

# ***AI-POWERED FOOD TRACKER WEB APPLICATION***

**BY- SREYA SHYJASH**

**BTECH CSE CYBER SECURITY**

**2<sup>ND</sup> YEAR**

# INDEX

| SI.NO | CONTENT              | PAGE NO. |
|-------|----------------------|----------|
| 1.    | AIM                  |          |
| 2.    | INTRODUCTION         |          |
| 3.    | METHODOLOGY          |          |
| 4.    | CODE                 |          |
| 5.    | RESULTS AND TESTING: |          |
| 6.    | CONCLUSION           |          |

# AIM

The primary aim of this project is to design and develop a dynamic web application, "Food Tracker," that simplifies nutritional logging. The application enables users to identify food items by uploading an image and provides detailed nutritional information for the identified food. A secondary objective is to provide alternative input methods, such as manual text search and quick suggestions, to ensure a comprehensive and user-friendly experience.

# INTRODUCTION

In an era of growing health consciousness, manual tracking of caloric and nutritional intake remains a tedious and often inaccurate process. Traditional methods rely on users searching for food items in vast databases and manually inputting quantities. This project addresses these challenges by leveraging modern artificial intelligence and API technologies to create an intuitive and automated food logging system.

The "Food Tracker" is a full-stack web application that serves as a proof-of-concept for a smarter nutrition diary. It provides users with two primary ways to log their meals: a state-of-the-art image recognition feature and a robust manual search function. By integrating with powerful third-party APIs for food recognition (Clarifai) and nutritional data (Nutritionix), the application offers a seamless user journey from capturing an image of a meal to viewing its detailed macronutrient breakdown. The project demonstrates key principles of modern web development, including frontend-backend separation, API consumption, and solving real-world browser security challenges like CORS.

# METHODOLOGY

The application was developed using a full-stack architecture, separating the user-facing client (frontend) from the data-processing and API-communicating server (backend).

## 1. Frontend Development (Client-Side)

The user interface was built using standard web technologies:

- **HTML:** Provided the core structure of the application, including the file uploader, image preview area, input fields, and data display sections.
- **CSS:** Used for styling the application, implementing a modern, dark-mode theme for visual appeal and user comfort. It also ensured the layout was responsive and user-friendly.
- **JavaScript:** Drove all client-side interactivity. Its key responsibilities included:
  - Handling user events like button clicks and file uploads.
  - Displaying the selected image in a preview area.
  - Sending requests to the backend server using the fetch API.
  - Dynamically rendering the results (food name, confidence score, and nutritional information) on the page.

## 2. Backend Development (Proxy Server)

A lightweight backend server was created using **Node.js** and the **Express.js** framework. This was a critical architectural decision made to solve the browser's Cross-Origin Resource Sharing (CORS) security policy, which prevented the frontend from directly calling the Clarifai API. The backend's functions are:

- To act as a secure intermediary or "proxy."
- To expose a single API endpoint (/analyze) for the frontend to call.
- To receive the image data from the frontend, securely embed the private API keys, and forward the request to the Clarifai API.
- To receive the response from Clarifai and pass it back to the frontend, effectively hiding the API keys from the end-user.

## 3. API Integration

The core functionality of the application is powered by two external APIs:

- **Clarifai API:** Utilized for its powerful food-item-recognition model. The backend sends a base64-encoded image to this API, which returns a list of potential food

concepts along with a confidence score for each. The application uses the top-scoring concept as the identified food item.

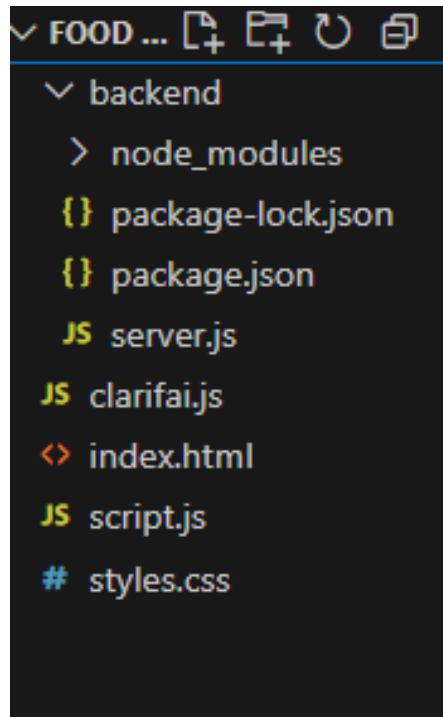
- **Nutritionix API:** Used to fetch detailed nutritional data. Once a food item is identified (either by Clarifai or manual user input), its name is sent to the Nutritionix API. The API returns precise macronutrient information, including calories, protein, fat, and carbohydrates, for a standard serving size.

#### **4. Development and Deployment Workflow**

- The frontend was tested in a live-reloading environment using the **VS Code Live Server** extension.
- The backend server was run locally using **Node.js**.
- The browser's Developer Tools were used extensively for debugging network requests, inspecting the console for errors, and ensuring seamless communication between the frontend and backend.

# CODE

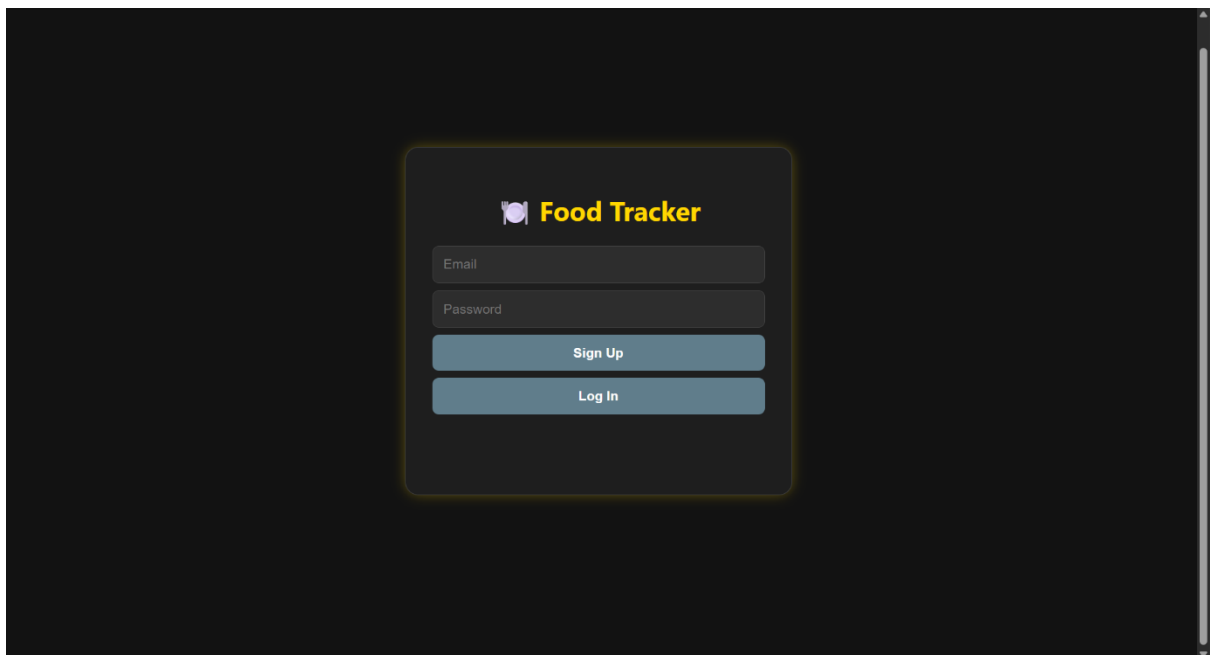
These are the files required. The codes are provided in the GitHub.



# RESULT AND TESTING


The project successfully culminated in a fully functional prototype of the "Food Tracker" application, achieving all the stated aims. The key outcomes are:

- **Successful Image-Based Food Recognition:** The application can accurately analyze an uploaded image and identify the primary food item. For example, an image of a pizza was correctly identified with a confidence score of 97.55%.
- **Accurate Nutritional Data Display:** Upon successful identification, the application fetches and displays the correct nutritional information for the food item, including calories, protein, fat, and carbohydrates.
- **Effective CORS Resolution:** The implementation of a Node.js proxy server completely resolved the "Failed to fetch" CORS errors, enabling reliable communication between the browser and the external APIs.
- **Fully Implemented Dual-Input System:** Users have the flexibility to either use the AI-powered image analysis or fall back on the manual search and quick suggestion buttons, making the tool versatile.
- **Intuitive and Responsive User Interface:** The final product features a clean, visually appealing, and easy-to-navigate interface that provides clear feedback to the user during the analysis and data-fetching processes.






# SIGN UP

 **Food Tracker**


Select gender

Submit Profile

 **Food Tracker**

Choose File

34f48f5c56c938642b80b0555e5ad...



Analyze

pizza detected with 97.55% confidence.

Quick suggestions:

Pizza

Burger

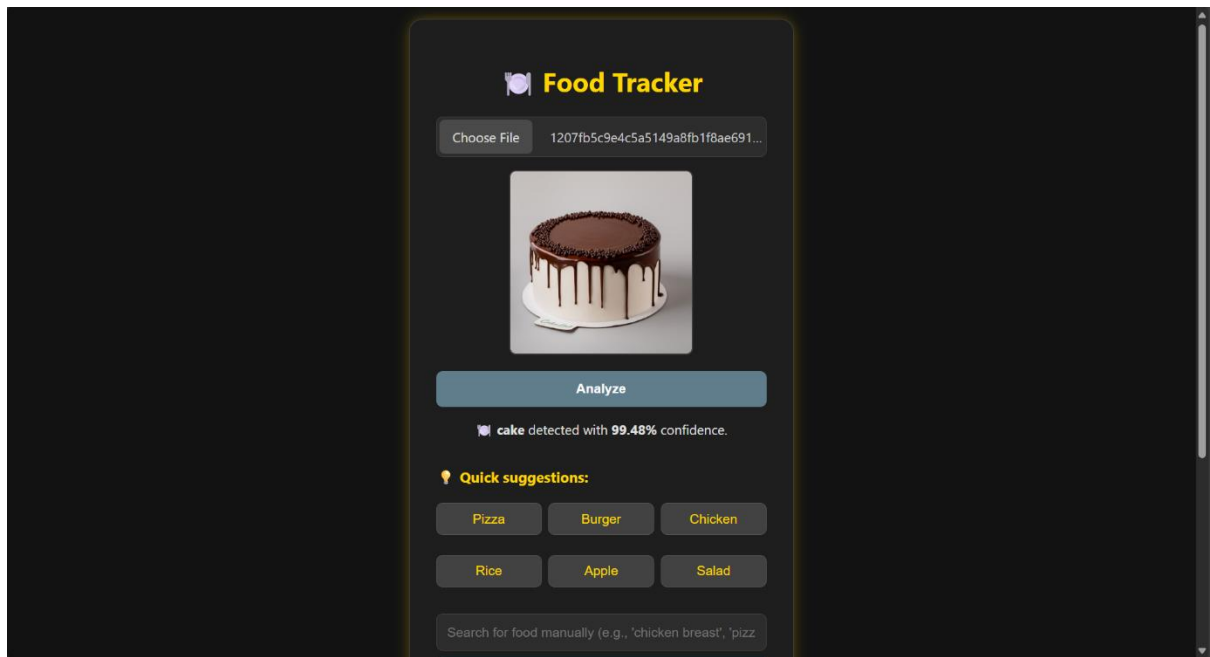
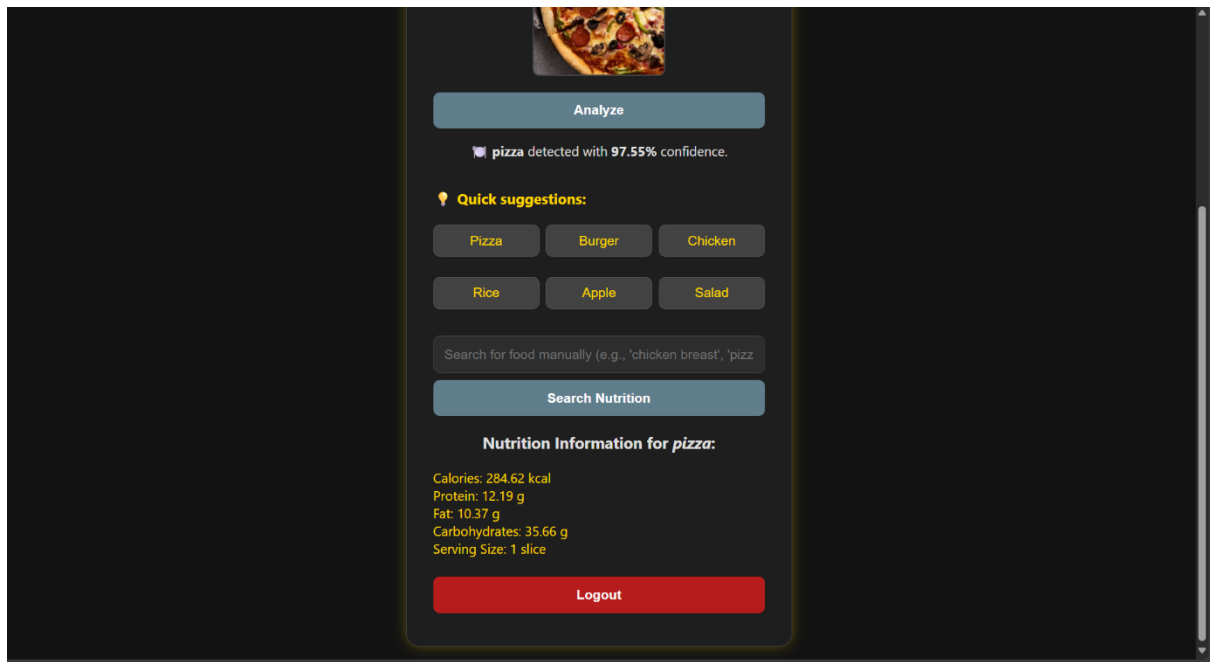
Chicken

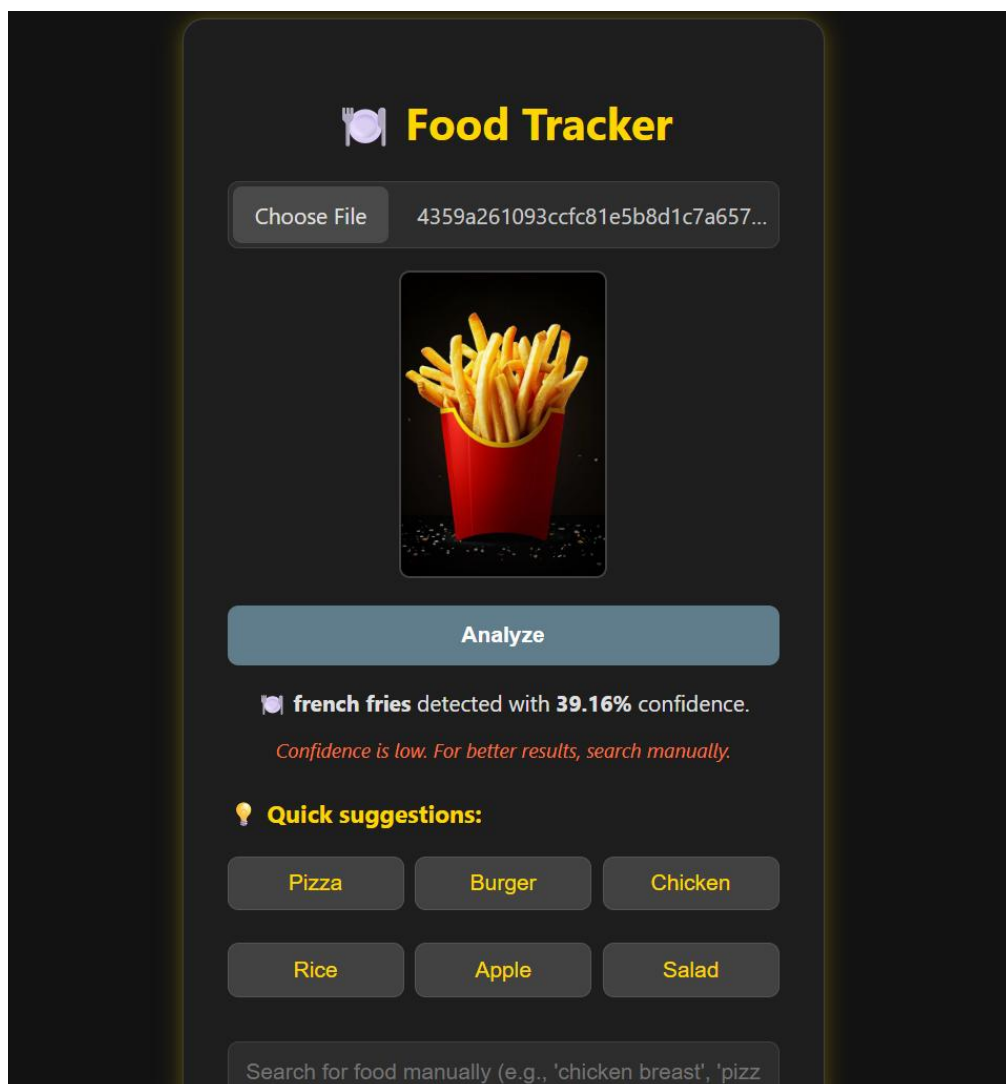
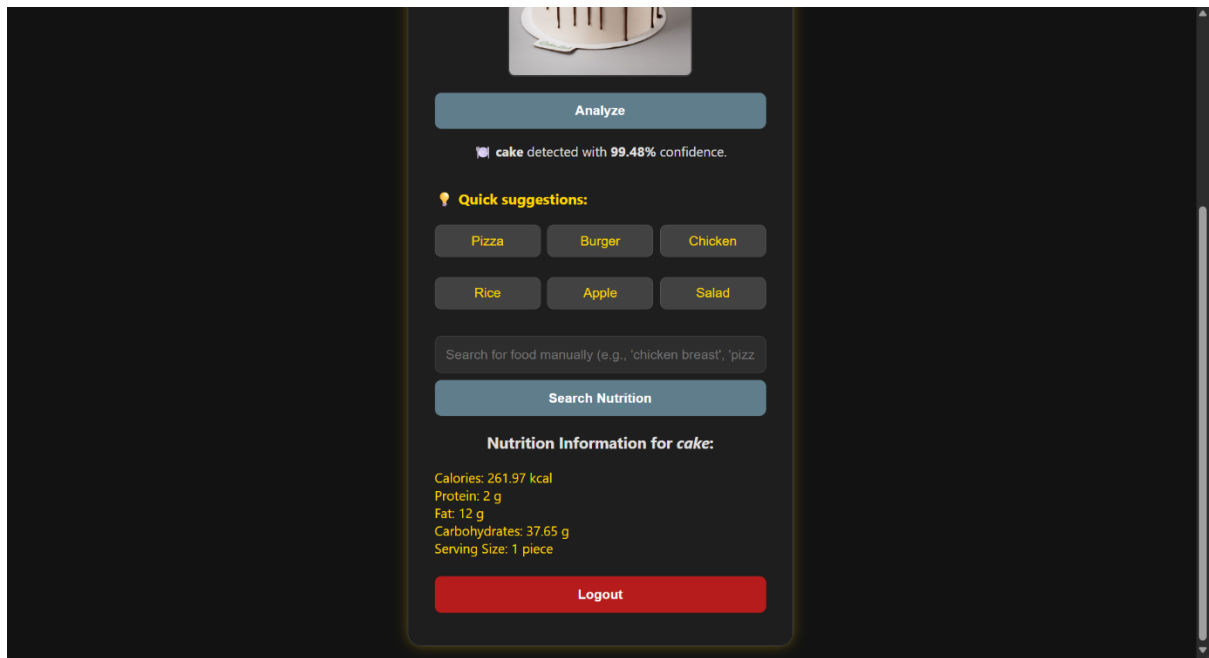
Rice

Apple

Salad

french fries





💡 **Quick suggestions:**

Pizza

Burger

Chicken

Rice

Apple

Salad

french fries

Search Nutrition

**Nutrition Information for *french fries*:**

Calories: 365.04 kcal

Protein: 4.01 g

Fat: 17.23 g

Carbohydrates: 48.48 g

Serving Size: 1 serving medium

Logout

# CONCLUSION

This project successfully demonstrates the power of combining modern web technologies and artificial intelligence to solve a common, everyday problem. By creating a seamless, full-stack application, the "Food Tracker" provides a user-friendly and efficient alternative to manual nutrition logging.

The development process provided invaluable practical experience in integrating third-party APIs, understanding and solving complex browser security policies like CORS, and architecting a client-server application. The final result is a robust proof-of-concept that is not only functional but also serves as a strong foundation for future enhancements. Potential future work could include adding a database (like Firebase Firestore) to save user food logs, creating daily and weekly nutritional summaries, and expanding the feature set to include barcode scanning.