**Alex Mann**

Candidate number: \_\_\_\_\_

Centre name: \_\_\_\_\_

Centre number: \_\_\_\_\_

**Environmentally friendly food recommendation app**

Qualification code: \_\_\_\_\_

Date: \_\_\_\_\_

Contents

[- 1](#_Toc94880224)

[Analysis 2](#_Toc94880225)

[Problem identification 2](#_Toc94880226)

[Why it is suited to a computational solution 2](#_Toc94880227)

[Computational methods that the solution lends itself to 3](#_Toc94880228)

[Stakeholders 3](#_Toc94880229)

[Existing product research 4](#_Toc94880230)

[Stakeholder research 5](#_Toc94880231)

[Requirements specification 5](#_Toc94880232)

[Essential features of solution 6](#_Toc94880233)

[Potential limitations of solution 6](#_Toc94880234)

[Hardware and software requirements 6](#_Toc94880235)

[Hardware 6](#_Toc94880236)

[Software 6](#_Toc94880237)

[Commentary 6](#_Toc94880238)

[Design 6](#_Toc94880239)

[Decomposition 6](#_Toc94880240)

[Product database 7](#_Toc94880241)

[Emissions algorithm 11](#_Toc94880242)

[Suggestion algorithm 11](#_Toc94880243)

[User interface 12](#_Toc94880244)

# Analysis

## Problem identification

In order to help combat climate change, many people are now trying to make more environmentally friendly purchasing choices in their everyday lives. However, it is often incredibly difficult to find suitable alternatives to food products and to know which products are better for the planet, as there are multiple factors which contribute to the emissions of an item. I would like to create a product to help a user find out what the best items are and offer alternatives which are similar and easy to switch to, but more sustainable.

This would require a program to scrape the internet to find relevant data and calculate the emissions of a product. This information would then be used in an app where a user could give a food item from a website, possibly in the form of a link, and the app would recommend alternatives. If I have time the interface could then be improved to either a web extension which would view a user’s online shopping basket and suggest alternatives on the page, or an item recognition software paired to a webcam or mobile camera which could take a photo of a physical item in a shop and show similar online products.

## Why it is suited to a computational solution

This problem lends itself well to a computational approach in many different ways. For example, the solution will need to collect a large amount of data on many products from various online sources, which would be incredibly time consuming to carry out manually, and the program can also continuously update its information. The software will also need to carry out a large number of calculations to work out the emissions of all the products, and then compare this data on many different products to inform the user of the best alternatives in terms of emissions. I may then integrate the software with online shopping websites to allow the user to alter their choices with minimal effort.

## Computational methods that the solution lends itself to

* Data retrieval with web scraping – the program would have to retrieve data from a variety of websites to find the environmental impact of a product and research suitable alternatives. This amount of data would be impractical for a person to do so the program will make the comparison process possible and enable users to make informed decisions on their food
* Database storage and analysis – the program will have to analyse a large quantity of data on production methods, production location, transport methods etc. on many different products to compare the emissions, making this perfect for a computer program
* Data presentation – the program will take a lot of data and represent it in a neat and easily comprehensible way to make the decisions as easy as possible for the user
* Problem decomposition – the problem can be broken down into a number of steps. These could be:
  + Scraping information about a product from the internet
  + Converting this data to an emissions statistic
  + Finding potential similar products
  + Calculating statistic for alternatives
  + Image recognition for products
  + Web scraping of online shopping basket
  + Data presentation of best changes
* Divide and conquer – the whole process would be unmanageable together, but by dividing into these subprograms, each step become challenging but doable, and then the processes can be combined with the divide and conquer method
* Abstraction – some of these processes rely on others to carry out their function, using abstraction the workings of the preceding functions can be ignored to simplify the problem

## Stakeholders

The potential clients for this software would be anyone who is interested in their environmental footprint which, in the modern day, is the majority of the population. I aim to make it as accessible and easy to use as possible to appeal to a wide range of people of all ages and technological abilities. The software could also be angled towards the vegetarian community who could use the software to find vegetarian alternatives to common meat-based foods.

The stakeholders for the software will be people with a particular interest in the environment who are keen to minimize their emissions from food as much as possible and may not know exactly which products have what effect on the environment.

During this project, I will a specific stakeholder, Kevin, a vegetarian and avid environmentalist who tries to buy environmentally friendly food wherever possible, but often finds it hard to find good alternatives and to know which items are better for the environment.

## Algorithm research

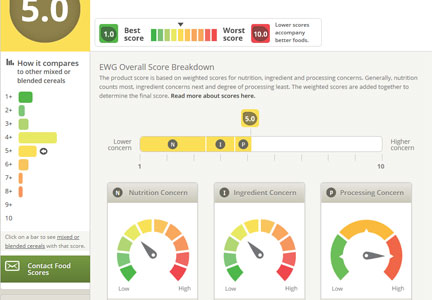
Before I begin, I need to know roughly how my emissions calculation algorithm will work and if it is possible to do accurately so I will research what the main emissions producing areas are in food production

My initial research showed 4 main sections: production, transportation, packaging, and cooking, and these can be calculated as follows:

* Production – this is the emissions it takes to make the food and is unique to each type of food. While it also depends on a few unknown factors such as the specific farming method, there exists accurate and well researched data online with the average emissions for each type of food, so I can use a database from the internet for this section
* Transportation – this is the emissions produced by transporting the food from the place of production to the distribution centre, and then to a user where it is delivered. I can write an algorithm to calculate this using the place of origin of the food and the location of the user, which can then be used to calculate the distance travelled, which can then be used with the method of transport, the emission rate of that mode of transport and the mass of the product to calculate transport emissions
* Packaging – my initial research showed the emissions of packaging are negligible compared to the other three sections, so I do not need to consider this section
* Cooking – this is the emissions produced to cook the item, which is affected by the cooking method and cooking time. However, most products will not have cooking information available on the website and the emissions will be heavily dependent on the specific appliances owned by the user, so I will only add this section if I have time at the end

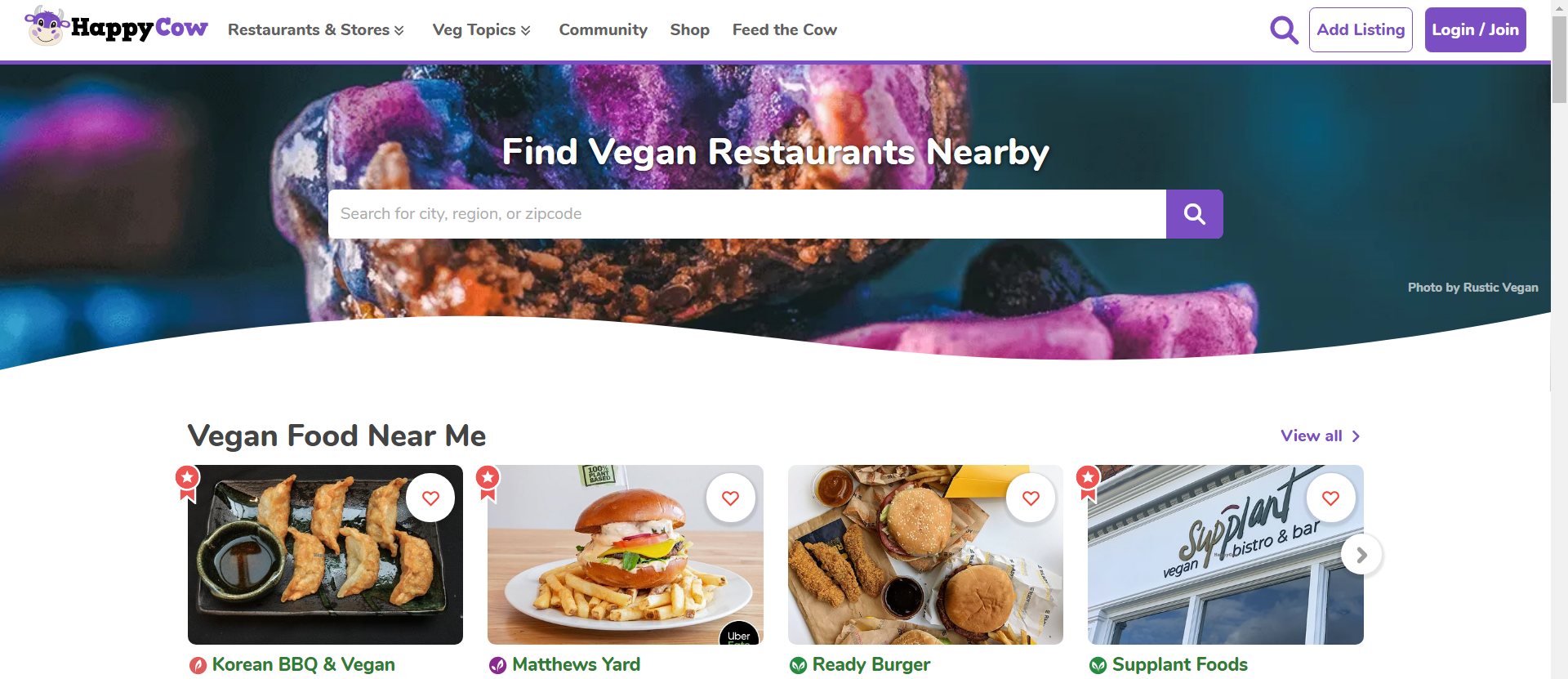
## Existing product research

There are some companies aiming to achieve a similar purpose, but I do not think any of them have solved the specific problem I would like to solve. For example, there are many apps which work in a similar way but instead of giving information on environmental impact they give health data. One of these is EWG Food Scores, which is an easy-to-navigate database with ratings for thousands of products based on nutrition, ingredients, and processing. I would like to take a similar format of data presentation with visual graphs and helpful colours to make the program as intuitive as possible.



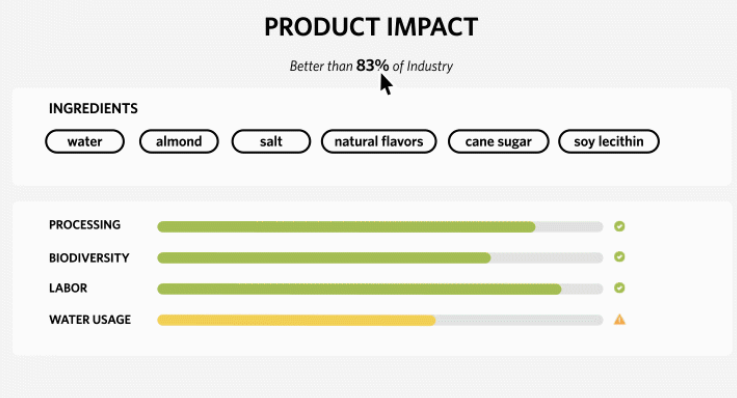
‘EWG Food Scores’ example page

However, some projects have aimed to tackle the environmental impact of food, and one example of this is ‘Happy Cow’, a simple and user-friendly app which tells you which local places offer vegan or vegetarian food. While this is useful, my aim is more to give environmentally better alternatives without drastic changes to someone’s diet, for example by changing from one brand of bread to another, or one cut of meat to another.



‘Happy Cow’ home page

Another example is ‘How Good’. This is the software which is most similar to my idea that I could find. It gives a detailed breakdown of the environmental impact of various foods and offers alternatives, but it is a largely commercial product aimed at big producers to make changes in their production line and stick a ‘How Good’ certificate on their products. I would like to aim for a smaller scale, personal product which enables individuals to make changes to their diets to benefit the environment. I would however like to achieve the detailed data analysis of ‘How Good’ on a smaller scale.



‘How Good’ example page

## Stakeholder research

### Survey

My first step in stakeholder research was to carry out an online survey to find out if my idea would be useful, so I wrote questions to find out people’s current attitudes and opinions towards food shopping and the environment, and then distributed the survey amongst people I knew, covering as diverse of a demographic as possible. My survey had 30 respondents, and the results are analysed in the table below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Question** | **Results** | **Why I chose this question** | **Results analysis** | **What this means for my project** |
| How often do you shop for groceries online? |  | I wanted to find out how frequently online grocery shopping is used to find out how much positive impact my product could have | 60% of users regularly shop for groceries online, with an average of 1.1 shops per week | Online grocery shopping is clearly widely used which means my product has the potential to have a significant positive impact on the environment |
| Which site do you use most often for grocery shopping? |  | My product will have to, at least originally, be specific to only one or two websites, so knowing which one more people use will help me decide which to focus on | Sainsbury’s was the most used website with 9 people, closely followed by Tesco with 7 | Dependent on later research, I will likely ensure my product is compatible with Sainsbury’s, and may also try to integrate with Tesco |
| How important is environmental impact to you when choosing food to buy? |  | I need to find out what people’s current attitudes are towards the impact of their food, if emissions are not something most people care about, my product will not have much impact. However, this also might be affected by how well informed they are about the food they buy, so I will clarify this with the proceeding questions | The majority of people said environmental impact was somewhat important, while no one said it was not at all important | All the respondents to my survey attached some importance to the environmental impact of their food, so it seems that with relevant and accurate information people would make some effort to reduce their emissions |
| How aware are you of the environmental impacts of different foods? |  | If most people are not particularly aware of the impacts of their food, but would be willing to change their habits if they had more information, then my product has the potential to greatly reduce emissions, while if everyone already knows all the relevant emissions statistics, my product is redundant | 56% of people said they were somewhat aware, and the data is clearly centred around the centre of the range | This suggests most people think they have a reasonable grasp of the environmental impacts of food, but most do not feel fully confident in their knowledge which could cause them not to make changes to their lifestyle based on unsure data. My product can therefore help this and encourage people to make changes to the food they consume |
| Rank the following items in order of emissions per kilogram you think they produce |  | This question quantitatively tests the perceptions people had in the previous question about their own knowledge and shows how accurate the data people have is. I tried to choose a wide variety of items to find out where, if any, are the most common misconceptions | The first graph shows how the average person ranked the items, while the order they are listed is the correct order. The scatter plot shows average rank on the x axis against emissions on the y axis, excluding beef. This has only a 60% correlation and it is clear from the graph the ranks are largely inaccurate | Most people correctly predicted the top product (beef) and the put the lowest 4 at the bottom (Tomatoes, pasta, frozen raspberries, potatoes), but people were not good at differentiating between the middle items. This shows the importance of my product to inform people about this data |
| Which of the following would you be willing to do to reduce the emissions of your food | |  |  | | --- | --- | | Choice | % Responses | | Switch to products produced more locally | 92 | | Switch to more expensive products | 47 | | Switch to slightly different products of the same type (e.g., different types of cheese or different sizes of fruit) | 72 | | Switch to products with a different production or transportation method (e.g., fresh to frozen) | 72 | | Switch to product alternatives (e.g., cow's milk to soy milk) | 39 | | Switch to similar foods (e.g., chicken to turkey) | 47 | | Switch to completely different foods | 17 | | I need to find out which types of sacrifice people would be willing to make to help the environment, so I know how best to suggest alternatives to a user. If people are only willing to make insignificant adjustments to their diet, then I need to only recommend products with minimal change, but if they are willing to change everything about what they eat then I can expand my range of suggestions | Most people are willing to make some sacrifices but there is a lot of variety in what sacrifices people would make | The fact most people ticked at least one box shows that my product can encourage positive change, while the variety suggests I need to allow the user to select the types of suggestions the app gives |

### Primary stakeholder research

My primary stakeholder is Kevin, he is an avid environmentalist and vegetarian so is very interested in this product. I initially chose to speak to him via email since there was no physical evidence I needed to show him, and so it was easier and more efficient to contact him remotely. Here is what he said:

“Hi

I really like the idea for the project and think it has the potential to change a lot of people’s habits and awareness and save a lot of emissions. Here are some things I would like to see in the product:

Firstly, the data obviously needs to be accurate and reliable otherwise the whole process is useless, getting people to switch to items which we don’t really know if they’re better or not. But it’s also important that the data is given to a reasonably high level of precision to give the impression of accuracy, since I think people would be more likely to switch if they thought the data was accurate and reliable, I think 2 significant figures should be enough

Secondly, the data needs to be presented in a sensible way, I need to be able to easily compare the relevant information from various products without having to navigate around lots of pages. The data should also be given in sensible units, I don’t want to have to read 10 decimal places for no reason, and it would also be quite nice to be able to filter or sort the data, for example to only include vegetarian items, or to sort by emissions or by customer rating. Also, in order to choose a product, I need to obviously know its name, price and customer rating, and the mass and/or number of portions would be nice. In terms of the way emissions statistics should be given, I think emissions per portion would be the most useful metric, but it would also probably be nice to be able to switch between a few slightly varying metrics. However whichever metric is given I would like definitely like to have either a traffic-light system to easily show how the products compare, and maybe the data could be compared to everyday events (this product is equivalent to a 5-minute drive).

In terms of the suggestions you’re giving, obviously these have to be relevant to my search, I don’t want to search for beef, and it suggests cereal, that’s not helpful and is not a change anyone would ever agree to. I also think it would be nice to be able to see why a product was swapped (“Asparagus from Peru -> Kale from Kent”) and to be able to change the strictness of the search since different people might have different opinions on how much or little they want to change

Finally, for the interface, it needs to be straightforward to use and it should be compatible with all common devices. Also, it needs to be compatible with either a lot of sites or one or two very popular sites otherwise no one will be able to use it. It would be great if the product looked nice too, but that’s not essential for a functional product like this. Some other things I would quite like would be for a search bar to always be present on any page, and to be able to search for a type of product or for the link to a product and it gives relevant suggestions, but either of these individually would be fine.

I hope that’s helpful and good luck

Kevin”

## Requirements specification

Key:

* Essential features/stakeholder needs
* Nice-to-have features/stakeholder wants
* Requested by primary stakeholder

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#** | | **User requirement** | **Solves** | **Evidence** | **Justification** | **Success Criteria** |
|  |  | **Emissions data** |  |  |  |  |
| 1 | 1.1 | Accurate data | Enable useful comparison between products | “The data obviously needs to be accurate and reliable” | Need data to be accurate enough to differentiate between various products’ emissions to recommend lower impact products | -Calculation algorithm considers all available factors using reliable and well-research data  -Checked by explaining algorithm to stakeholder and asking if acceptable |
|  | Precise data | Gives sufficient information to differentiate similar products | “It’s also important that the data is given to a reasonably high level of precision” | Shows a user that the data is accurate and enables differentiation between similar products, for example giving 1.3 and 1.1 shows a difference, rounding to 1 and 1 does not | -Data given to at least 2 significant figures |
| 1.2 | Production data | Significant part of important emissions data | My initial research showed this as a key are of emissions production | This is a major factor in the environmental impact of a food product, so I need to make sure I consider this in my calculations | -Product uses an algorithm to calculate production emissions  -Algorithm uses reliable and well-researched data  -Checked by explaining algorithm to stakeholder and asking if acceptable |
| 1.3 | Transport data | -Product uses an algorithm to calculate transport emissions  -Algorithm uses reliable and well-researched data  -Checked by explaining algorithm to stakeholder and asking if acceptable |
| 1.4 | Cooking data | -Product uses an algorithm to calculate cooking emissions  -Algorithm uses reliable and well-researched data  -Checked by explaining algorithm to stakeholder and asking if acceptable |
| 1.5 | Statistics given in sensible units | Enables people to understand the impact of their food more easily | “The data should also be given in sensible units” | If people are able to easily understand the information, they can make more informed decisions | -Statistics given in sensible units  -Number of non-significant figures never greater than 2  -Ask for stakeholder approval |
| 1.6 | Emissions data compared to everyday events | “Maybe the data could be compared to everyday events” | -Emissions data compared to suitable and simple everyday event  -Ask for primary stakeholder’s approval |
| 1.7 | Traffic light system for emissions data | “Definitely like to have … a traffic-light system” | -Colour system to indicate emission level  -Ask for primary stakeholder’s approval |
| 1.8 | User options for data format | Enables people to view emissions in different units, i.e., per kilogram, per serving, per portion or per calorie | “Would also probably be nice to be able to switch between a few slightly varying metrics” | Different metrics have different uses, e.g. people buy a product based on mass, emissions per kilogram is more useful, while if someone is aiming to buy a certain number of portions, emissions per portion is better | -Option for user to view data on, sort by and suggest based on multiple emissions metrics  -Options for units include CO2 emissions per kilogram, serving, portion and calorie |
|  |  | **Suggestion algorithm** |  |  |  |  |
| 2 | 2.1 | Relevant suggestions | Makes suggested alternatives more likely to be accepted by user | This is clearly advantageous  “Obviously [the suggestions] have to be relevant to my search” | I want to make people make as many environmentally-positive changes as possible, and relevant suggestions would clearly increase how many changes people make | -Trial should have at least a 20% suggestion accepting rate |
|  | 2.2 | Customisable suggestion criteria | Enables user to change which type of suggestions are given | My survey showed people have a variety of opinions on what changes they would make  “I also think it would be nice … to be able to change the strictness of the search” | It is clear this should be a user option to maximise positive changes | -User has option to change the strictness of the search |
|  |  | **Interface** |  |  |  |  |
| 3 | 3.1 | Easy to use | Enables user to use product easily | “Needs to be straightforward to use” | Increases likelihood of people using and continuing to use the product | -Primary stakeholder and majority of users in a trial are happy with ease of use |
|  | 3.2 | Compatibility with shopping sites | Product works for a reasonable number of potential users | “Needs to be compatible with … one or two very popular sites”  My survey showed the highest use for Sainsbury’s | Needs to be easily compatible with the most common website(s) to increase usership | -Compatible with at least Sainsbury’s, the most common website from my survey |
|  | 3.3 | Compatibility with devices | Product works on desktop, tablet and mobile | “Should be compatible with all common devices” | Increases usership and ease of use | -Product works and stakeholder is satisfied with ease of use on desktop, tablet and mobile |
|  | 3.4 | Aesthetically pleasing | Makes it nicer for users to use | “Would be great if the product looked nice too” | Increases likelihood of people using and continuing to use the product | -Primary stakeholder and majority of users in a trial are happy with layout |
|  | 3.5 | Constant search bar | Users always able to search for product | “a search bar to always be present on any page” | Enables users to quickly and easily be able to search for new products or comparisons wherever they are on the app | -Search bar always visible in app  -Search bar works |
|  | 3.6 | Search by product or URL | Gives users range of search options | “Be able to search for a type product or for the link to a product and it gives relevant suggestions” | Enables users to search for a product (e.g., bread) and be given comparisons of different bread products, or a link to a specific bread product on the Sainsbury’s website and similar options would be recommended | -Option to search by search term or URL |
|  | 3.7 | List of multiple options displayed | Gives users multiple choices on environmentally friendly options | “I need to be able to easily compare the relevant information from various products” | When a user searches for an item, it needs to give multiple alternatives with relevant data so the user can choose between them | -At least 3 alternatives displayed  -All alternatives relevant in some way |
|  | 3.8 | Name of product displayed for each item | Shows relevant data | “Also, in order to choose a product I need to obviously know its name, price and customer rating, and the mass and/or number of portions would be nice” | This is a detail which may influence which product a user chooses so it is vital to show this information for each product | -Product name displayed for each item |
|  | 3.9 | Price displayed for each item | -Accurate price displayed for each item |
|  | 3.10 | Customer rating displayed for each item | -Accurate rating displayed for each item |
|  | 3.11 | Mass displayed for each item | -Stated mass displayed for each item |
|  | 3.12 | Number of portions displayed for each item | -Stated number of portions displayed for each item |
|  | 3.12 | All data comparison given on one page | Easy comparison of products | “I need to be able to easily compare the relevant information from various products without having to navigate around lots of pages” | People need to be able to see all relevant in one place at the same time to easily be able to make decisions on which products to buy | -All data comparison given on one page |

## Essential features of solution

In summary, the main features which are required for my product are as follows:

* Easy-to-use system – must be minimal effort to encourage users to make sacrifices in their diets
* Accurate data – must give correct and useful information otherwise recommendations are meaningless
* Helpful alternatives – must have balance between environmental benefit and similarity to initial item. The user is unlikely to change from steak to artificial meat but change to a locally produced or organic meat.

## Potential limitations of solution

* May be difficult to get exact data – producers will never publish environmental impacts and not all factors will be discoverable with just the product name
* People may be more concerned with health, price, or taste rather than environmental impact
* Will only work with a limited number of online shopping websites, probably only Sainsburys, since each site is incredibly time consuming to cover
* Products with multiple ingredients (pies, ready meals, cakes) do not have listed quantities of ingredients so it will not be possible for me to calculate its emissions, so my product will only work for single ingredient products or products with known emissions (apples, steak, bread)

## Hardware and software requirements

### Hardware

* Smart mobile device capable of running the app, with touchscreen to interact with the program
* Computer capable of running the web extension
* Camera to take a photo of a product to search for, must be of sufficient resolution in order to recognise the product from an image

### Software

* Windows, Linux, or Mac operating system – operating systems supported by python
* Apple or Android mobile device to run app
* Python interpreter – the project will be written in python, so requires an interpreter
* Tkinter for Python – library used to make apps
* BeautifulSoup for Python – library used to scrape website data
* Webdriver for Python – library used to open pages on a web browser from my app
* Pandas for Python – library used to handle the large datasets
* Heroku – scheduler and web-based database storage, used to store, update, and access the database of products and relevant information about them used to calculate environmental data
* Psycopg2 module for Python – to access and update databases on Heroku from Python

## Commentary

Having researched all areas of my project from various sources, I now know the requirements for my product and the rough form it will take, so I can begin to design the product

# Design

## Decomposition

My project will be divided into 4 main sections:

* A database of all the products on one or more food shopping websites with key information such as price, size, rating etc. and also relevant data used to calculate emissions
* An algorithm which calculates the emissions of a product based on available data such as transport times, recycling material and ingredients
* An algorithm which takes an input product and recommends similar products which are more environmentally friendly, while also considering cost and similarity
* An interface for a user to use the rest of algorithms to change their online shopping choices, either in the form of a web extension or an app

### Product database

In order to create the product database, I will write python scripts to scrape the product data off supermarket websites.

My first step is to choose which site(s) to use for the data, and I will choose this based on which has the most useful data for calculating emissions and how popular each website is. I have already gathered from my survey that Sainsbury’s is the most common site for my users, followed by Tesco, but I will also look at other factors. I compared the product pages on a variety of popular websites in the following table

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Brand** | **Product details** | | | | | | | | **Estimated UK users (millions)[[1]](#footnote-1)** |
| **Expiry duration** | **Vegan/ vegetarian** | **Detailed nutritional information** | **Package recycling** | **Cooking instructions** | **Environmental certificates** | **Religious information** | **Place of production** |
|  | ✔ | ✔ | ✖ | ✔✔ | ✔ | ✔ | ✔ | ✔ | **30.1** |
|  | ✔ | ✔ | ✖ | ✖✖ | ✔ | ✔ | ✖ | ✖ | **23.2** |
|  | ✔ | ✔ | ✔ | ✔✔ | ✔ | ✔ | ✖ | ✔ | **17.1** |
| Amazon.co.uk: Amazon Fresh Stores | Now Open in London | ✖ | ✔ | ✖ | ✖✖ | ✔ | ✖ | ✖ | ✖ | **15.7** |
|  | ✖ | ✔ | ✖ | ✔✖ | ✔ | ✖ | ✖ | ✖ | **15.7** |
|  | ✔ | ✔ | ✖ | ✔✔ | ✔ | ✖ | ✔ | ✖ | **15.0** |
| The Co-operative Group - Wikipedia | ✖ | ✖ | ✖ | ✔✔ | ✖ | ✖ | ✖ | ✔ | **6.8** |
|  | ✔ | ✔ | ✖ | ✔✔ | ✔ | ✔ | ✖ | ✔ | **5.5** |
|  | ✖ | ✔ | ✖ | (✔✔)[[2]](#footnote-2)\* | ✖ | ✖ | ✖ | ✔ | **5.5** |

From this table I think that the most suitable websites for my project are Sainsbury’s and Tesco due to their combination of a large amount of useful information and their high number of users. Having better data makes it easier for me to make accurate emissions estimations on products and the website having more customers makes my product useful to a wide range of people. I will begin by collecting data from Sainsbury’s since it has the easiest layout for web scraping and had the highest number of users in my survey, and then I will add products from Tesco later if I have time.

Then I will have to write a python script to scrape the data off these websites. I will use the Python library Beautiful Soup to parse the page and extract the data. Firstly, I will have to scrape the list of sectors within the websites into a ‘sectors’ database. These sectors make up the tree of categories in which the products are sorted on the websites (as below). Then for each of these sectors, I will scrape the home page of this sector for a list of all the products in the sector and visit each product page to gather the relevant information. The products and their data will be stored in a ‘products’ database.

Before I can collect the data, I need to design the databases to store the information:

**‘Sectors’ database:**

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Data type | Description | Example |
| uuid | uuid | Unique ID of sector | 36cc5ac9-86a4-471e-bba6-6e32505abde6 |
| name | string | Name of sector | “Bacon” |
| path | string | Full path of sector in tree | “Meat & fish>Bacon & sausages>Bacon” |
| num\_products | int | Number of products in the sector | 150 |
| date\_updated | date | Last date at which the products in this sector were scraped, None if never updated | 01/01/2021 |
| website | string | Which website the sector is on | “Sainsbury’s” |
| url | string | The link to the sector page | “https://www.sainsburys.co.uk/shop/gb/groceries/meat-fish/bacon” |

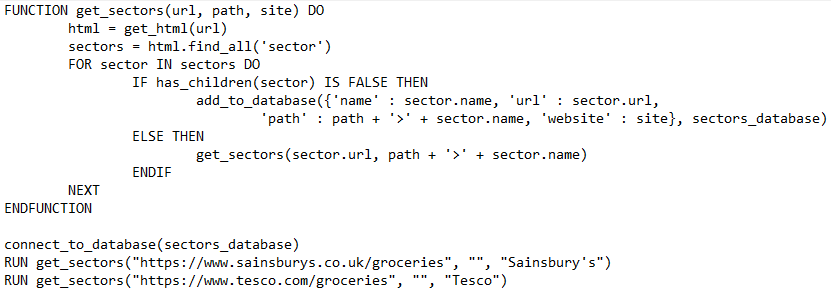
**‘Products’ database:**

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Data type | Description | Example |
| uuid | uuid | Unique ID of product | 5c87faef-cf83-4f9d-9964-3137aa9cd23a |
| name | string | Name of product | “Sainsbury’s Smoked Streaky British Bacon, Taste the Difference 220g” |
| date\_updated | date | Last date at which the product information was updated | 01/01/2021 |
| url | string | The link to the product page | “https://www.sainsburys.co.uk/gol-ui/product/bacon/sainsburys-ultimate-smoked-streaky-bacon--taste-the-difference-220g” |
| sector\_id | uuid | Uuid of sector which the product is in | 36cc5ac9-86a4-471e-bba6-6e32505abde6 |
| price | float | Price of product in £ | 2.00 |
| rating | float | Star rating out of 5 from customer reviews | 3.6 |
| num\_reviews | int | Number of customer reviews on the product | 61 |
| expiry\_duration | int | Typical life of product after purchase, in days | 7 |
| vegetarian | bool | Whether the product is vegetarian or not | FALSE |
| vegan | bool | Whether the product is vegan or not | FALSE |
| religious\_info | string | List of religious information e.g. Kosher, Halal | “[]” |
| description | string | Product description from website, for if a user wants to see more information about a potential replacement | “Oak smoked, dry cured, air dried, streaky bacon rashers made with Outdoor Bred British Pork.” |
| nutrition\_100 | string | Dictionary of nutritional information per 100g, all quantities measured in g except energy in kcal | “{‘Energy’: 337, ‘Fat’: 21.3, ‘Saturates’: 8.5, ‘Mono-unsaturates’: 14.3, ‘Polyunsaturates’: 2.5, ‘Carbohydrate’: 1.0, ‘Sugars’: 0.5, ‘Fibre’: 0.6, ‘Protein’: 31.0, ‘Salt’, 4.50}” |
| pack\_servings | int | Number of servings per pack | 7 |
| mass | int | Mass of product in grams | 220 |
| ingredients | string | List of ingredients | “[British Pork, Sea Salt, Sugar, Preservatives: Sodium Nitrite, Sodium Nitrate, Antioxidant: Sodium Ascorbate]” |
| allergens | string | List of allergens, empty list if none | “[]” |
| standards | string | List of any environmental/animal welfare certificates or standards associated with the product | “[RSPCA]” |
| info | string | List of any extra environmental/health information associated with the product | “[Outdoor bred]” |
| cook\_times | string | Dictionary of the average cooking time for various appliances, in minutes | “{‘Grill’: 5.5, ‘ShallowFry’: 3.5}” |
| origin\_country | string | Country of origin of product | “United Kingdom” |
| recycling | string | Dictionary of recycling instructions of the various parts of the packaging | “{‘base tray’: None, ‘top film’, None, ‘base Label’: ‘recycle’}” |

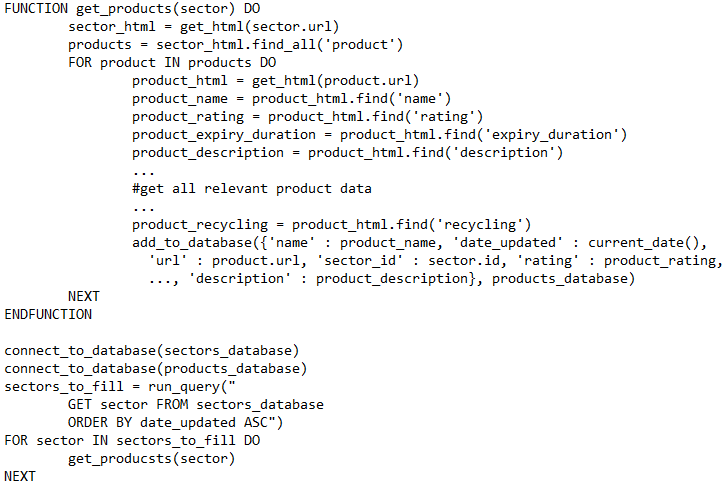
There is a one-to-many relationship between products and sectors between ‘sector\_id’ in the products database and ‘id’ in the sectors database. The databases are normalised to remove data redundancy, with all fields depending only on the ‘id’ column in each database

Then I will have to populate the database which I will do using 2 python scripts. The first will collect the data of all the sectors on the website and the second collect all the products for a given sector. Then by running the second program for each sector found by the first program I can populate the databases with all the products on the website. I created flow charts for each program to plan how they will work:

Program 1 – get sectors:



Program 2 – get products:



The final database is the ‘emissions\_stats’ database. This is initially the SuEatable food emissions database I downloaded from the internet, which was a result of extensive research done by the company. It initially contains the names of 325 products with the min, median and max emissions of production for that product in kg CO2 per kg food, but I will only use the median emissions for my calculations. I will have to do some modification to the database later to enable matching of each product in the products database with a row in this database. This database is constant, it will never be modified later

**‘Emissions\_stats’ database:**

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Data type | Description | Example |
| id | int | Unique ID of food type | 0 |
| name | string | Name of food type | “Beef” |
| emissions | float | kg CO2 per kg food type | 25.75 |

### Emissions algorithm

In order to calculate emissions of various food products, I had to research how various factors affect environmental impact. The three factors which produce emissions are production, which is determined by the type of food, transportation, and packaging

#### Production:

Firstly, I looked for existing research on the base impact of producing various food, and I found an article and database produced by scientists at SU-EATABLE[[3]](#footnote-3) with data on Carbon Footprints on 3350 products of 325 types. For each sector in my database, I can then link the sector to a row in the SU-EATABLE database to give the production emissions of the food type.

#### Transport:

Map

Description automatically generatedThe Sainsbury’s and Tesco’s websites have data on place of origin for most products, and I can use this to calculate the emissions of transport. I can create a list of all the Tesco’s and Sainsbury’s distribution centres and their coordinates using Google Maps (as below) and then use python to find the closest one to a user’s current location using ‘geocoder’ and ‘geopy’ python modules. From some internet research I found out that all products produced outside the UK are transported in by ship or by air, and I was able to obtain a list of products which are transported in each of these two methods. For transport by sea, I can assume the products are being transported to the nearest port from the place of origin and transported by road to the distribution centre and then to the user’s location, while for air-freighted goods I will assume they are flown directly to the distribution centre and then driven to the user location. I can calculate these distances, and then using the mass of the product and the emission coefficients (kilograms CO2 emitted per kilometre per kilogram) for each transport type I can calculate the total emissions due to transportation for each product. The distances will not be completely accurate due to the specific shapes of roads and coasts, but I think they will be accurate enough to form a useful estimate.

Map of UK Sainsbury’s distribution centres

I can use known values for the coefficients for transport type in the calculations:[[4]](#footnote-4)

|  |  |  |
| --- | --- | --- |
| **Transport type** | **Emission coefficient** | |
| **Ambient** | **Temperature-controlled** |
| **Road** | 0.20 | 0.50 |
| **Sea** | 0.01 | 0.02 |
| **Air** | 1.13 | 1.13 |

The general formula for transport emissions is:

As in the table, the coefficients are different for temperature controlled or ambient transport, and this will depend on the food type. The specific algorithms for each food depending on whether it needs to be temperature controlled and whether it is transported by air or sea will therefore be as follow:

|  |  |  |
| --- | --- | --- |
| **Transport emissions** | **Ambient** | **Temperature-controlled** |
| **Sea** |  |  |
| **Air** |  |  |

#### Cooking:

In my initial data collection, I will gather data where it is available on the cooking instructions for each product. Then I can find estimates on the internet for emissions caused by each cooking method (microwave, oven, etc.) and how the cooking time affects these emissions. However I don’t think enough products will have this information to be useful, so I will first collect the data and then decide whether or not to implement an algorithm

#### Pseudocode:

These 3 components: food, transport, and cooking, make up the emissions of each product, and so I will write another program to combine these and calculate the total emissions for each record in my products database, although my initial design will not include cooking emissions as previously discussed. Emissions per kg, per portion and per calorie are also important metrics which people will want to compare the products by, so the program should also calculate these and put the information in the database. I also realised that the road emissions for each product will be effectively identical per kg, so I can ignore this part since the primary purpose of the program is to compare different items

The full pseudocode is shown below, broken down into sections:

Firstly, I set up the constants and data for the program:

Text

Description automatically generated

* **country\_list** is a list of all the countries in the world in lowercase and in the format used by the **geopy** python module
* **emissions** is a table of the emissions rates of various mode of transport in kg CO2 per kg of item per km travelled, stored as a dictionary of strings to lists
* **air\_food** is a list of all food items which are transported by air rather than sea
* **temp\_controlled\_food** is a list of all food items which need to be temperature controlled during transport
* **current\_location** is the country of residence of the user, for now this is always set to ‘united kingdom’ since the websites I am catering for only operate in the UK

Next, I define some functions for my program:

Text

Description automatically generated

* **get\_country\_coords** returns the geographical coordinates of the centre of a given **country** using the **geopy** python module. The country name must be in the format used by geopy
* **get\_countries\_distance** returns the geographical distance in km between two countries using the **geopy** module. The names are these two countries are passed in the parameters **to** and **from**
* **get\_travel\_emissions** returns the emissions caused by the transport of a food item of mass **mass**, travelling a distance **dist**, by a mode of transport **transport** (sea or air). The parameter **temp** is a boolean value indicating whether or not the product needs to be temperature controlled during transport

Then I get the full **products** data from Heroku:

Text

Description automatically generated

* This query selects all the products from the **products** database along with the ID of the name of the sector they are in
* A **LEFT JOIN** is used to get the sectors name by linking the **products** and **sectors** databases with the sector\_id
* This program as well as the program to update the **sectors** and **products** databases will be run on a regular e.g. hourly basis, so the query only returns rows where the emissions have not already been calculated (line 24)
* In line 26, **execute\_query** is a function which executes an SQL query (passed as a parameter) on Heroku and has access to my databases. The function returns the result of running the query as a pandas dataframe, and here this is stored in **result**

Then I get the **emissions\_stats** database:



* The query selects the whole **emissions\_stats** database
* **execute\_query** executes the query and stores the result in **links**

For each product, I then calculate the emissions. This first section begins to calculate transport emissions by finding the possible places of origin of the product:

Text

Description automatically generated

* Line 31 loops through each product in **result**
* **transport\_emissions** will store the emissions caused by the transport of this product. It is initially set to -1 so if errors are thrown later and this value is not updated, -1 signifies this error and the program will act accordingly
* In line 33, the ‘origin\_country’ field of the given product is made lower case and split up into a list of words, which is stored in the variable **origin\_country**. For example, the string ‘this product is grown in France Germany or the United States’ is converted to the list [‘this’, ‘product’, ‘is’, ‘grown’, ‘in’, ‘france’, ‘germany’, ‘or’, ‘the’, ‘united’, ‘states’]
* In line 34, **product\_countries** is set to an empty list. This will later store the list of countries the product can be produced in
* In lines 35-36, the program loops through each combination of up to 5 consecutive words in the **origin\_country**. It only checks up to length 5 since this is the number of words in the longest country name
* In line 37, the set of words selected is joined by spaces, for example some values of **new\_string** during the execution of this selection on the above data would be ‘this’, ‘product’, ‘is’, …., ‘this product’, ‘product is’, …, ‘this product is’,…
* In lines 38-39, if **new\_string** is an item in **country\_list**, then this string is a country and the country is added to **product\_countries**. On the above data, the if statement will be true when new\_string equals ‘france’, ‘germany’, or ‘united states’, and so after this section is run, **product\_countries** will have the value [‘france’, ‘germany’, ‘united states’]

Then I convert the list of countries or origin into an average total transport emissions figure:

Text

Description automatically generated

* Line 43 checks if **product\_countries** is empty. If it is, the following code shouldn’t run otherwise an error will be thrown. This means if no countries are found, no transport emissions will be calculated and **transport\_emissions** will keep the default value -1
* Line 44 uses a list comprehension to convert each country in **product\_countries** into a distance from the user’s location using the **get\_countries\_distance** function, and then takes the average value of this list, hence finding the average distance the food item will travel from production to delivery
* **transport\_type** and **temp\_controlled** store the type of transport which this food is transported by and whether or not it needs to be temperature controlled respectively. They are initially set to default values: ‘sea’ and False
* Lines 46-50 check if any of the items in **air\_food** are in the name of the product, and if so, change **transport\_type** to air. E.g. if the product is called ‘Sainsburys green beans 200g’, and ‘green beans’ is in air\_food, them transport\_type will be set to ‘air’. The lower() function is used to make all the information lower case
* Lines 51-55 check if any of the items in **temp\_controlled­\_food** are in the name of the product, and if so, change **temp\_controlled** to True
* In lines 56-59, if mass data is available for the product, then the transport emissions are calculated using **get\_travel\_emissions()** and stored in **transport\_emissions**

Next, I calculate the production emissions of the product: Text

Description automatically generated

* **food\_emissions** is initially set to -1 as a default value
* Again, the data is only calculated if mass data is available
* Lines 59-63 see if the product matches any of the records in the **links** table, and if so the corresponding emissions per kg data is retrieved
* This is multiplied by the mass of the product to calculate production emissions

The emissions data is then combined and uploaded to the database on Heroku

Text

Description automatically generated

* Line 71 checks if both the **transport\_emissions** and **food\_emissions** were calculated as expected
* **total\_emissions** is the sum of the **transport\_emissions** and **food\_emissions**
* **update\_query** is an SQL query to set the ‘emissions’ field of the relevant row in the **products** database to **total\_emissions**.
* Line 78 runs the query to modify the table on Heroku

Once the program has looped through all the products, other metrics in the database are updated

Text

Description automatically generated

* **update\_query** is an SQL query to calculate the **emissions\_per\_kg**, **emissions\_per\_portion** and **emissions\_per\_calorie** columns in the database
* The **nutrition\_100[‘calories’]** value is the calories per 100g, so it is multiplied by 10 to find calories per kg, and then divided from the emissions multiplied by the mass to find the emissions per calorie
* In practice, this part will be more complicated for 3 reasons:
  + There may be divide by zero errors if the **mass**, **pack\_servings**, or **nutrition\_100[‘calories’]** data was not available, so these need to be checked for each field
  + The **nutrition\_100** field is actually stored in the form of a dictionary inside string, so this needs to be unpacked
  + **nutrition\_100** will often incomplete, and may not have calorie data, so I need to make sure the correct data is being used

### Suggestion algorithm

I need to only show relevant items which the user wants to see. I will try to have 2 modes of input, where the user can either search by a URL or for a food type (e.g. ‘Chicken’).

In the case of searching by food type, I can simply show all products where each word in the search term is either in the name of the product or the name of the sector. This can be done via an SQL query. For example, searching for ‘pork sausage’ would return ‘Pork sausages’, ‘Sausages made of pork’, and a ‘Sausages’ product in the ‘pork’ sector. I also may want to give the option for a user to manually toggle whether certain sectors are shown after the search, as for example searching for ‘raspberries’ might show strawberries in the ‘raspberries and strawberries’ section, so the user should be able to manually filter the sectors.

For searching by a URL, the user copies in a link to a product and then the program should show all products which are similar. The easiest way to do this would simply be to show all products in the same sector, and I think this would an effective suggestion technique. The user could also have the option as to what level of sector the search is in, e.g. if the product searched is in the sector ‘Dairy, eggs & chilled > Dairy and eggs > Cheese > Soft and cottage cheese’, the user could select whether to show products in ‘Soft and cottage cheese’, ‘Cheese’, ‘Dairy and eggs’, or ‘Dairy, eggs & chilled’, depending on how much of a change the user is willing to make

### User interface

**GUI plan and design**

# Development

Now that the various sections of my product have been designed and planned, I can begin on the various programs I need to write. My first step is to write the program to populate the sectors database.

## Create databases

//

## Populate sectors



* Lines 1-2 imports required modules
* **psycopg2** is an SQL database adapter for python, allowing me to send queries and receive data from the Heroku database
* **pandas** is a library for storing databases as Dataframes in Python. Used to store the database in the program before uploading to Heroku
* Text

  Description automatically generated
* **postgres\_connect()** is a function to connect to the Heroku database and return any errors that are encountered
* lines 5-8 contain my Heroku account details so are blocked out. This information will not be visible for 2 reasons. Firstly, the data will be uploaded to the database privately, and only another script which accesses the database will be distributed. Furthermore, the source code will not be distributed so even that code will not be available
* line 12-15 try to connect to Heroku and returns the connection, if there is an error this is handled in lines 16-20
* Text

  Description automatically generated
* **postgres\_execute()** runs an SQL query on the database
* Lines 27-29 create a connection with the database and run the query
* Lines 30-33 get the result of the query and format it as a pandas DataFrame
* In line 36, if there is no result the result is set to an empty DataFrame
* Lines 37-40 commit any updates to the database, close the connections and return the result
* Text

  Description automatically generated
* **heroku\_upload()** uploads a DataFrame to the sectors database on Heroku into **sainsburys\_sectors**
* It runs an SQL insert query, using text formatting to insert the values in the DataFrame into the values in the query
* Lines 58-61 create a connection and commit the upload
* Once I had written these functions, I realised that I would need the same functions again later for updating the products database and running the GUI, so in order to minimise code duplication I moved these functions to a file called Heroku\_functions.py. Then I could import these functions from the file and begin to write the rest of the code in a new file:
* Text

  Description automatically generated
* Lines 1-6 import required modules
* **requests** and **bs4** are used to parse information from the Sainsbury’s website to collect the sectors
* **pandas** is used to store the database before uploading to Heroku
* **Heroku\_functions** is the file storing **postgres\_execute** and **postgres\_connect**. Although I later realised postgres\_execute was not actually used for this program since no data is retrieved from a database
* **datetime** and **time** are used to get the current date and time to fill the **date\_updated** column of the database
* Text

  Description automatically generated
* I then also realised that **heroku\_upload** had to be in the file itself since it contains the name of the table and fields to be inserted, so must be different for each file
* I also removed the parameter and replaced references to the parameter with **leaf\_sectors** (defined below) since it is a global variable
* Text

  Description automatically generated
* This section uses the BeautifulSoup library to get the HTML from the Sainsbury’s homepage and sets up the key data structures
* **bad\_urls** is initially an empty list and will be a list of any URLs which produce an error during execution of the code. I can then output these once the code has run so I can manually check the causes of these errors
* **soup** will store the soup object containing the HTML of the page
* Lines 35 and 36 use the **soup.find()** function to get all top level sectors on the Sainsbury’s page and store a list of their name and link into **all\_sectors**
* **leaf\_sectors** is a pandas DataFrame that will store all the details about any leaf sectors (sectors in the tree which have no children). E.g. the ‘Meat & fish’ sector is not a leaf node since it has child sectors: ‘Meat & fish>Chicken’, ‘Meat & fish>Lamb’, ‘Meat & fish>Beef’ etc., while ‘Meat & fish>Chicken>Coooked chicken’ is a leaf sector since it has no sub-sectors)
* Text

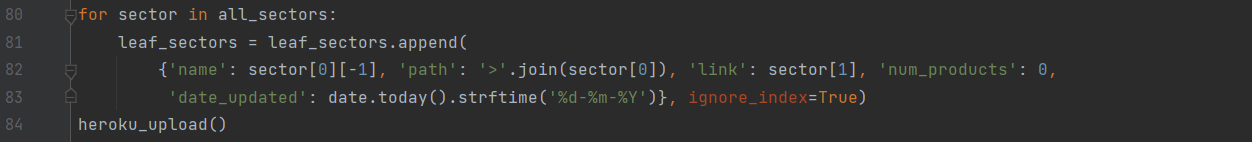
  Description automatically generated
* The Sainsbury’s website’s tree is organised into 3 layers: departments, aisles, and shelves. So in line 39 I loop through each of these layers, each time finding all leaf nodes and adding any non-leaf nodes to all\_sectors. This will form a **breadth-first traversal** to find all leaf nodes in the tree. The various layers act as the recursion depth for the traversal algorithm
* Lines 40-41 print updates on the progress of the program so I can track how long it will take and spot any errors as the program is run
* **new\_sectors** stores the children of all the nodes in the current layer. For example when looking through departments, all the aisles within these departments will be stored in new\_sectors, and then once all departments have been looked through, all\_sectors will be replaced with new\_sectors to form the new queue for the next layer
* Line 43 loops through each sector in the queue, within the loop the sector will be checked to see if it is a leaf node, or if not its children will be found
* Line 44 prints the path of the sector to help track execution progress
* In line 45 **URL** is set to the url of the current sector being checked
* Text

  Description automatically generated
* Lines 53-55 try to access the given URL and store the page in **page**. Initially line 54 was the only line in this section but in my initial testing, the site occasionally shut down while I was accessing it, causing an error to be thrown
* Because of this, I added a try-except clause in a while loop to catch any error and reboot the connection up to 3 times for each URL before giving up and setting **bad\_url** to True

Text

Description automatically generated

* If the URL was accessed, then **bad\_url** will be False so the condition in line 66 will be satisfied and the rest of this section executed
* Line 67 gets the HTML of page of the current sector to be checked and stores into **soup**
* Line 68 gets all the children of the current sector and stores them into **new**
* If children are found, then a for loop is used to loop through all the children and append them to **new\_sectors**
* If no children are found, the sector is added to the **leaf\_sectors** DataFrame with the relevant information:
  + **sector[0]** contains the path of the current sector in list form e.g. [‘Meat & Fish’, ‘Chicken’, ‘Cooked chicken’]
  + **sector[0][-1]** therefore contains the name of the current sector e.g. ‘Cooked chicken’, so this is the value of the ‘name’ field
  + The value of the ‘path’ field is set to the list sector[0], joined by a ‘>’ character e.g. ‘Meat&Fish>Chicken>Cooked chicken’
  + **sector[1]** contains the URL of the current sector, so this is the value of the ‘link’ field
  + ‘num\_products’ is set to its default value of 0, this will contain the number of products in the sectors, but we do not know this information until we run the next program I will write
  + ‘date\_updated’ is set to the current date in dd-mm-yyyy form using the date module in python
  + The ignore\_index=True parameter stops pandas automatically adding a numerical id to the dataframe, since the path is a unique key within the program, and the database on Heroku contains an automatically generated uuid
* The program waits for 1 second to reduce the chance of the website closing the connection
* all\_sectors is set to new\_sectors as the new queue once the traversal of that layer has been completed



* The final layer (shelves) will never be traversed since they are all leaf nodes, so all of the shelves are added to leaf\_sectors in the same way as sectors were appended previously
* The data is uploaded to the Heroku database

Text

Description automatically generated

* Here you can see the start of the execution of the program as it goes through the departments and begins to look through aisles

However, a few URLs were causing issues, and I was able to track using the bad\_urls list that this was caused by some inconsistencies on the website where some links were formatted without the ‘sainsburys.co.uk’ part, so I added the following code to correct this error:



I then ran the code, and the data was uploaded to Heroku, as can be seen below:

**//Add image of Sainsburys\_sectors**

[**https://www.bieroundtable.com/wp-content/uploads/49d7a0\_7a5cfa72d8e74c04be5aeb81f38b136b.pdf**](about:blank)

1. Estimated from data at [https://www.statista.com/forecasts/997903/online-grocery-shopping-by-store-brand-in-the-uk](about:blank) and https://www.worldometers.info/world-population/uk-population/ [↑](#footnote-ref-1)
2. \* Only on own-brand products [↑](#footnote-ref-2)
3. https://www.nature.com/articles/s41597-021-00909-8 [↑](#footnote-ref-3)
4. https://ourworldindata.org/food-transport-by-mode#:~:text=Transporting%20food%20by%20air%20emits,CO2eq%20by%20air. [↑](#footnote-ref-4)