Software Design Document

Project Name: Data Analysis and Visualisation
Group Number: 015
Team members

Student No.	Full Name
s5387195	Ethan Davis
s5347877	Tristan Martins
s5294045	Ethan Baker

Table of Contents

1. System Vision

1.1 Problem Background

1.1.1 Problem Identification: What problem does this system solve?

The 'Nutritional Food Comparison' system, including its features like Food Search, Nutrition Breakdown, Nutrition Range Filter, Nutrition Level Filter, and the additional Meal Planner feature, aims to solve the following problems:

- **1)** Complexity in Nutritional Data Analysis: Nutritional information can be overwhelming and difficult to interpret manually. The system simplifies this process by providing tools that enable users to search for foods, analyse their nutritional content, and create personalised labels.
- **2)** Inaccessible Visualisation: The system makes nutritional data more accessible by incorporating simple visual elements like pie charts, bar graphs, and detailed nutritional breakdowns.
- **3)** Efficient Decision-Making: Tools such as the system's filtering and range selection allow users to quickly make informed food choices based on nutritional content and dietary requirements.
- 4. Personalised Meal Planning: The Meal Planner feature, a unique and valuable component of the 'Nutritional Food Comparison' system, encourages users to create balanced meals by combining foods and analysing their nutritional value. This aids in achieving dietary goals such as weight management or nutrient intake optimisation.
- 5) Time and Effort Reduction: The system automates analysing and comparing nutritional data, reducing the time and effort required to plan meals and make healthy food choices.

1.1.2 Dataset: What is the dataset used?

The Nutritional Food Database provides a comprehensive dataset showing detailed nutritional information for various food items. Each food item in the dataset is analyzed based on various nutritional parameters, making it a valuable resource for dietary and health inquiries.
The Column descriptions that are used in the dataset are listed below:

Dataset	Description
Food	The name or type of the food item.
Caloric Value (in kcal)	Total energy provided by the food, typically measured in kilocalories per 100 grams.
Fat (in g)	Total amount of fats in grams per 100 grams, including the breakdowns that follow.
Saturated Fats (in g)	Amount of saturated fats (which can raise cholesterol levels) in grams per 100 grams.
Monounsaturated Fats (in g)	Amount of monounsaturated fats (considered heart-healthy fats) in grams per 100 grams.
Polyunsaturated Fats (in g)	Amount of polyunsaturated fats (essential fats needed by the body) in grams per 100 grams.
Carbohydrates (in g)	Total carbohydrates in grams per 100 grams, including sugars.
Sugars (in g)	Total sugars in grams per 100 grams, a subset of carbohydrates.
Protein (in g)	Total proteins in grams per 100 grams, essential for body repair and growth.
Dietary Fiber (in g)	Fiber content in grams per 100 grams, important for digestive health.
Cholesterol (in mg)	Cholesterol content in milligrams per 100 grams, pertinent for cardiovascular health.
Sodium (in g)	Sodium content in milligrams per 100 grams, crucial for fluid balance and nerve function.

Dataset	Description
Water (in g)	Water content in grams per 100 grams, which affects the food's energy density.
Vitamin A (in µg)	Amount of Vitamin A in micrograms per 100 grams, important for vision and immune functioning.
Vitamin B1 (Thiamine) (in mg)	Essential for glucose metabolism.
Vitamin B11 (Folic Acid) (in mg)	Crucial for cell function and tissue growth, particularly important in pregnancy.
Vitamin B12 (in µg)	Important for brain function and blood formation.
Vitamin B2 (Riboflavin) (in mg)	Necessary for energy production, cell function, and fat metabolism.
Vitamin B3 (Niacin) (in mg)	Supports digestive system, skin, and nerve health.
Vitamin B5 (Pantothenic Acid) (in mg)	Necessary for making blood cells and helps convert food into energy.
Vitamin B6 (in mg)	Important for normal brain development and maintaining the nervous and immune systems.
Vitamin C (in mg)	Important for the repair of all body tissues.
Vitamin D (in µg)	Crucial for the absorption of calcium, promoting bone growth and health.
Vitamin E (in mg)	Acts as an antioxidant, helping to protect cells from damage caused by free radicals.
Vitamin K (in µg)	Necessary for blood clotting and bone health.
Calcium (in mg)	Vital for building and maintaining strong bones and teeth.
Copper (in mg)	Helps with the formation of collagen, increases iron absorption, and plays a role in energy production.
Iron (in mg)	Essential for the creation of red blood cells.
Magnesium (in mg)	Important for regulating muscle and nerve function, blood sugar levels, and blood pressure.
Manganese (in mg)	Involved in bone formation, blood clotting, and metabolism of fats and carbohydrates.
Phosphorus (in mg)	Helps with the formation of bones and teeth and is necessary for cell growth and repair.
Potassium (in mg)	Helps regulate fluid balance, muscle contractions, and nerve signals.
Selenium (in µg)	Important for reproduction, thyroid function, DNA production, and protecting the body from free radicals.
Zinc (in mg)	Necessary for immune function, cell division, growth, wound healing, and carbohydrate breakdown.
Nutrition Density	A metric indicating the nutrient richness of the food per calorie.

1.1.3 Data Input/Output: What kind of data input and output is required?

Ensuring that both input and output mechanisms are intuitive and robust will enhance the user experience and accuracy of the 'Nutritional Food Comparison' project. The data input and output requirements are essential for the functionality and usability of the application. Below is a detailed breakdown:

Data Input

- User Input for Food Search: Search Queries: This is text-based input where users can search for food items by name. This might include partial names or exact matches.
    - Filters: Criteria for filtering data, such as: Nutritional Langes: Users can select one of the nutrition and input minimum and maximum values, and the tool will display a list of foods that fall within those ranges. Nutritional Levels: Users can filter foods by their nutritional content

levels—low,
mid, and high—including fat, protein, carbohydrates, sugar, and nutritional density.
- Low: Less than 33% of the highest value.
- Mid: Between 33% and 66% of the highest value.
- High: Greater than 66% of the highest value.

- Upload and Intergration: CSV File Upload: This option allows administrators to upload a CSV file containing nutritional data, which the system must parse and integrate into the database.

Data Output
- Search Results:
  - List of Food Items: Display search results based on the user's query, including relevant nutritional information. Detailed Information: Provide detailed nutritional data for each food item, such as calories, fat content, vitamins, etc.
- Nutritional Breakdown:
  - Tables: Detailed tables showing the nutritional breakdown of selected food items.
  - Graphs and Charts: Visual representations like bar charts. Pie charts or line graphs to illustrate comparisons between different foods or nutrient levels.
  - Comparison Tables: Allow users to compare nutritional values of multiple food items side by side.
  - Summary Statistics: Provide summary statistics or insights based on user queries, such as average calories, highest protein content, etc.
- User Interface (UI):
  - Interactive Elements: Input fields for searching, filtering, and sorting (Ascending/Descending), as well as buttons for generating reports or viewing detailed comparisons.

1.1.4 Target Users: Who will use the system, and why?

The 'Nutritional Food Comparison' system is designed to cater to a range of users who have different needs related to dietary management, health, and nutrition. Below is a detailed breakdown of the projected Proto- Personas.

Proto Persona 1: Fitness Enthusiast

Attribute	**Description**
Name	Alex
Age	28
Goal	Balanced meal planning
User Needs	Quick access to nutritional data for planning meals

---

Proto Persona 2: Professional Nutritionist

Attribute	**Description**
Name	Dr. Emma
Age	45
Goal	Personalized client meal plans
User Needs	Detailed nutritional info and tools for creating meal plans

---

Proto Persona 3: Health-Conscious Parent

Attribute	**Description**
Name	Sarah
Age	35
Goal	Planning meals for family

Attribute	Description
User Needs	Focus on vegan options for family meals

Proto Persona 4: Elderly Individual

Attribute	Description
Name	John
Age	70
Goal	Maintain a balanced diet for healthy aging
User Needs	Low-sodium/sugar and heart-healthy meal options

Proto Persona 5: College Student

Attribute	Description
Name	Emily
Age	21
Goal	Affordable and nutritious meals
User Needs	Quick and easy meal planning with budget-friendly ingredients

---

Proto Persona 6: Busy Professional

Attribute	Description
Name	David
Age	34
Goal	Efficient meal planning amidst a busy schedule
User Needs	Quick, grab-and-go meal options with high nutritional value

---

Proto Persona 7: Young Athlete

Attribute	Description
Name	Mia
Age	18
Goal	Optimized nutrition for peak athletic performance
User Needs	High-protein and energy-rich meal options

The **Nutritional Food Comparison** system is designed to serve a wide range of users, including fitness enthusiasts, health-conscious parents, professional nutritionists, food manufacturers, elderly individuals, busy professionals, young athletes, and the general public. All these users share a fundamental need for reliable and easily accessible nutritional information to help them achieve their specific health and dietary goals.

The system offers personalized tools and resources for meal planning, nutritional analysis, and informed decision-making. By empowering users to make choices that align with their lifestyles and dietary needs, it supports a variety of objectives. Whether someone is looking to maintain a balanced diet, enhance athletic performance, or ensure product compliance, the system provides tailored solutions to meet these diverse requirements.

1.2 System capabilities/overview

System Functionality:
The 'Nutritional Food Comparison' desktop application is built with a user-friendly interface, aimed at simplifying the management and analysis of nutritional information. It effectively addresses challenges such as data interpretation complexity and the need for accessible visualization. Users can quickly search for food items and analyze their nutritional content, enabling efficient access to essential dietary information.
The application allows users to apply filters based on nutritional ranges and levels, tailoring search results to meet specific dietary needs—such as identifying low-sodium or high-protein foods. Additionally, it offers

detailed nutritional breakdowns that are easy to understand, enhanced by visual aids like tables. The system supports meal planning by allowing users to combine various foods and assess their overall nutritional values, assisting in achieving dietary goals such as weight management or optimized nutrition for athletic performance.

Features and Functionalities:
The 'Nutritional Food Comparison' system includes several key features to enhance the user experience:

- Food Search Tools: Users can easily locate food items and quickly retrieve detailed nutritional data.
- Nutrition Range and Level Filters: Customizable searches based on specific criteria, such as calorie content or nutrient levels, allow for more tailored results.
- Meal Planner: A standout component that helps users create balanced meals by analysing the combined nutritional content of selected foods.

Designed to cater to a diverse audience—from fitness enthusiasts to professional nutritionists—the system empowers users to make personalized, informed decisions about their dietary management. Furthermore, the application enables administrators to import data using CSV files, facilitating the easy integration of new nutritional information.

1.3 Benefit Analysis

The 'Nutritional Food Comparison' system provides significant value by streamlining the complex process of nutritional data analysis and meal planning, making it more accessible and actionable for users. Intuitive tools like the Food Search, Nutrition Filters, and Meal Planner benefit users by saving time and reducing the effort required to make informed dietary choices. By offering personalized insights and visual representations of nutritional information, the system helps users meet their health goals, whether managing weight, optimizing athletic performance, or ensuring a balanced diet. The ability to easily compare foods, and create customized meal plans, empowers users to take control of their nutrition with confidence and precision. Additionally, the system's user-friendly interface and comprehensive data integration capabilities ensure that users of all backgrounds, from busy professionals to health-conscious parents, can efficiently achieve their dietary objectives.

2. Requirements

2.1 User Requirements

The 'Nutritional Food Comparison' system is designed with a diverse range of users in mind, including fitness enthusiasts, nutritionists, busy professionals, and health-conscious individuals. These users are expected to interact with the program to achieve their dietary and nutritional goals efficiently. The system must provide the following functionalities from the end-user perspective:

Narrative Descriptions

1) Fitness Enthusiast "Alex": Alex, a 28-year-old fitness enthusiast, uses the system to plan meals that support his workout routines. He needs quick access to nutritional data to ensure his meals are balanced and aligned with his fitness goals. Alex frequently searches for high-protein foods, filters results based on specific nutrient levels and uses the Meal Planner to create meals that optimise his performance and recovery.

---

2) Nutritionist "Dr Emma": Dr Emma, a 45-year-old professional nutritionist, relies on the system to develop personalised meal plans for her clients. She needs detailed nutritional information and the ability to compare various food items to meet her clients' dietary needs. The system's export feature allows her to generate reports that can be shared with her clients, ensuring they have clear guidelines to follow.

---

3) Busy Professional "David": David, a 34-year-old professional, interacts with the system to quickly plan nutritious meals that fit his hectic schedule. He uses the system's filters to find grab-and-go meal options that are high in nutrients and low in unhealthy fats. The ability to save and export meal plans allows him to stay organised and maintain a balanced diet despite his busy lifestyle.

Search and Filter Functionality:

- Users need the ability to search for specific food items using text-based queries.
- The system must provide filters for nutritional content, allowing users to search based on calories, protein, fats, vitamins, and other nutrients.
- Filters should include ranges (e.g., low, medium, high) to help users find foods that match their dietary requirements.

Detailed Nutritional Breakdown:

- Users need access to comprehensive nutritional information for each food item, presented in an easy- to-understand format.
- Visual tools such as comparison tables should be available to help users interpret the data.

Data Integration and Export:

- Admin must be able to upload their data, such as CSV files with nutritional information, to customise the database.

Meal Planning Tools:

- The system must provide a Meal Planner feature that allows users to combine various foods and evaluate the overall nutritional content of the planned meal.
- Users can select and foods to add to their meal plans, adjusting them as needed to meet their goals.

User-Friendly Interface:

- The system must have an intuitive interface that allows users of varying technical skill levels to navigate and efficiently utilise all features.
- Interactive elements, such as clickable buttons and easy-to-use filters, should enhance the user experience and efficiency.

2.2 Software Requirements

- R1.0 Food Search and Filtering:
  - R1.1) The system shall allow users to search for food items using a text-based query.
  - R1.2) The system shall provide filters for users to narrow down search results based on nutritional ranges (e.g., calories, fats, proteins).
  - R1.3) The system shall enable users to apply nutritional level filters (low, medium, high) for specific nutrients such as sodium, sugar, and cholesterol.
  - R1.4) The system shall display a list of food items that match the search criteria, including relevant nutritional information.
  - R1.5) The system shall allow users to sort food items in ascending or descending order, and the search will match filtered items.

- R2.0 Nutritional Data Presentation:
  - R2.1) The system shall provide a detailed nutritional breakdown for each food item, including metrics such as caloric value, fats, carbohydrates, proteins, vitamins, and minerals.
  - R2.2) The system shall present nutritional data in tables to compare multiple food items easily.
  - R2.3) To aid data interpretation, the system shall include visual representations of nutritional data, such as comparison tables.
  - R2.4) The system shall allow users to compare the nutritional values of multiple food items side by side.

- R3.0 Meal Planning:
  - R3.1) The system shall provide a Meal Planner feature that allows users to create meal plans by selecting multiple food items.
  - R3.2) The system shall calculate and display the total nutritional content of the selected foods within a meal plan.
  - R3.3) The system shall enable users to modify meal plans by adding or removing food items,

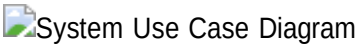
updating the nutritional content.

- R4.0 Data Integration and Export:
  - R4.1) The system shall allow administrators to upload CSV files containing nutritional data, which will be parsed and integrated into the system's database.

- R5.0 User Interface (UI) and Interaction:
  - R5.1) The system shall provide an intuitive user interface that supports easy navigation and use of all features.
  - R5.2) The system shall include interactive elements like buttons, dropdown menus, and input fields to facilitate user interaction.
  - R5.3) The system shall offer real-time feedback and updates as users search for foods, apply filters, or create meal plans.

These requirements define the essential functionalities of the 'Nutritional Food Comparison' system, ensuring a comprehensive and user-friendly experience for all intended users.

2.3 Use Case Diagram



2.4 Use Cases

Use Case ID: UC-01

Use Case Name	**Nutritional Database**
Actors	Administrator
Description	The administrator uploads nutritional data via CSV files and integrates it into the system's database.
Flow of Events	1) The administrator selects uploading a CSV file. 2) The system prompts the administrator to select a file from the local system. 3) The system validates and parses the data, integrating it into the database. 4) The administrator receives confirmation of a successful upload.
Alternate Flow	If the CSV file contains errors, the system prompts the administrator to correct and re-upload the file.

Use Case ID: UC-02

Use Case Name	**Search Filters**
Actors	User
Description	The user applies filters to narrow down food search results based on specific nutritional criteria.
Flow of Events	1) The user selects a mineral filter option (e.g., calories, protein, fat). 2) The user sets a range or level (low, medium, high) for the selected nutrient. 3) The system filters and displays the food items that meet the criteria.
Alternate Flow	The user can view the nutritional breakdown in different formats (e.g., pie chart, bar graph).

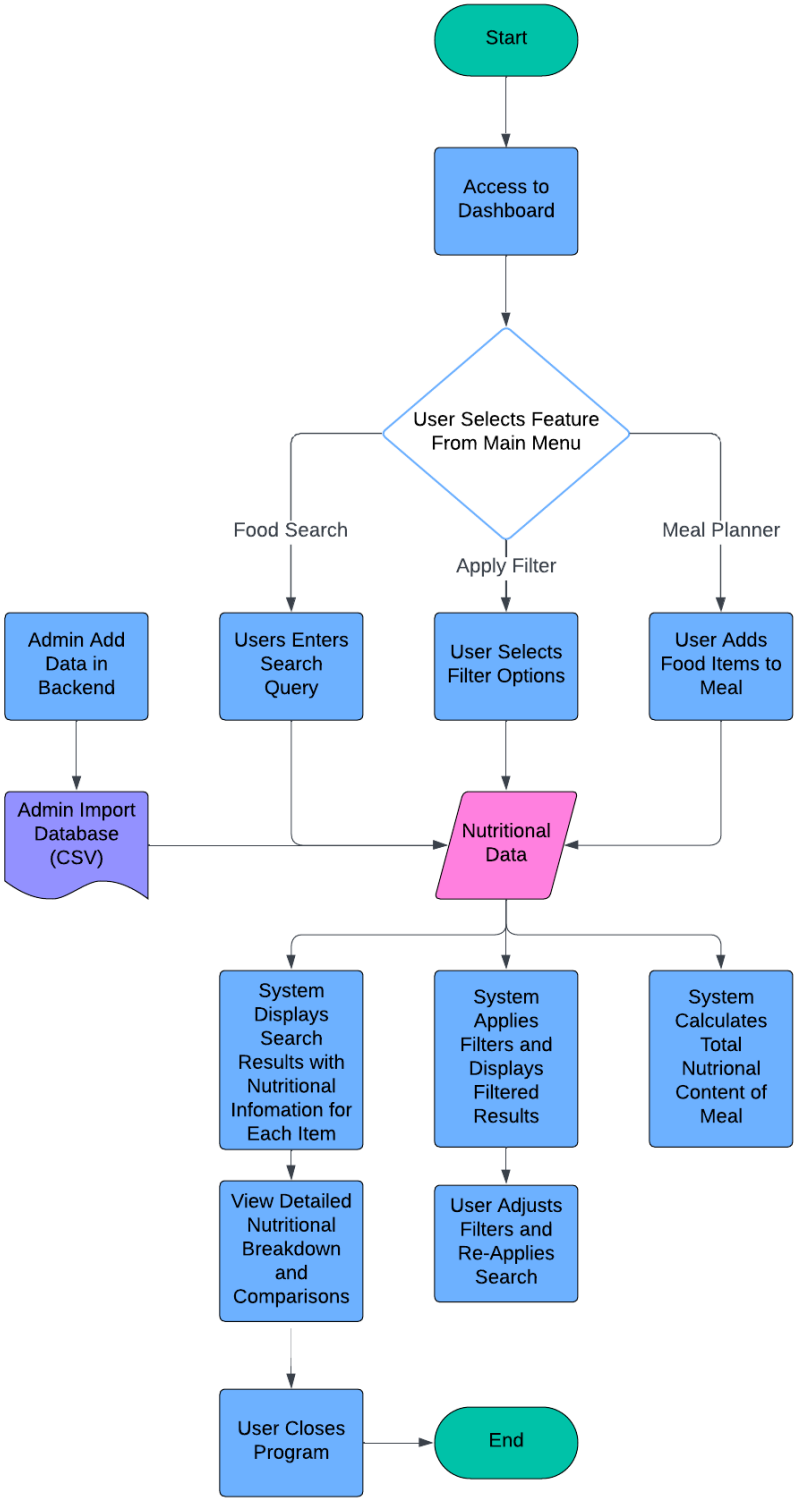
Use Case ID: UC-03

Use Case Name Nutritional Breakdown

Actors	User
Description	The user views a detailed report of nutritional information for selected food items.
Flow of Events	1) The user selects a food item from the search results. 2) The system displays a detailed nutritional breakdown in tabular and graphical formats. 3) The user can compare the breakdown with other food items.
Alternate Flow	The user can view the nutritional breakdown in different formats (e.g., pie chart, bar graph).

3. Software Design and System Components

3.1 Software Design



3.2 System Components

3.2.1 Functions

Start Program:
When the application starts, users can immediately begin interacting with its features. There is no login or registration required, allowing direct access to the app's functionalities without the need for account creation or authentication.
Dashboard:
Users are taken to the main dashboard, where they can navigate to key functions:

- **Food Search:** Users enter a search query to find food items, and the app displays detailed nutritional information for each item from the database.
- **Apply Filters:** Users can filter food items based on nutritional ranges, such as calories, protein, and fat content. Filtered results are displayed dynamically, and users can adjust the filters as needed to refine their search.
- **Meal Planner:** Users can add food items to a meal plan, where the app calculates the total nutritional values for the meal. This helps users see how their choices fit into their dietary needs.

Admin Import Functionality:
Instead of exporting or saving data locally, administrators have the option to upload new nutritional datasets through CSV files. These datasets are imported into the app’s database, ensuring that the latest food data is available to users.
End Program:
Once users have completed their tasks, they can simply close the app. There is no need for logging out since there is no account management feature, making the user experience more streamlined.

3.2.2 Data Structures / Data Sources

3.2.2.1 Food Database

Type: Dictionary (or Database Table)
Usage: Stores detailed nutritional information for each food item, where each food item is a key, and its corresponding nutritional data (calories, fats, proteins, etc.) is the value.
Functions:

- Search Function: Searches the food database for food items matching the search query.

```
filtered_data = self.dataset[self.dataset['food'].str.lower().str.contains(food_name, na=False)]
```

- Sort Order: Sorts the food items based on selected criteria (e.g., ascending or descending by nutrient):

```
sort_order = self.m_choice_sort.GetStringSelection()
if sort_order == "Ascending":
    filtered_data = filtered_data.sort_values(by=selected_vitamin)
else:
    filtered_data = filtered_data.sort_values(by=selected_vitamin, ascending=False)
```

- Level Filter: Filters the food items based on nutrient levels (e.g., Low, Medium, High):

```
level_filter = self.m_choice_level.GetStringSelection()
highest_value = filtered_data[selected_vitamin].max() if not filtered_data.empty else 0
low_threshold = highest_value * 0.33
high_threshold = highest_value * 0.66

if level_filter == "Low":
    filtered_data = filtered_data[filtered_data[selected_vitamin] < low_threshold]
elif level_filter == "Medium":
    filtered_data = filtered_data[(filtered_data[selected_vitamin] >= low_threshold) &
(filtered_data[selected_vitamin] <= high_threshold)]
elif level_filter == "High":
    filtered_data = filtered_data[filtered_data[selected_vitamin] > high_threshold]
```

- Populate Grid: Displays the filtered food items in the grid:

```
def populate_grid(self, filtered_data):
    self.m_grid2.ClearGrid()

    num_rows = len(filtered_data)
    num_cols = len(filtered_data.columns)

    if num_rows > self.m_grid2.GetNumberRows():
        self.m_grid2.AppendRows(num_rows - self.m_grid2.GetNumberRows())
    elif num_rows < self.m_grid2.GetNumberRows():
        self.m_grid2.DeleteRows(num_rows, self.m_grid2.GetNumberRows() - num_rows)

    if num_cols > self.m_grid2.GetNumberCols():
        self.m_grid2.AppendCols(num_cols - self.m_grid2.GetNumberCols())
    elif num_cols < self.m_grid2.GetNumberCols():
        self.m_grid2.DeleteCols(num_cols, self.m_grid2.GetNumberCols() - num_cols)

    for col_idx, col_name in enumerate(filtered_data.columns):
        self.m_grid2.SetColLabelValue(col_idx, col_name)

    for row_idx, row_data in enumerate(filtered_data.iterrows()):
        for col_idx, value in enumerate(row_data[1]):
            self.m_grid2.SetCellValue(row_idx, col_idx, str(value))

    self.m_grid2.Fit()
    self.Layout()
```

- On Row Click: Adds a food item to the current meal plan when the user clicks on a row in the grid:

```
def on_row_click(self, event):
    row_index = event.GetRow()

    food_data = [self.m_grid2.GetCellValue(row_index, col_index) for col_index in
range(self.m_grid2.GetNumberCols())]

    col_labels = [self.m_grid2.GetColLabelValue(col_index) for col_index in range(self.m_grid2.GetNumberCols())]

    if hasattr(self, 'open_window') and self.open_window is not None:
        self.open_window.append_row(food_data)
    else:
        self.open_window = FoodDetails(self, col_labels)
        self.open_window.append_row(food_data)
        self.open_window.Show()
```

3.2.2.2 Meal Plan

Type: List (of dictionaries)
Usage: Stores a list of food items selected by the user for a meal plan. Each food item is represented as a dictionary containing nutritional information.
Functions:

- Add to meal Plan: Adds a food item to the current meal plan.

```
self.append_row(food_data)
```

- Remove from Meal Plan: Removes a selected food item from the meal plan when the user clicks on the row.

```
self.on_row_click(event)
```

- Calculate Total Nutritional Value: Calculates the total nutritional value (calories, proteins, etc.) of all food items in the meal plan:

```
def update_totals(self):
    num_rows = self.grid.GetNumberRows() - 1
    num_cols = self.grid.GetNumberCols()

    totals = [0.0] * num_cols

    for row in range(1, num_rows + 1):
        for col in range(1, num_cols):
            try:
                value = float(self.grid.GetCellValue(row, col))
                totals[col] += value
            except ValueError:
                pass

    for col in range(1, num_cols):
        self.grid.SetCellValue(self.total_row_index, col, str(round(totals[col], 3)))

    for col in range(num_cols):
        self.grid.SetReadOnly(self.total_row_index, col, True)
```

3.2.2.3 Search Results

Type: List (of dictionaries)
Usage: Temporarily stores search results after the user enters a query. Each result is a dictionary containing the food item and its basic nutritional information.
Functions:

- Sort Order: Sorts the food items based on selected criteria (e.g., ascending or descending by nutrient).

```
sort_order = self.m_choice_sort.GetStringSelection()
if sort_order == "Ascending":
    filtered_data = filtered_data.sort_values(by=selected_vitamin)
else:
    filtered_data = filtered_data.sort_values(by=selected_vitamin, ascending=False)
```

- On Search: Searches Database based on filtered results.

```
def on_search(self, event):
    selected_items = self.m_choice1.GetStringSelection()
    food_name = self.m_textCtrl7.GetValue().lower()

    min_value = None
    max_value = None

    if self.m_textCtrl5.GetValue():
        try:
            min_value = float(self.m_textCtrl5.GetValue())
        except ValueError:
            wx.MessageBox("Please enter a valid numeric value for Min.", "Error", wx.OK | wx.ICON_ERROR)
            return

    if self.m_textCtrl6.GetValue():
        try:
            max_value = float(self.m_textCtrl6.GetValue())
        except ValueError:
            wx.MessageBox("Please enter a valid numeric value for Max.", "Error", wx.OK | wx.ICON_ERROR)
            return

    if selected_items == "Food":
        filtered_data = self.dataset[self.dataset['food'].str.lower().str.contains(food_name, na=False)]
    else:
        filtered_data = self.dataset[(self.dataset[selected_items].notna()) &
(self.dataset['food'].str.lower().str.contains(food_name, na=False))]

        if min_value is not None:
            filtered_data = filtered_data[filtered_data[selected_items] >= min_value]
        if max_value is not None:
            filtered_data = filtered_data[filtered_data[selected_items] <= max_value]

    self.populate_grid(filtered_data)
```

3.2.2.4 Nutritional Filters

Type: Dictionary

Usage: Stores the filters applied to search results. The dictionary keys represent the nutrient type (e.g., calories, protein), and the values represent the range or level.

Functions:

- Apply Filters: Filters the food items in the search results based on the selected ranges provided in the filter dictionary.

```
    def on_search(self, event):
        selected_items = self.m_choice1.GetStringSelection()
        food_name = self.m_textCtrl7.GetValue().lower()
        min_value = None
        max_value = None
        if self.m_textCtrl5.GetValue():
            try:
                min_value = float(self.m_textCtrl5.GetValue())
            except ValueError:
                wx.MessageBox("Please enter a valid numeric value for Min.", "Error", wx.OK | wx.ICON_ERROR)
                return
        if self.m_textCtrl6.GetValue():
            try:
                max_value = float(self.m_textCtrl6.GetValue())
            except ValueError:
                wx.MessageBox("Please enter a valid numeric value for Max.", "Error", wx.OK | wx.ICON_ERROR)
                return
        if selected_items == "Food":
            filtered_data = self.dataset[self.dataset['food'].str.lower().str.contains(food_name, na=False)]
        else:
            filtered_data = self.dataset[(self.dataset[selected_items].notna()) &
(self.dataset['food'].str.lower().str.contains(food_name, na=False))]
            if min_value is not None:
                filtered_data = filtered_data[filtered_data[selected_items] >= min_value]
            if max_value is not None:
                filtered_data = filtered_data[filtered_data[selected_items] <= max_value]
        sort_order = self.m_choice_sort.GetStringSelection()
        if sort_order == "Ascending":
            filtered_data = filtered_data.sort_values(by=selected_items)
        else:
            filtered_data = filtered_data.sort_values(by=selected_items, ascending=False)

        level_filter = self.m_choice_level.GetStringSelection()
        if level_filter != "None":
            highest_value = filtered_data[selected_items].max() if not filtered_data.empty else 0

            low_threshold = highest_value * 0.33
            high_threshold = highest_value * 0.66

            if level_filter == "Low":
                filtered_data = filtered_data[filtered_data[selected_items] < low_threshold]
            elif level_filter == "Medium":
                filtered_data = filtered_data[(filtered_data[selected_items] >= low_threshold) & (filtered_data[selected_items] <=
high_threshold)]
            elif level_filter == "High":
                filtered_data = filtered_data[filtered_data[selected_items] > high_threshold]

        self.result_count_text.SetLabel(f"Results found: {len(filtered_data)}")

        self.populate_grid(filtered_data)
```

- Clear Filters: While there is no set code to clear filters, in the drop-down menu there is options for ‘none’ for some of the options.

3.2.2.5 Nutritional Breakdown

Type: Dictionary

Usage: Stores detailed nutritional information for a selected food item. The dictionary contains key-value pairs representing nutrients (e.g., calories, fats) and their corresponding amounts.

Functions:

- Nutritional Data: Fetches detailed nutritional data for a food item.

```
def populate_grid(self, filtered_data):
    self.m_grid2.ClearGrid()

    num_rows = len(filtered_data)
    num_cols = len(filtered_data.columns)

    if num_rows > self.m_grid2.GetNumberRows():
        self.m_grid2.AppendRows(num_rows - self.m_grid2.GetNumberRows())
    elif num_rows < self.m_grid2.GetNumberRows():
        self.m_grid2.DeleteRows(num_rows, self.m_grid2.GetNumberRows() - num_rows)

    if num_cols > self.m_grid2.GetNumberCols():
        self.m_grid2.AppendCols(num_cols - self.m_grid2.GetNumberCols())
    elif num_cols < self.m_grid2.GetNumberCols():
        self.m_grid2.DeleteCols(num_cols, self.m_grid2.GetNumberCols() - num_cols)

    for col_idx, col_name in enumerate(filtered_data.columns):
        self.m_grid2.SetColLabelValue(col_idx, col_name)

    for row_idx, row_data in enumerate(filtered_data.iterrows()):
        for col_idx, value in enumerate(row_data[1]):
            self.m_grid2.SetCellValue(row_idx, col_idx, str(value))

    self.m_grid2.Fit()
    self.Layout()
```

- Compare Foods: While there is no ‘compare’ option, the foods are listed in a table, and users can see the total nutritional value of selected foods.

```
def update_totals(self):
    num_rows = self.grid.GetNumberRows() - 1
    num_cols = self.grid.GetNumberCols()

    totals = [0.0] * num_cols

    for row in range(1, num_rows + 1):
        for col in range(1, num_cols):
            try:
                value = float(self.grid.GetCellValue(row, col))
                totals[col] += value
            except ValueError:
                pass

    for col in range(1, num_cols):
        self.grid.SetCellValue(self.total_row_index, col, str(round(totals[col], 3)))

    for col in range(num_cols):
        self.grid.SetReadOnly(self.total_row_index, col, True)
```

3.2.2.6 CSV Data Source

Type: CSV File (External Source)
Usage: Administrators upload CSV files containing nutritional data to be integrated into the system.

Functions:

- upload_csv(file): Allows administrators to upload and import CSV files with nutritional data.

```
self.dataset = pd.read_csv('Food_Nutrition_Dataset.csv')
```

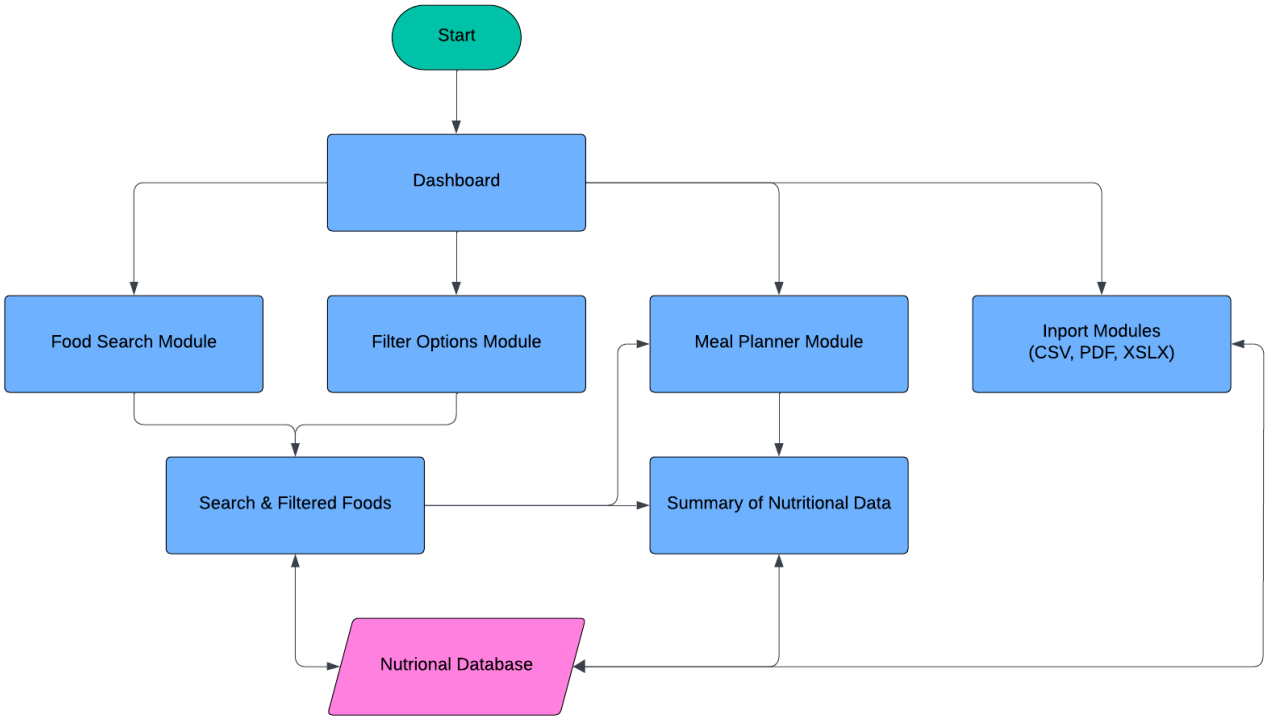
- parse_csv(file): Parses the uploaded CSV file and integrates it into the food database, ensuring the correct format.

```
self.m_grid2 = wx.grid.Grid(self, wx.ID_ANY, wx.DefaultPosition, wx.DefaultSize, 0)
self.m_grid2.CreateGrid(0, 9)
self.m_grid2.EnableEditing(True)
self.m_grid2.EnableGridLines(True)
self.m_grid2.SetMargins(0, 0)
self.m_grid2.EnableDragColMove(False)
self.m_grid2.EnableDragColSize(True)
self.m_grid2.SetColLabelAlignment(wx.ALIGN_CENTER, wx.ALIGN_CENTER)
self.m_grid2.EnableDragRowSize(True)
self.m_grid2.SetRowLabelAlignment(wx.ALIGN_CENTER, wx.ALIGN_CENTER)
self.m_grid2.SetDefaultCellAlignment(wx.ALIGN_LEFT, wx.ALIGN_TOP)

bSizer3.Add(self.m_grid2, 0, wx.ALL, 5)
```

4. User Interface Design

4.1 Structural Design



4.1.1 Functions

Start Program	The application starts with an intuitive interface, giving users immediate access to food-related search and nutritional information. Since no login or registration is required, users can directly begin searching for food items, making it easy for anyone to use the app right away.
Dashboard	The dashboard acts as the primary navigation hub, offering users quick access to features like food search, filtering, meal planning, and CSV data import (for administrators). All essential functions are readily available in one place, ensuring smooth navigation and efficient use of the application.
Food Search	Users can search for food items to view detailed nutritional information from the app's food database. The search is optimized for speed and efficiency, allowing users to quickly find foods based on criteria such as calories, fats, proteins, and other nutritional details.
Filter Options	This module allows users to filter food search results according to specific nutritional needs, such as limiting results to low-fat or high-protein options. These filtering tools provide a customizable experience, helping users tailor their searches to suit personal dietary requirements or preferences.
Meal Planner	The meal planner lets users create meal plans by adding selected food items. The app will then calculate the total nutritional value of the meal, helping users balance their intake and make informed decisions about their food choices.
CSV Data Import	While users cannot export data, administrators have the ability to import new nutritional datasets via CSV files. These datasets are integrated into the food database, ensuring that the nutritional information remains up-to-date.
Nutritional Database	This database stores all the nutritional details for the food items, providing the information displayed during food searches and meal planning. The data is updated via administrator-uploaded CSV files, ensuring accurate and comprehensive information.
Filter Options	After performing a search, users can apply filters to refine their results based on their nutritional preferences. The system will display the filtered foods, allowing users to focus on items that meet their dietary goals.
Grid Population	After filtering or creating a meal plan, users are presented with a summary of the total nutritional content of their selected items. This information helps users evaluate their choices and ensure that their nutritional intake aligns with their goals.

4.1.2 Navigation & Design Choices

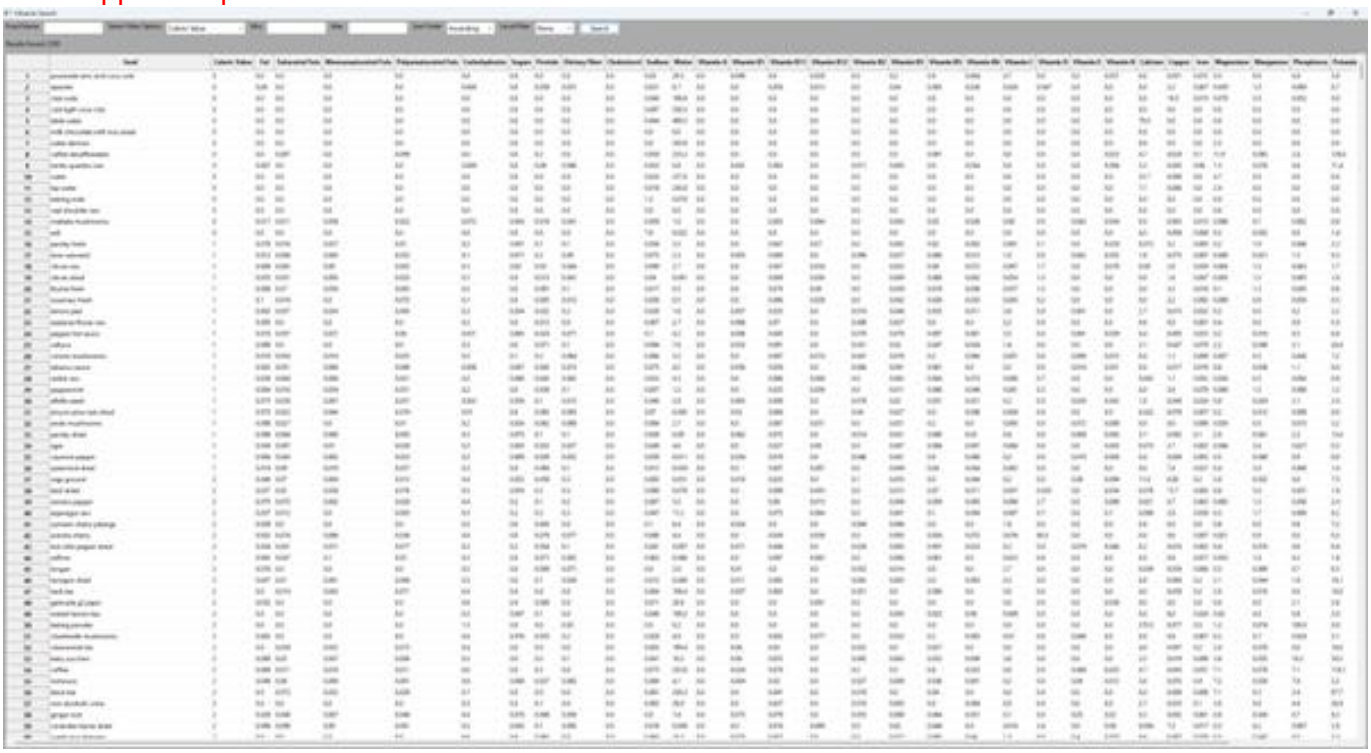
Simplicity: The app's design prioritizes ease of use, offering a straightforward interface without any complicated login or registration steps. Users can access all key functions—such as food search, filtering, and meal planning—directly from the dashboard. Each module is clearly defined and accessible, ensuring a smooth experience for users as they navigate the app.

Mobile & Desktop Compatibility: While primarily developed for desktop environments, the app's design ensures it remains accessible and usable across both desktop and mobile devices. The layout is clean and responsive, allowing users to engage with the app's features effortlessly, regardless of the platform they are using.

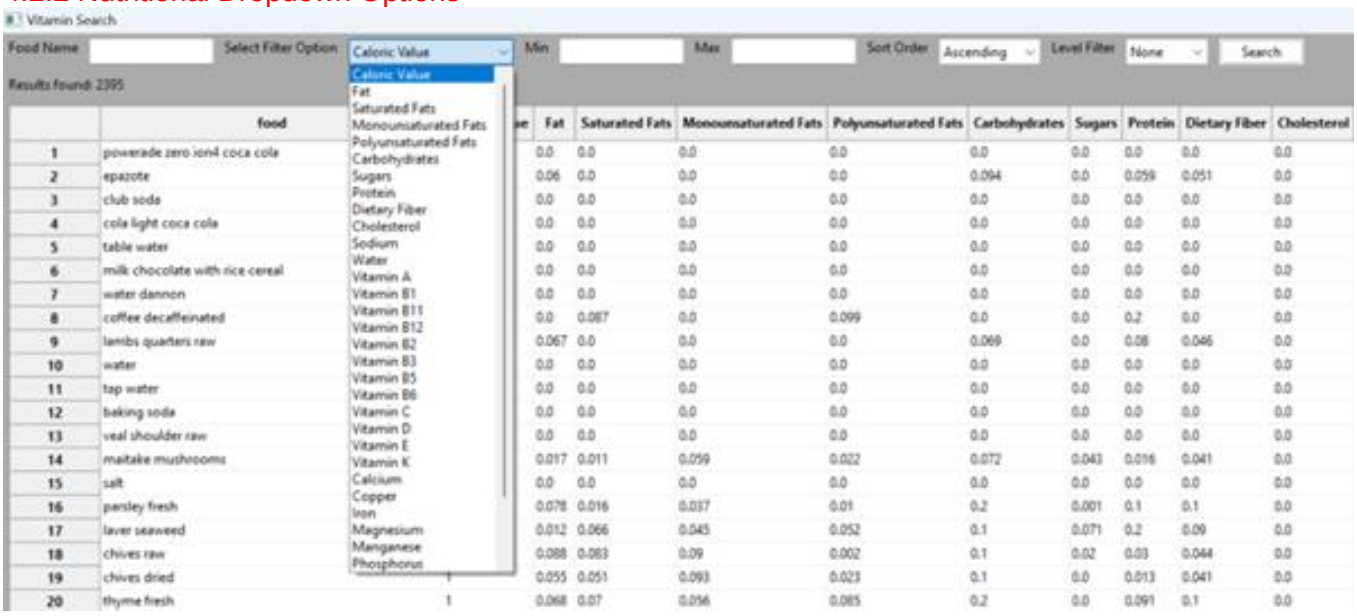
Data Visualization: Although the app does not yet incorporate advanced graphs or charts, the nutritional data is presented clearly in a table format. This table allows users to view and compare food items, filtering by nutritional values such as calories, proteins, and fats. The simplicity of this layout helps users quickly understand and analyze the nutritional content of different food items.

4.2 Visual Design

4.2.1 App Startup



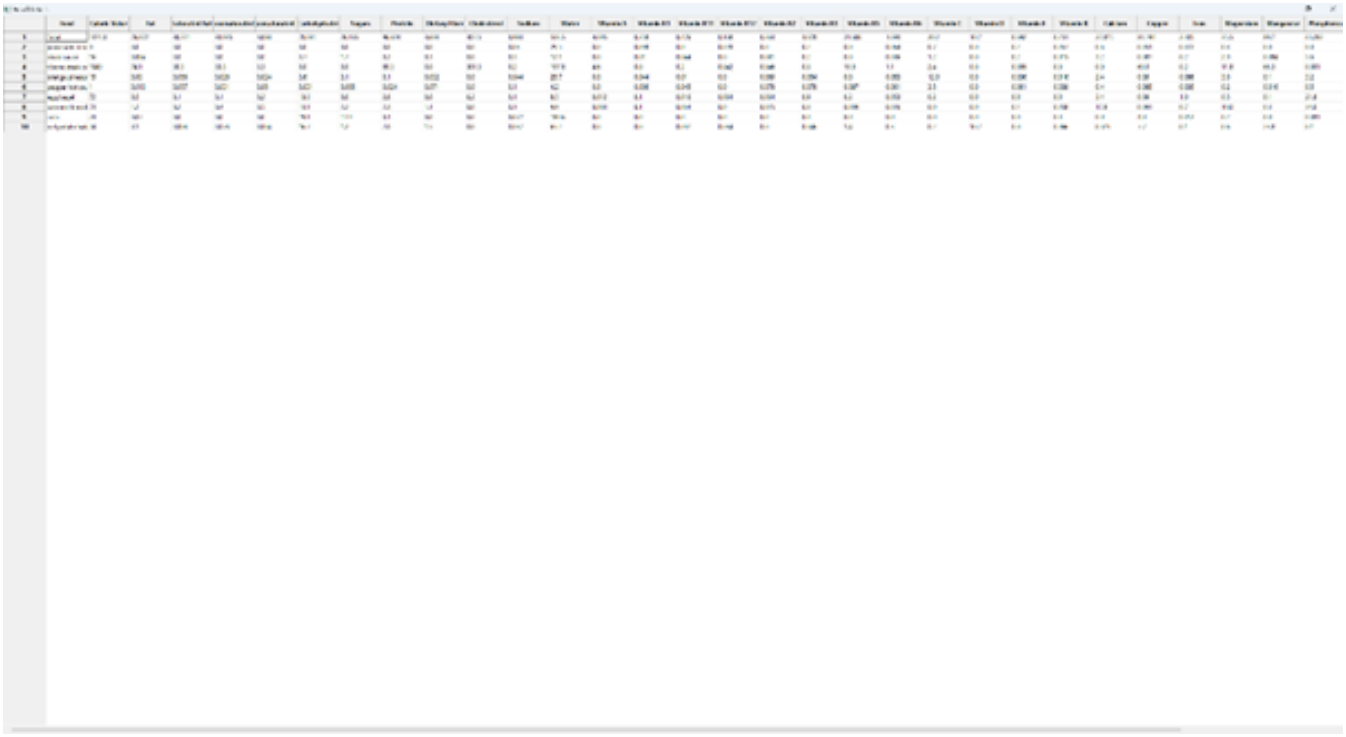
4.2.2 Nutritional Dropdown Options



	food	Fat	Saturated Fats	Monounsaturated Fats	Polyunsaturated Fats	Carbohydrates	Sugars	Protein	Dietary Fiber	Cholesterol
1	powerade zero ion4 coca cola	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	epazote	0.06	0.0	0.0	0.0	0.094	0.0	0.059	0.051	0.0
3	club soda	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	cola light coca cola	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	table water	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	milk chocolate with rice cereal	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	water dannon	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	coffee decaffeinated	0.0	0.087	0.0	0.099	0.0	0.0	0.2	0.0	0.0
9	lambs quarters raw	0.067	0.0	0.0	0.0	0.069	0.0	0.08	0.046	0.0
10	water	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11	tap water	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
12	baking soda	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
13	veal shoulder raw	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
14	maitake mushrooms	0.017	0.011	0.059	0.022	0.072	0.043	0.016	0.041	0.0
15	salt	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
16	parsley fresh	0.078	0.016	0.037	0.01	0.2	0.001	0.1	0.1	0.0
17	laver seaweed	0.012	0.066	0.045	0.052	0.1	0.071	0.2	0.09	0.0
18	chives raw	0.088	0.083	0.09	0.002	0.1	0.02	0.03	0.044	0.0
19	chives dried	0.055	0.051	0.093	0.023	0.1	0.0	0.013	0.041	0.0
20	thyme fresh	0.068	0.07	0.056	0.085	0.2	0.0	0.091	0.1	0.0

4.2.3 Filtered Search Results

4.2.4 Meal Planner



Note: Meal Planner is opened in a separate window, so both the Meal Planner and Database are visible at the same time.