IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, July 2026
Impact Factor: 8.2

AI-Based Food Calorie Detection System Using YOLOv8 and Deep Learning for Personalized Nutrition Tracking

Vimal R Rao[1], Sagar R[2], Dr. Manjunath B[3]
PG Student, Department of Master of Computer Applications[1,2]
Faculty, Department of Master of Computer Applications[3]
Maharaja Institute of Technology, Mysore, Visvesvaraya Technological University, Belagavi, Karnataka, India

Abstract: *Maintaining an accurate record of daily food intake is central to healthy dietary behaviour, yet conventional calorie-tracking applications rely on manual food search or barcode scanning, a process most users abandon within days because of its repetitive nature. This paper presents an AI-Based Food Calorie Detection System that replaces manual logging with a single photograph of a meal. A YOLOv8 object detection model, fine-tuned on a food-specific image dataset, identifies and localises multiple food items within one frame, returning a class label, confidence score and bounding box for every detected object instead of a single whole-image label. The bounding-box area of each detection is used to approximate its relative portion size, so the retrieved calorie and macronutrient values scale with the visible quantity of food rather than a fixed serving size. Detected items are cross-referenced against a nutrition reference dataset, and the results are combined with the user's personal body profile to generate individualised daily calorie and macronutrient targets. The system is implemented as a lightweight, self-contained web application using a FastAPI backend, an HTML/CSS/JavaScript frontend, and an embedded SQLite database, removing the need for external cloud services during inference. Functional testing across twenty-seven test cases spanning authentication, detection, calorie estimation, and reporting confirms reliable end-to-end operation of the proposed system*

Keywords: Food calorie detection, YOLOv8, object detection, deep learning, FastAPI, nutrition tracking, computer vision, portion estimation, SQLite

I. INTRODUCTION

Keeping an accurate record of daily food intake is one of the most effective ways to manage weight, chronic conditions such as diabetes, and general dietary health, yet most people who attempt it give up within the first few weeks. Manual calorie logging requires the user to weigh each item, search for its nutritional information, and enter it into an application for every single meal, a routine that quickly becomes tedious and is eventually abandoned. Computer vision offers a more natural alternative: instead of asking the user to describe a meal in words, the system can simply look at a photograph of it.

Most calorie-tracking applications available today depend on manual database search or barcode scanning, an approach that works reasonably well for packaged food but struggles with home-cooked or restaurant meals that combine several ingredients on a single plate. Applications that do accept a food photograph typically use single-label image classifiers, which return one label for the entire image; a plate containing rice, dal and a vegetable side dish is therefore reduced to a single, often inaccurate, class [2], [3]. Existing systems also tend to assign a fixed calorie value to each recognised class regardless of the quantity actually present in the image, and they rarely relate the estimated calorie count to the user's own body profile or daily nutrition budget.

DOI: 10.48175/IJARSCT-37891

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, July 2026
Impact Factor: 8.2

Object detection architectures address the multi-item limitation directly by locating every object instance in an image rather than assigning one label to the whole frame. Since Redmon et al. introduced the YOLO (You Only Look Once) family of detectors, which predict bounding boxes and class probabilities in a single network pass instead of the two-stage region-proposal pipeline used by earlier detectors [1], YOLO-based models have become a popular choice for interactive applications that must return a result within seconds of an image being submitted. This work fine-tunes the YOLOv8 detector on a food-specific dataset and combines it with a calorie and macronutrient mapping engine, a personalised nutrition profile, and a lightweight full-stack web application to produce an end-to-end system that requires nothing more from the user than a photograph of their meal.

Related work in this space follows a few recurring patterns. Systems that add an explicit segmentation or volumetric-estimation stage after detection, such as the Mask R-CNN and YOLOv5 hybrid reported by Agarwal et al. [4] or the YOLOv4-plus-image-processing pipeline of Patil et al. [3], achieve higher calorie-estimation accuracy but at the cost of a heavier, multi-stage pipeline that is not well suited to a fast, browser-based application. Systems without this extra stage, including the early automatic calorie estimator of Deshmukh et al. [2], typically fall back on a single fixed calorie value per class and do not attempt multi-item detection. A second recurring gap is regional representation: work such as Dhar et al.'s calorie estimator for Bangladeshi street food [7] shows that most public food-recognition datasets and pretrained models are skewed toward Western or East Asian cuisine, leaving cuisines such as Indian food under-represented and requiring local fine-tuning. A third gap is personalisation: existing systems generally report the calorie content of a detected meal in isolation, leaving it to the user to judge whether that value fits within their daily allowance. The system proposed in this paper is designed to close all three gaps by fine-tuning YOLOv8 on food-specific classes, using the bounding-box area as a lightweight proxy for portion size without an additional segmentation stage, and linking every detection to the user's individual daily calorie and macronutrient targets.

The remainder of this paper is organised as follows. Section II describes the methodology, covering the system architecture, data acquisition, preprocessing, detection pipeline, calorie mapping, database design, security and deployment. Section III presents the results of functional testing and discusses system behaviour. Section IV concludes the paper, and Section V outlines directions for future work.

Methodology

The methodology followed in developing the AI-Based Food Calorie Detection System focuses on producing an intelligent, self-contained pipeline capable of turning a single food photograph into a personalised nutritional summary. The approach integrates browser-based image acquisition, YOLOv8-based deep-learning detection, a calorie and macronutrient mapping engine, and a lightweight persistence layer, all coordinated by a FastAPI backend.

System Design Rationale : The architecture is organised around a single guiding principle: the user should never have to identify or search for a food item manually. Every design decision, from choosing an object detector that runs in a single pass to storing data in an embedded database rather than a remote server, was made to keep the round-trip time from photograph to result as short as possible while keeping the deployment footprint small enough to run on an ordinary computer without additional infrastructure.

System Architecture : The system is organised into four independent layers: a presentation layer, an application layer, a detection and AI layer, and a data layer. The presentation layer, built with HTML, CSS and JavaScript, handles image upload and camera capture, renders the nutrition dashboard, and displays meal history, but performs no computation itself. The application layer is a FastAPI backend that validates incoming requests, coordinates calls to the detection layer, and returns JSON responses to the frontend. The detection and AI layer hosts the fine-tuned YOLOv8 model together with the calorie mapping logic, while the data layer persists user accounts, meals, detections, nutrition reference data and daily targets in an embedded SQLite database. Keeping these layers independent allows the detection model to be retrained or replaced without any change to the frontend or database schema.

Input Data Acquisition Layer : The user supplies a meal image either by uploading an existing photo in JPEG or PNG format or by capturing one directly through the browser's camera API. The frontend performs a basic client-side check

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, July 2026
Impact Factor: 8.2

on file type and size before the image is sent to the backend, and the server independently re-validates both properties so that an unsupported or oversized file is rejected before it reaches the detection pipeline.

Image Preprocessing : On arrival at the backend, the image is decoded and pre-processed using OpenCV, which resizes and normalises the image to the input dimensions expected by the YOLOv8 model. This step removes format inconsistencies between different cameras and devices and ensures that every image presented to the model has a uniform scale before inference.

YOLOv8 Model Selection and Fine-Tuning : YOLOv8 was selected over two-stage detectors such as Faster R-CNN because it predicts bounding boxes and class probabilities for an entire image in a single forward pass, an approach first established for the YOLO family by Redmon et al. [1], which keeps inference latency low enough for an interactive web application. Architecturally, the model follows the standard YOLO backbone-neck-head design: a backbone extracts feature maps at multiple scales, a neck fuses these features so that both small and large food items can be detected in the same frame, and a detection head predicts class scores and bounding-box coordinates for every candidate region. The base YOLOv8 model was fine-tuned on a food-image dataset so that its predicted classes correspond to food items rather than the generic object categories of the default pretrained weights.

Detection and Non-Maximum Suppression Pipeline : Inference produces a set of candidate detections, each carrying a class label, a confidence score and bounding-box coordinates. Detections whose confidence falls below a configured threshold are discarded to suppress false positives, and non-maximum suppression is then applied to remove duplicate, overlapping boxes that correspond to the same physical food item. If no detection survives this filtering, the system reports that no food item was found instead of returning an empty or misleading result.

Calorie and Nutrition Mapping : Each surviving detection is looked up in a nutrition reference dataset that stores the calorie, protein, carbohydrate, fat and fibre content per standard serving of every food class. Rather than applying this reference value unmodified, the system scales it by the relative area the bounding box occupies within the image, so a large detected portion contributes proportionally more to the meal total than a small one. This avoids the complexity of a full volumetric or depth-based estimation step while still adapting the estimate to the amount of food actually visible. The scaled values for every detected item are summed to produce the calorie and macronutrient totals for the meal, and any detected class absent from the reference dataset is flagged as unrecognised rather than silently ignored.

Database Design : Persistent data is stored in an embedded SQLite database structured around five entities: User, Meal, Detection, Nutrition Item and Daily Target. A User has a one-to-many relationship with Meal records and a one-to-one relationship with a Daily Target record derived from the user's body measurements and activity level. Each Meal is linked to one or more Detection records, reflecting the possibility of several food items being logged from a single image, and every Detection references exactly one Nutrition Item that supplies its reference calorie and macronutrient values. Using a file-based database removes the need for a separate database server and keeps the application easy to deploy on a single machine.

Authentication and Security : User passwords are never stored in plain text; each password is salted and hashed before being written to the database, and login requests are verified by comparing hashes rather than raw credentials. A successful login returns a session token that must accompany every subsequent request to a protected route, so that dashboard, profile and meal-history data cannot be accessed without a valid, matching token. Uploaded files are also checked for type and size before being handed to the detection pipeline to prevent the ingestion of unexpected or unsafe content.

Deployment Architecture : The complete application, including the frontend, the FastAPI backend, the YOLOv8 model and the SQLite database file, is deployed as a single unit on one machine and served through Uvicorn, so that no separate content-delivery network or database server is required. The YOLOv8 model is loaded into memory once when the server starts and is reused for every subsequent request, which avoids the overhead of reloading the model for each detection and keeps response times low.

Testing Methodology : The system was tested continuously throughout development rather than only after completion. Individual functions such as authentication, image preprocessing and calorie calculation were unit tested; the connections

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, July 2026
Impact Factor: 8.2

between the frontend, backend, detection model and database were integration tested; and the complete application was exercised end to end, from login through image submission to an updated dashboard and meal-history entry. Twenty-seven test cases were defined across authentication, image upload, camera capture, detection, calorie calculation, the database layer, the dashboard, profile settings, nutrition reports, security, performance, logout and an end-to-end flow, summarised in Table I.

Results and Discussion

The proposed system was evaluated primarily through functional and integration testing rather than a held-out accuracy benchmark. The objective at this stage was to verify that the complete pipeline, from image capture through detection to a personalised dashboard update, behaves correctly and reliably across normal and edge-case inputs.

Table I. Summary of Test Case Categories and Outcomes

Module	Test Cases	Result
Authentication	4	All Passed
Dashboard	2	All Passed
Image Upload	3	All Passed
Camera Capture	1	Passed
Food Detection	4	All Passed
Calorie Calculation	3	All Passed
Database	2	All Passed
Profile Settings	2	All Passed
Nutrition Reports	1	Passed
Security	2	All Passed
Performance	1	Passed
Logout	1	Passed
End-to-End Flow	1	Passed
Total	27	27/27 Passed

As summarised in Table I, all twenty-seven test cases passed. Authentication correctly created new accounts, rejected incorrect credentials, and denied access to protected routes without a valid session token. Image upload and camera-capture tests confirmed that valid JPEG and PNG files were accepted and forwarded for detection, while unsupported formats and oversized files were rejected with a clear error message before reaching the detection model.

The detection module was verified against three categories of input: a single food item, multiple food items on one plate, and an image containing no food at all. In the first two cases the model returned correctly labelled bounding boxes for each item, including cases where several items of different sizes appeared together, confirming that the multi-item detection approach behaves as intended instead of collapsing the plate into a single label. In the third case the system reported that no food item was detected rather than returning a false or empty result, which is an important usability property since users are more likely to trust a system that admits uncertainty than one that silently fails.

Calorie-calculation tests confirmed that the nutrition values returned for a single detected item matched the reference dataset, that totals for multi-item meals equalled the sum of the individual scaled values, and that a detected class absent from the nutrition dataset was flagged as unrecognised rather than ignored. Database tests confirmed that meal records, including all associated detections, were stored and retrieved correctly and in chronological order, and that the dashboard updated immediately after a new meal was logged without requiring a page refresh.

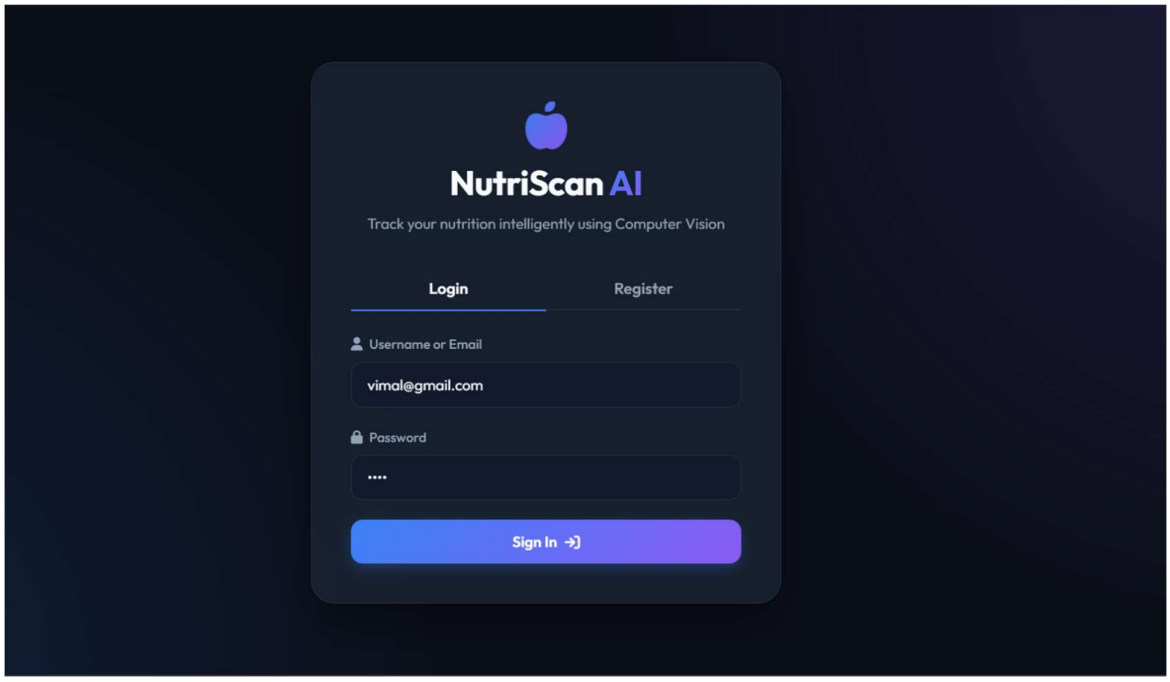
Security testing confirmed that a request attempting to access another user's meal data through a direct API call was rejected due to a session-token mismatch, and that requests to protected routes without a valid token were denied and redirected to the login page. Performance testing showed that detection results were returned within a few seconds of

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, July 2026
Impact Factor: 8.2

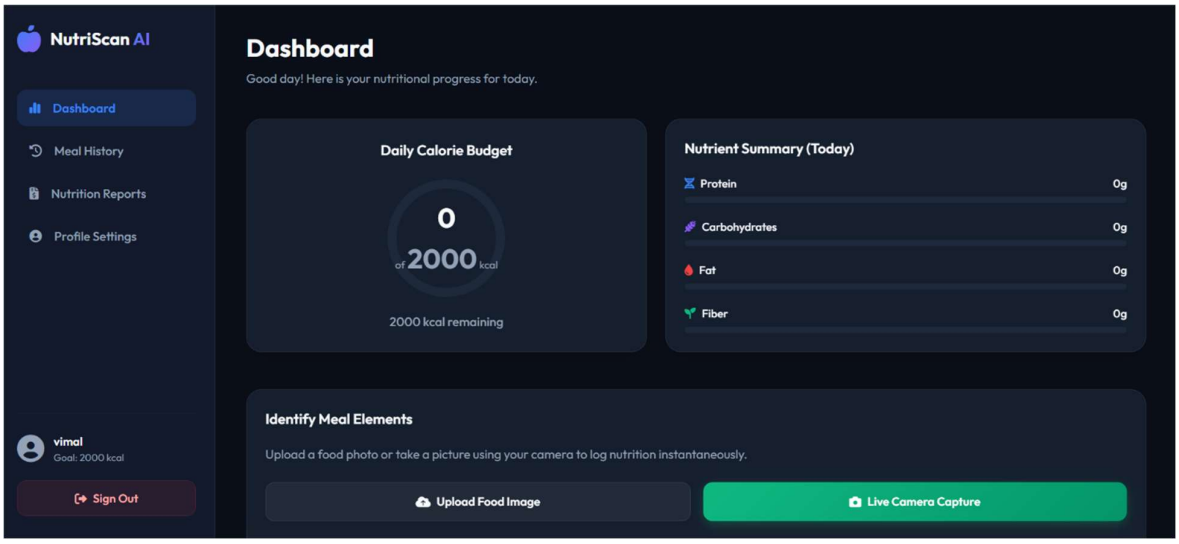
image submission, which is acceptable for an interactive application and is aided by the decision to keep the YOLOv8 model resident in memory rather than reloading it for every request.

The layered architecture, which separates presentation, application, detection and data concerns, allowed each component to be verified on its own while still working correctly as part of the integrated system. Multi-item detection and bounding-box based portion scaling directly address the single-label and fixed-serving-size limitations identified in related work [2], [3], while the personalised daily target links every logged meal to the user's own nutrition goals rather than reporting calorie values in isolation. The main limitation is that, as with any two-dimensional image-based estimate, portion size is approximated from bounding-box area rather than true volume. Occluded food, garnishes and liquid ingredients such as oil or sauce are therefore not captured precisely, a limitation shared with comparable systems in the literature [5] and revisited in Section V.

Snapshot



IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, July 2026
Impact Factor: 8.2



II. CONCLUSION

This paper presented an AI-Based Food Calorie Detection System that replaces manual food logging with a single photograph, using a YOLOv8 model fine-tuned on food-specific classes to detect and localise multiple food items within one image. Bounding-box area was used as a lightweight proxy for portion size, allowing calorie and macronutrient estimates to scale with the visible quantity of food without the overhead of a separate segmentation or volumetric-estimation stage. The system was implemented as a self-contained full-stack application using FastAPI, OpenCV, an embedded SQLite database and a browser based frontend, and functional testing across twenty-seven test cases confirmed reliable operation of authentication, detection, calorie estimation, dashboard reporting and security. By tying every detected meal to the user's personal daily calorie and macronutrient targets, the system moves beyond simply reporting a calorie count and gives the user a meaningful reference point for their own dietary goals. The clear separation between the presentation, application, detection and data layers keeps the system maintainable and leaves room for future extension without requiring the application to be rebuilt from scratch.

Future Enhancement

Several directions can extend the present work. Portion-size estimation can be made more precise by incorporating a depth-estimation technique, using either a stereo camera or a reference object of known size placed in the frame, rather than relying solely on bounding-box area. The detection model can be trained on a larger and more diverse set of regional cuisines and mixed dishes, ideally through an active-learning loop in which the system flags its most uncertain detections for manual annotation and retraining, addressing the regional-representation gap noted for Indian and other under-represented cuisines [7]. Support for short video clips, rather than single static images, would allow the system to observe a dish as it is being prepared or consumed, and a voice-input or barcode-scanning fallback would help the user log foods the model does not yet recognise. Finally, moving the deployment from a single local machine to a cloud-hosted, multi-user service would allow the same detection pipeline to serve an entire college hostel or workplace, potentially paired with a dietician-facing dashboard and integration with existing fitness-tracking applications.

DOI: 10.48175/IJARSCT-37891

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, July 2026
Impact Factor: 8.2

Acknowledgment

I would like to express my sincere gratitude to my project guide for their invaluable guidance and support throughout this research. I am also thankful to the faculty of the Department of Master of Computer Applications, Maharaja Institute of Technology Mysore, for providing the resources and environment necessary to carry out this work, and to my peers for their feedback during development and testing.

REFERENCES

1. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016.
2. P. B. Deshmukh et al., "Calorimeter: Food Calorie Estimation with Machine Learning," 2021.
3. S. Patil, S. Patil, V. Kale, and M. Bonde, "Food Item Calorie Estimation and Image Processing," 2021.
4. Agarwal et al., "Hybrid Deep Learning Algorithm-Based Food Recognition and Calorie Estimation," 2023.
5. T. Ege and K. Yanai, "Image-Based Food Calorie Estimation Using Knowledge on Food Categories, Ingredients and Cooking Directions," 2017.
6. G. Lakshmi Pravalika, G. Harsha Vardhan, and M. Ashok Kumar, "Fruit Recognition and Calorie Estimation Using YOLO," 2025.
7. Dhar, M. T. Hossain, and P. Barua, "Estimating Calories of Bangladeshi Street Foods Using Enhanced YOLOv8 and Regression Model," 2025.