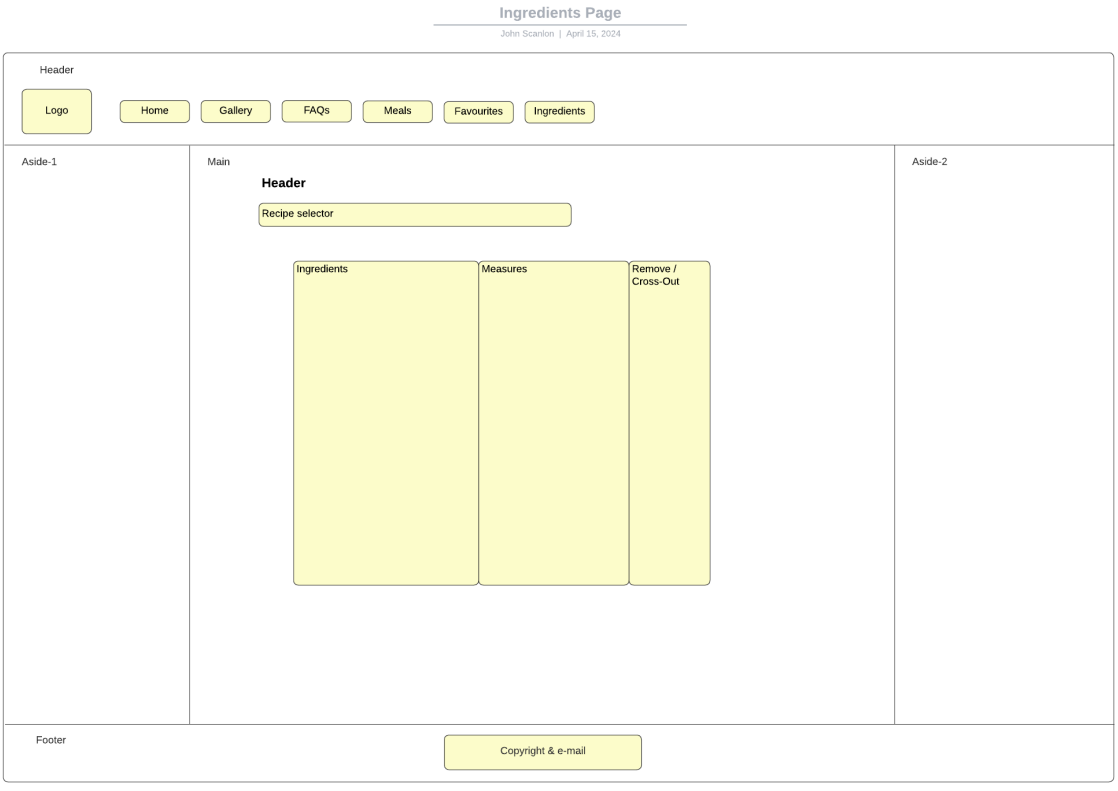
- Expanded instructions



- Confirm delete that appears if delete favourite icon clicked.



Ingredients Page:

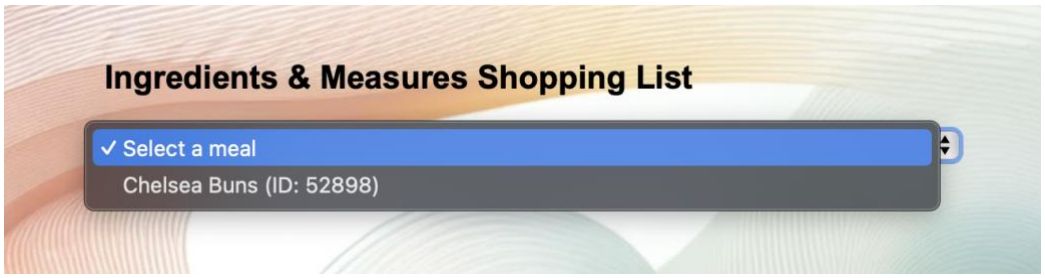


- The ingredients page is serviced by the ingredients.js script.
- The page opens with a selector to choose a recipe from those saved as favourites.

- The selection of a recipe triggers a javascript function to contact the API again and pull in the ingredients and the measures for the recipe.
- The data appears as a table with 3 columns. The first 2 columns are the ingredients and the measures. The third column contains 2 links, one to delete an ingredient; the second to cross out an ingredient.

    - On load of Ingredients page

    - Selecting a recipe



    - Having selected a recipe

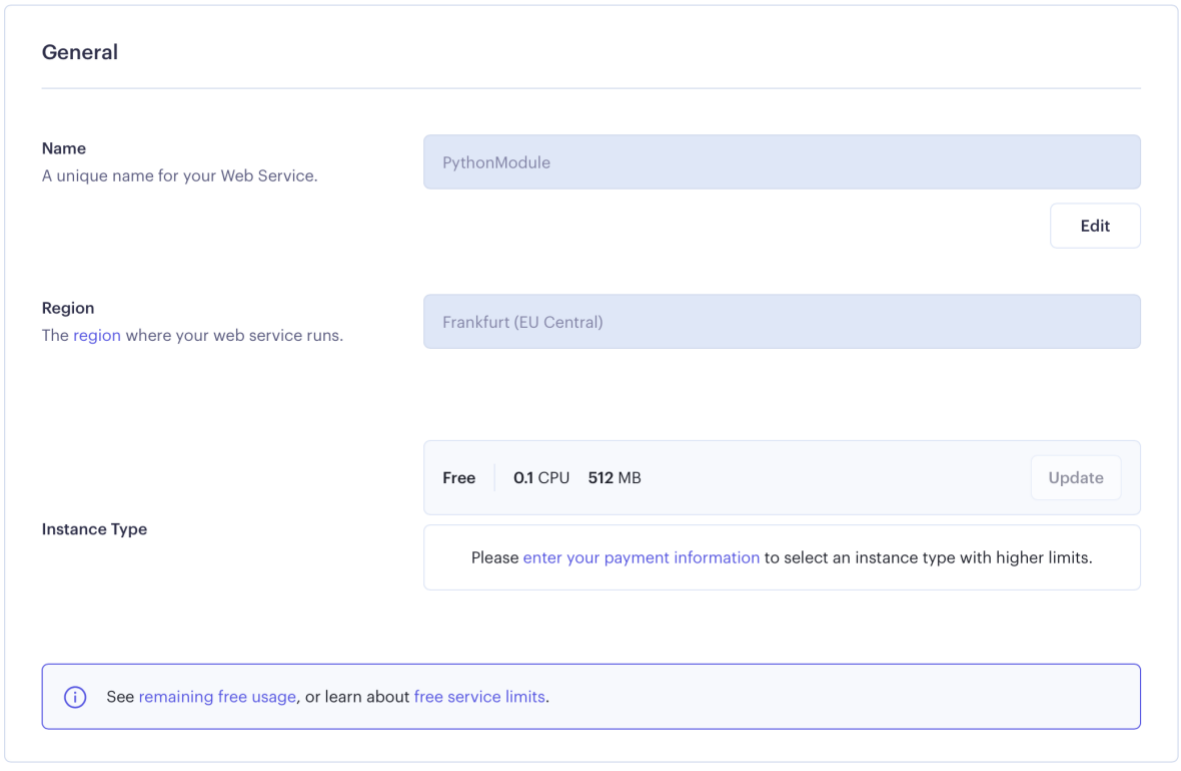
- Note deleted “white flour” ingredient and crossed out “yeast”. “milk” and “butter”

Ingredients & Measures Shopping List

Chelsea Buns (ID: 52898)

Salt	1 tsp	✗ ✓
~~Yeast~~	~~7g~~	✗ ✓
~~Milk~~	~~300ml~~	✗ ✓
~~Butter~~	~~40g~~	✗ ✓
Eggs	1	✗ ✓
Vegetable Oil	Dash	✗ ✓
Butter	25g	✗ ✓
Brown Sugar	75g	✗ ✓
Cinnamon	2 tsp	✗ ✓
Dried Fruit	150g	✗ ✓
Milk	2 tbs	✗ ✓
Caster Sugar	2 tbs	✗ ✓

Render Deployment Set-up Requirements



General

Name
A unique name for your Web Service.
PythonModule
Edit

Region
The region where your web service runs.
Frankfurt (EU Central)

Instance Type
Free 0.1 CPU 512 MB Update
Please enter your payment information to select an instance type with higher limits.

See remaining free usage, or learn about free service limits.

Build & Deploy

Repository
The repository used for your Web Service.

`https://github.com/djonskanlyn/PythonModule`

Edit

Branch
The repository branch used for your Web Service.

main

Edit

Build Command
This command runs in the root directory of your repository when a new version of your code is pushed, or when you deploy manually. It is typically a script that installs libraries, runs migrations, or compiles resources needed by your app.

`$ pip install -r requirements.txt`

Edit

Pre-Deploy Command Optional
This command runs before starting your service. It is typically used for tasks like running a database migration or uploading assets to a CDN.

`$`

Edit

Start Command
This command runs in the root directory of your app and is responsible for starting its processes. It is typically used to start a webserver for your app. It can access environment variables defined by you in Render.

`$ gunicorn app:app`

Edit

Auto-Deploy
Automatic deploy on every push to your repository, or changes to your service settings or environment. Disable this to handle your deploys manually. Learn more.

Yes

Edit

Deploy Hook
Your private URL to trigger a deploy for this server. Remember to keep this a secret.

`https://api.render.com/deploy/srv-coe4mqgl6cac73btt5f0?key=baJtb_cPsT0`

Regenerate Hook