AI Ingredients & Nutrition Analyzer

An AI-Based Framework for the Analysis of Consumable Products

Naif Almalik
naifalmalikk@gmail.com

Ahmad Algwaiz
Ahmadf605@hotmail.com

Mohammed Alabdulmuhsin
Mm.alabdulmuhsin@gmail.com

Ibrahim Alshayea
ibrahimalshaya5@gmail.com

Faisal Alwadie
faisal11bander@gmail.com

Mentor: Dr. Salah Abdeljabar
salah.abdeljabar@kaust.edu.sa

August 27, 2025

Abstract

Food product labels often provide limited ingredient details and require consumers to perform calculations to interpret nutritional values, making them difficult to understand. This project presents a **novel AI-based system** that automatically extracts and analyzes both ingredient and nutrition information, delivering deeper and more accessible insights. The system integrates Optical Character Recognition with a Large Language Model to process label text, interpret its meaning, and generate structured outputs. These results are then presented through a mobile application in a clear and user-friendly format. The system demonstrates reliable extraction and analysis of label information, highlighting how AI can enhance food transparency and empower consumers to make healthier decisions.

Contents

1 Introduction

1.1 Contributions

- **Naif Almalik (Team Leader, OCR Developer)**
  - Coordinated all meetings with the mentor and organized internal team sessions, including brainstorming and follow-up discussions to keep the team aligned.
  - Led the development and Implementation of the OCR module with collaboration from **Faisal Alwadie**, Including preprocessing and integration into the overall workflow.
  - Contributed significantly to report and poster preparation, with collaboration from **Ahmad Algwaiz**.
  - Authored the weekly progress reports.
  - Played a central role in team management, balancing coordination across the full team and within smaller sub-groups to maintain steady progress.
- **Ahmad Algwaiz (Idea Proposer, App Engineer & Developer)**
  - Proposed the project idea and constructed the initial framework pipeline, establishing the architecture for the system.
  - Designed the end to end UI/UX flow, Providing 4 high quality pages with complex widgets, dynamic layouts, and seamless animations.
  - Implemented the full stack, engineering both the front-end and back-end components using Flutter framework.
  - Managed team deliverables, aligning outputs to product requirements and ensuring high-quality, efficient results.
  - Contributed significantly to report and poster preparation, with collaboration from **Naif Almalik**.
- **Mohammed Alabdulmuhsin (LLM Developer)**
  - Worked in collaboration with **Ibrahim Alshayea** on the development and implementation of the Large Language Model (LLM) component.
  - Collaborated with **Ibrahim Alshayea** to design and build the overall LLM pipeline.
  - Tested the end-to-end system flow in collaboration with **Ibrahim Alshayea**.
  - Assisted in creating the final JSON output to be presented in the app, working with **Ahmad Algwaiz** and **Ibrahim Alshayea**.
  - Contributed to the integration of the Optical Character Recognition (OCR) module, the Large Language Model (LLM), and the mobile application.

- **Ibrahim Alshayea (LLM Developer)**
  - Worked in collaboration with **Mohammed Alabdulmuhsin** on the development and implementation of the Large Language Model (LLM) component.
  - Collaborated with **Mohammed Alabdulmuhsin** to design and build the overall LLM pipeline.
  - Tested the end-to-end system flow in collaboration with **Mohammed Alabdulmuhsin**.
  - Assisted in creating the final JSON output to be presented in the app, working with **Ahmad Algwaiz** and **Mohammed Alabdulmuhsin**.
  - Created the initial warnings dataset to support enrichment in the absence of reliable APIs.
- **Faisal Alwadie (OCR Developer)**
  - Assisted with the development and implementation of the OCR module, including preprocessing and integration tasks, in collaboration with **Naif Almalik**.
  - Contributed to collecting and analyzing statistics related to consumer food awareness.
  - Documented the main problems and challenges associated with reading and understanding ingredient labels.

1.2 Problem motivations and real-world impact

Many consumers, particularly those who are health-conscious, struggle to identify suitable products that meet their dietary needs. This challenge arises from the difficulty of interpreting ingredient lists and nutritional information on product labels, which are often complex, inconsistent, or require additional calculations. According to an IFIC study, **only 9%** of consumers fully understand food labels, and **6 out of 10** consumers want healthier products but struggle to find them due to difficulties in interpreting labels. As a result, consumers may overlook important details about additives, allergens, or nutritional values, leading to less informed decisions. By addressing this issue, the project aims to make it easier and faster for health-conscious consumers to understand product contents, improve transparency, and ultimately support healthier lifestyles.

1.3 Key Assumptions or Limitations of the problem

Our main assumption is that health-conscious consumers are highly selective and expect precise, reliable information when making food choices. In matters related to health, particularly allergies and warnings, accuracy is critical and incorrect assumptions cannot be made. Therefore, the system must carefully address these aspects to ensure trustworthiness. Another assumption is that food labels contain sufficient information to be analyzed in a meaningful way, and that standardized labeling practices are generally followed, allowing the system to interpret data consistently. It is also assumed that consumers are willing to rely on technology to assist them in understanding product contents and making healthier decisions.

A key limitation, however, is the limited availability of public datasets on ingredients, which poses challenges in expanding the depth and coverage of the system's knowledge. Furthermore, ingredient and nutrition labels can vary significantly across regions and manufacturers, which makes consistent interpretation more difficult. In addition, some labels may present vague or incomplete information, limiting the insights that can be extracted. Finally, the perception of what constitutes a "healthy product" may differ among consumers, which places a natural limit on how universally applicable the system's recommendations can be.

1.4 Project objectives and goals

The primary objective of this project is to design and implement a system that enables consumers to easily access and understand ingredient and nutrition information from product labels. The project aims to transform raw label data into meaningful insights that support informed and healthier decision-making. To achieve this objective, the goals include:

- Developing a reliable text extraction process capable of handling diverse and complex label formats.
- Ensuring accurate analysis and interpretation of extracted information, with a focus on clarity and usability.
- Delivering outputs in a structured and user-friendly format that can be integrated into a consumer-facing application.
- Maintaining a high level of accuracy and consistency to address health-critical aspects such as allergens, additives, and warnings.
- Integrating the OCR, LLM, and mobile application components into a complete end-to-end pipeline that functions smoothly under real-world conditions.
- Supporting consumer usability by presenting results in a clear, concise manner that empowers users to make confident, informed food choices.

1.5 Summary of the Proposed system

The proposed system is designed to automatically extract text from nutrition and ingredient labels and analyze it to provide clear and actionable information for consumers. The process begins with an image of a food product label, which is processed by an Optical Character Recognition (OCR) tool to accurately extract the relevant text. The extracted text is then passed to a Large Language Model (LLM), which interprets the ingredients and nutritional information and organizes it into a unified JSON format. Finally, this structured data is delivered to the mobile application, where it is presented in a concise and user-friendly manner, enabling consumers to quickly understand product contents and make informed decisions.

2 System Overview

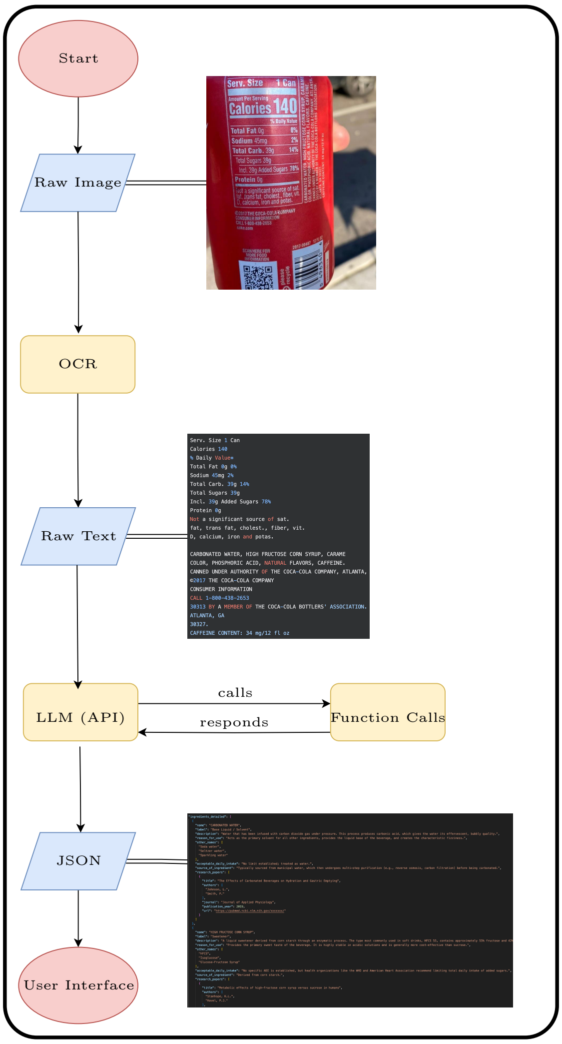

Figure 1: Initial proposed pipeline for the System.

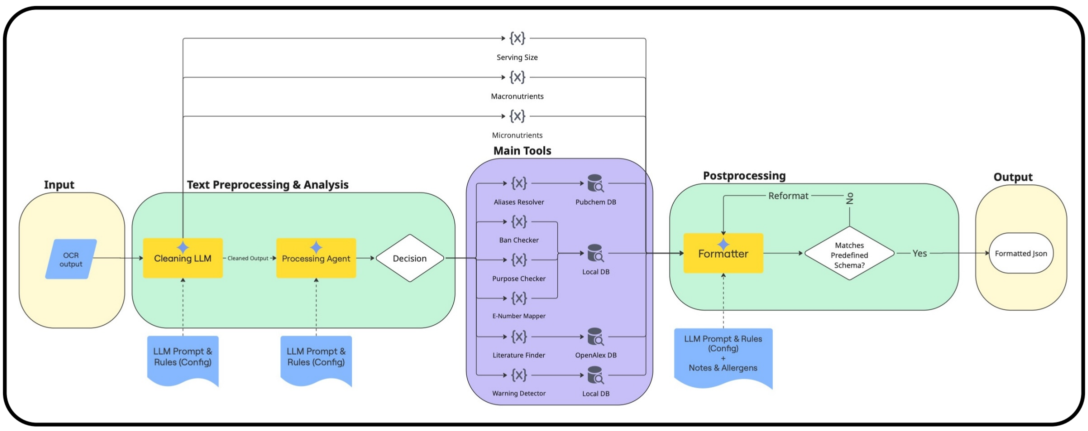

Figure 2: Detailed pipeline of the core LLM.

3 Methodology

- Agentic Workflow Design
  - Agent controls tool calls based on extracted ingredients to get more reliable and consistent answers.
- Parallelization & Efficiency
  - Tool calls for multiple ingredients run in parallel using a thread pool executor.
  - In-memory caching is applied per ingredient-tool pair to reduce repeated queries.
- Formatter Model
  - A secondary LLM formats final results into a consistent schema with fallback/retry logic.
  - Ensures outputs adhere to predefined structure for downstream applications.
- FastAPI Deployment
  - Entire pipeline wrapped in a **FastAPI** service with process-image endpoint.
  - Supports image upload, OCR, agent reasoning, and final structured output in JSON.
  - Includes health check endpoints for monitoring.

3.1 Resources and Tools

- Optical Character Recognition
  - Local OCRs: PaddleOCR, EasyOCR, Tesseract, TrOCR
  - API OCRs: Google Vision OCR, AWS Textract, Azure AI OCR
- Preprocessing
  - Grayscale conversion, Binarization, Noise removal, Contrast adjustment, Deskewing, Perspective correction, Resizing, Cropping/ROI extraction, Edge detection, Text segmentation.
- Frameworks and APIs
  - FastAPI: for building and deploying the REST API endpoints.
  - Uvicorn: ASGI server to run the FastAPI app.
  - Requests: for making external API/tool calls.
- AI/LLM Infrastructure
  - Google Generative AI (Gemini): used for both cleaning OCR text and formatting final outputs (Model 1.5 Flash)
  - Azure AI Vision: used as the OCR backend for extracting text from images.
- Data Handling and Processing
  - Pandas: for tabular data manipulation (warnings database, E-numbers, etc.).
  - Regex: for cleaning raw OCR outputs and enforcing valid JSON structure.

- Parallelization & Caching
  - ThreadPoolExecutor (concurrent.futures): for running multiple tool calls in parallel.
  - In-memory Caching (custom Python dicts with TTL): to reuse recent results of LLM cleaning and tool outputs.
- Project Configuration
  - dotenv: for environment variable management (API keys and endpoints).
  - tempfile / os: for handling uploaded images securely.
- Development and Collaboration
  - Git + GitHub: version control, code sharing, and collaboration.
  - Markdown + README.md: project documentation with usage instructions.
- Mobile Application
  - Flutter (Dart): Cross-platform client for iOS/Android with smooth animations and responsive UI.
  - Local storage (sqflite): On-device database to cache scans/results for offline viewing and quick retrieval.
  - FastAPI integration (HTTP/JSON): Sends captured images to the backend and renders parsed ingredient/nutrition outputs.

3.2 Evaluation Procedure

The evaluation procedure was conducted in multiple stages to assess both the OCR and LLM components of the system. For the OCR, we initially tested several local OCR tools using food product images collected from online sources. Different preprocessing techniques were applied, resulting in over 64 combinations being evaluated. Based on these trials, we transitioned to cloud-based solutions and selected **Azure OCR**, which consistently provided higher accuracy. The evaluation criterion for OCR performance was the accuracy of extracted words compared to the ground truth text. For the LLM, the evaluation focused on the reliability and correctness of the information generated after analyzing the OCR output. This included verifying whether the extracted nutritional and ingredient details were correctly interpreted, structured, and aligned with the intended output format. Together, these procedures ensured that both modules were tested for accuracy and reliability before integration into the final system.

3.3 Steps of system operation or experiment

- Image Input – A food product label image is captured or collected and prepared for processing.
- OCR Processing – The preprocessed image is passed to **Azure OCR**, which extracts the raw text from the nutrition and ingredient sections of the label.
- Text Analysis – The extracted text is sent to a **gemini (LLM)** API, which analyzes and interprets the information.
- Structured Output Generation – The gemini LLM organizes the analyzed information into a unified JSON format containing ingredients and nutritional details.
- App Integration – The structured JSON output is delivered to the mobile application.
- User Presentation – The mobile app presents the information in a clear and user-friendly format, enabling consumers to quickly understand product contents and make informed decisions.

4 Current Progress and Milestones achieved

4.1 OCR Development

We evaluated multiple local OCR tools, including **PaddleOCR, EasyOCR, and Tesseract**, alongside cloud-based APIs such as **Google Vision and AWS Textract**. While local tools provided acceptable results on simple labels, they performed poorly on more complex inputs, such as curved or low-contrast packaging. After extensive testing, we finalized **Azure AI OCR** as the primary solution, as it consistently outperformed alternatives in both accuracy and speed. This decision ensured that the OCR stage of the pipeline would be robust enough for real-time use in the mobile application.

4.2 Reliable Text Extraction

Once **Azure AI OCR** was integrated, we achieved consistent and accurate text recognition for English food labels. This was an important milestone, as reliable text extraction formed the foundation for all subsequent analysis by the LLM. By reducing the amount of noise and errors passed forward, the OCR results significantly improved the overall stability of the pipeline. This robustness enabled us to move forward confidently with downstream interpretation and structuring.

4.3 Structured Output Design

To ensure consistency across the system, we developed a unified JSON schema for representing ingredients, nutrition facts, and warnings. This schema standardizes how different types of extracted information are stored and transferred through the pipeline. It also simplifies integration with the mobile application by enforcing a predictable and uniform data format. This milestone was key to making the system scalable and maintainable as new features or enrichment modules are added.

4.4 LLM Integration

We successfully integrated **Gemini** models into the pipeline to handle tasks such as text cleaning, enrichment (through agentic function calls), and formatting. The use of LLMs allowed the raw OCR output to be transformed into meaningful, structured insights for the user. In addition, the integration of fallback and retry mechanisms ensured that the outputs adhered to the predefined schema even in edge cases. This step enabled the system to reliably interpret ingredient and nutrition information at scale.

4.5 Dataset Preparation

Due to the absence of reliable public APIs for certain types of ingredient information, we built a local warnings dataset to capture essential health-related alerts such as allergens and additives. To further strengthen the system, we also scraped the full E-number dataset from **Wikipedia**, ensuring that this information is available offline and remains accessible even if external resources become unavailable. These datasets added resilience to the system and enabled full end-to-end testing without dependency on unstable third-party services. This preparation ensured that the LLM had reliable sources of truth for enrichment tasks.

4.6 Research Enrichment

We explored external sources such as **PubChem** and **OpenAlex** to enhance ingredient analysis through **retrieval-augmented generation (RAG)**. While some sources like **Crossref** proved unsuitable due to latency and rate limits, **OpenAlex** was responsive and effective. This enrichment highlighted the potential of research-backed knowledge to provide consumers with deeper insights.

5 Reflections on Challenges

5.1 Optical Character Recognition Challenges

- Due to the low accuracy of local OCR tools, a significant amount of time was spent testing multiple frameworks and experimenting with preprocessing techniques in an attempt to improve performance. This effort ultimately highlighted the need to shift toward cloud-based OCR APIs.
- The project faced several image-related challenges, including curved product surfaces, poor lighting conditions, and low-contrast packaging. These factors often caused OCR tools to either misread text or fail entirely.
- Balancing accuracy and efficiency was another ongoing challenge. While cloud-based OCR solutions provided strong results, they also introduced considerations such as dependency on external services and the need for stable connectivity.

5.2 Long Processing Time

- Initial pipeline execution took around 5 minutes due to sequential execution of the cleaning LLM, agentic model, and formatting LLM.
- The agent model was identified as the main bottleneck.
- Through parallelization of tool calls and caching repeated requests, the runtime was reduced progressively from 5 minutes → 3 minutes → 2 minutes → 30–50 seconds.

5.3 Output Inconsistencies

- The LLM sometimes produced slight wording variations (e.g., synonyms or phrasing differences) while conveying the same meaning.
- Although semantically correct, these inconsistencies may affect downstream integration.
- Improved a few times to make it more consistent with different methods, such as more precise prompting.
- Could potentially be mitigated with more prompt engineering tricks, schema enforcement, or post-processing normalization.

5.4 Data Availability

- Faced difficulty finding reliable databases or APIs for warnings, allergens, and additives.
- For E-numbers, no usable API was found, requiring us to scrape and locally store the entire **Wikipedia** page using Python, which improved speed.
- The warnings.csv file had to be created manually and currently contains only limited entries as a demonstration dataset.

5.5 FastAPI Setup and Deployment

- Encountered challenges while configuring **FastAPI** and managing imports locally.
- Initially the service was only available locally, requiring tunneling through Cloudflare to expose the API for external testing.
- This was necessary so that Ahmad (responsible for the Flutter app) could integrate and test the API.

6 Preliminary Results and Findings

6.1 What worked

- Cloud-based OCR tools (e.g., Azure AI) provided consistently high accuracy across varied conditions, including curved and low-contrast labels.
- Cloud-based OCR required minimal preprocessing, simplifying the pipeline while improving reliability.
- A unified JSON schema stabilized outputs and improved downstream handling.
- Text cleaning prior to LLM input reduced parsing errors and improved consistency.
- Function-calling (agentic approach) effectively orchestrated enrichment steps such as warnings, aliases, E-numbers, and reference papers.
- A local warnings.csv dataset enabled full end-to-end testing without relying on external services.
- A scraped E-number dataset from **Wikipedia** provided offline robustness and ensured availability.
- OpenAlex proved to be a more responsive and reliable source for enrichment compared to Crossref and Semantic Scholar.

6.2 What didn't work

- Local OCR tools (e.g., Tesseract, EasyOCR, PaddleOCR) performed poorly on challenging inputs such as curved product images.
- Even with multiple preprocessing techniques (unwarping, denoising, text detection), local OCR accuracy showed little improvement.
- Many local trainable OCR tools required extensive training on large datasets to reach acceptable accuracy, which was impractical within project time constraints.
- Direct OCR $\rightarrow$ LLM pipelines, without preprocessing the extracted text, resulted in noisy and inconsistent outputs.
- External enrichment services such as Crossref and Semantic Scholar were slow, occasionally rate-limited, and not well suited for real-time or near–real-time use.
- LLM outputs occasionally generated invalid JSON before formatting, requiring mitigation through automated fixes and retries.

7 Conclusion and Future Work

7.1 Conclusion

Many consumers struggle to understand the information presented on food product labels in their daily lives. Our project addresses this challenge by providing detailed ingredient and nutrition information through a simple picture capture. We successfully developed a system that delivers this functionality with high accuracy, integrated into an easy-to-use mobile application. The system enables consumers to clearly understand the contents of labeled food products, making it easier to identify healthier options that fit their individual needs.

7.2 Future Work

One of the main limitations we faced was the lack of public datasets to fine-tune or train an Optical Character Recognition (OCR) model, as well as the absence of reliable datasets for refining a Large Language Model (LLM). This limitation led us to adopt cloud-based solutions. Additionally, we found no comprehensive and reliable public database that could be used to enrich ingredient information during analysis. For future work, we aim to build a dedicated dataset to improve OCR performance, train a YOLO-based text extractor to enhance OCR results, and fine-tune an LLM on a food products dataset. Another important step will be constructing a specialized ingredients database that can provide accurate and consistent information for the LLM to reference, making the system more robust and reliable.

References

[1] International Food Information Council, "2021 Food & Health Survey," 2021. [Online]. Available: `https://foodinsight.org/2021-food-health-survey/`

[2] Microsoft, "Read text in images and documents with the OCR capability," Azure AI Vision documentation, 2024. [Online]. Available: `https://learn.microsoft.com/en-us/azure/ai-services/computer-vision/overview-ocr`

[3] Google DeepMind, "Gemini: A family of multimodal models," Google AI, 2024. [Online]. Available: `https://deepmind.google/technologies/gemini/`

[4] PaddleOCR Team, "PaddleOCR: Open-source OCR toolkit," GitHub repository, 2024. [Online]. Available: `https://github.com/PaddlePaddle/PaddleOCR`

[5] JaidedAI, "EasyOCR: Ready-to-use OCR with 80+ supported languages," GitHub repository, 2024. [Online]. Available: `https://github.com/JaidedAI/EasyOCR`

[6] R. Smith, "An overview of the Tesseract OCR engine," in *Proc. 9th Int. Conf. Document Analysis and Recognition (ICDAR)*, pp. 629–633, 2007. doi: 10.1109/ICDAR.2007.4376991

[7] M. Li, Y. Li, L. Cui, S. Huang, F. Wei, and M. Zhou, "TrOCR: Transformer-based optical character recognition with pre-trained models," *arXiv preprint arXiv:2109.10282*, 2021. [Online]. Available: `https://arxiv.org/abs/2109.10282`

[8] Google Cloud, "Cloud Vision API," 2024. [Online]. Available: `https://cloud.google.com/vision`

[9] Amazon Web Services, "Amazon Textract," 2024. [Online]. Available: `https://aws.amazon.com/textract/`

[10] National Center for Biotechnology Information, "PubChem," National Library of Medicine, 2024. [Online]. Available: `https://pubchem.ncbi.nlm.nih.gov/`

[11] OpenAlex, "OpenAlex: The open catalog of the global research system," 2024. [Online]. Available: `https://openalex.org/`