ASHESI UNIVERSITY

**OOP Final Project**

**CS 213: Object-Oriented Programming**

**SEM1 - Fall (Sept-Dec) 2025**

**Group 5**

**Group Members:** Miriam Wepiya Gale

Nana Banyin Kwofie

Sebenemaryam Ashebir Asnake

Haris Tiyumtaba Issah

Meh Ayite Elinam Ajavon

**Lecturer**: David Ebo Adjepon-Yamoah

December 9, 2025

## Topic: Detecting Environmental Impact of Food Preferences Using Image Detection

**Introduction**:

Agriculture is the number one sector on which Africa's economy relies. It consists of 50% of Africa's workforce and contributes to the GDP (*Topic: Agriculture in Africa*, 2025). However, it has many environmental implications that are ignored, but it has a major effect on the degradation of forests, water source limitations, and greenhouse emissions. Research by Ritchie et al. (2022) found that food production accounts for over 26% of the global greenhouse gas emissions, because of carbon dioxide and nitrogen gas emissions. Around 70% of global freshwater is used for agriculture, and 78% of it is polluted by eutrophication, the fertilizers and pesticides that are used for farming. Also, half of the world's habitable land is used for agriculture, and hat increases greenhouse gas emissions.

Unless organizations that work for environmental protection, individuals care less about how the food they consume contributes to environmental degradation. But understanding the amount of carbon emissions, water usage, and land usage of the food one consumes daily can help each individual make an intentional choice that is sustainable for the environment. Therefore, our project aims to provide a tool that is easy to implement and allows users to check the environmental impact of their meals and arrange them accordingly.

**Description of our Project:**

Our project topic focuses solely on estimating the environmental impact of meals, especially West African meals, based on the food items the user builds to form a meal. This system is a web-based architecture that allows a user to take a picture of their meal, and the system we built identifies the food items and proportions using a recognition module. After

identifying the food items, the system calculates the food items' impact using environmental metrics such as carbon footprint per kilogram, water usage, and land usage. Then it combines the individual food items' contributions to calculate the total impact of the meal. The user can also compare the impact of different food choices and make a decision on which food to consume.

**System Overview:**

1. High-Level Architecture

Our system is made of several components, in addition to the main Java logic, that interact to give the final product. The components used are:

- **GUI (Frontend):** this is the part that the users see on the website, a list of food items, buttons, and so on. It is very important as it provides a user-friendly interface so that users do not have to interact with raw code, which they might not understand.
- **Backend API:** This is what connects the frontend(GUI) with the backend (Java logic). It receives the image that's uploaded from the frontend and sends it for the appropriate processing, such as calculations.
- **Image Recognition Module (Gemini Generative API):** This external service was used to take the image and identify food items in the image.
- **Database:** This is where the data about the food items is stored. It has information such as water usage, land usage, and carbon footprint of several food items. It is stored in a csv file in our application. This component is important because it makes our data organized.

System Functionality and Flow:

The flow of our system is:

- Accept an image of the meal the user built
- Determine food items (ingredients) that build the meal
- Calculate the portion (mass) of the each food items in the meal
- Calculate the carbon footprint, water usage, and land usage of the meal, which is built from several food items
- Display the results

2. Technology Framework

In our system, we have applied Java, HTML, CSS, JavaScript, and GitHub. We have used Java to do the backend logic and the environmental impact calculations. The classes, interfaces, and methods were created using Java. For the GUI, we implemented HTML, JavaScript, and CSS to make the application user-friendly. In addition, after each class has been created, we used GitHub for collaboration and code management.

3. Class & Design Description

Our system has classes:

- **EnvironmentalImpactCalculator:** This is our central class, as the name indicates, which calculates the environmental impact of a food. It has attributes such as foodItemDataset and imageAnalysis. And some methods like the constructor EnvironmentalImpactCalculator(), getFoodItem(), and so on.
- **FoodItem**: this class represents ingredients that make a meal. It has attributes like carbonFootprintPerKg and LandUsePerKg, and methods like getCarbonFootprintPerKg() and getNitrogenFootprint.
- **Meal**: this class is made up of the FoodItem Class. It contains multiple food items that the user built and computes the total environmental impact. It has fields like name and

an array of food items. Some of the methods are getFoodItems(), calculateTotalFootprints(), and more.

- **FoodPortion:** By taking the image of the meal, it calculates the mass of the food item that builds the meal. Then it gives the portion in kg. portionKg and foodItem, are attributes and getFoodItem(), toString(), and getPortionKg() are the methods.
- **ImageAnalysis:** This class reads the image of the meal inserted by the user and analyzes the image to give the food items of the meal. It has attributes like SYSTEM_PROMPT and methods like ImageAnalysis() and analyseImage().
- **WebServer:** this class is implemented to send the information to the website so that the users can easily calculate the environmental impact.

Our project Interfaces are:

- **InterfaceFoodItem**: this interface has methods that will later be implemented in the FoodItem class, such as getWaterUsagePerKg() and getFoodCategory(). And we call these types of methods abstract methods. When the class FoodItem implements this interface, environmental-based calculations for the specific food item will be provided.
- **InterfaceMeal:** it contains abstract methods that are implemented by the Meal class, these are: calculateTotalNitrogenFootprints(), calculateTotalWaterUsage(), and calculateLandUsage(). These methods are used to calculate the environmental impact of a meal that is made of several food items.
- **FileOperations:** this interface was used to load environmental data from a file. It has the method loadFromFile(), which takes the information from the file.

**Design Decisions & Justifications**

Our project demonstrates basic Object-Oriented Programming concepts such as Encapsulation, Aggregation, File I/O, and Error handling, which makes the web app that we developed feasible.

- **Encapsulation:** attributes of classes such as FoodItem, meal, and other classes (like carbonFootprintPerKg and waterUsage) are protected through private fields and public accessors and mutators. This protects the date from easy manipulation and modification that are not acceptable, so we controlled them through encapsulation.
- **Aggregation**: The Meal class aggregates multiple FoodItem objects, demonstrating a "has-a" relationship. A meal has several food items, but each food item can stand by itself. So, this allows us to calculate their environmental impact one by one and then sum up their result for the meal impact.
- **Interface**: we've created interfaces such as InterfaceMeal and InterfaceFoodItem that define the behavior of the classes. Interface helps with flexible calculations and the reusability of the Java design.
- **Error handling:** to ensure our system functions smoothly without any unexpected errors, we used error handling. For instance, for reading data from a file, we've used IOException handling so that no error occurs when reading the file.
- **File I/O:** it was used to load environmental impact information about the food items, such as water usage, carbon footprint, and nitrogen footprint, from a stored data file.
- **API:** After being introduced to API during class, we were able to learn more about it by ourselves. We've utilized API as a communication bridge between backend and frontend. So, after the user uploads the image and it is sent to the API, which will API then send it to the Gemini Generative API. The Gemini Genenerate API will then recognize the food items and send the results back through the API.

References

Ritchie, H., Rosado, P., & Roser, M. (2022, December 2). *Environmental impacts of food production*. Our World in Data. https://ourworldindata.org/environmental-impacts-of-food

*Topic: Agriculture in Africa*. (2025, May 28). Statista. https://www.statista.com/topics/12901/agriculture-in-africa/