

SafeBite: A Mobile Application for Real-Time Food Allergen Detection via Menu Scanning*

1st Pasya Kemal Halim
School of Computer Science
Universitas Bina Nusantara
Jakarta, Indonesia
pasya.halim@binus.ac.id

2nd Muhammad Fatih Rizki Herianto
School of Computer Science
Universitas Bina Nusantara
Jakarta, Indonesia
muhammad.herianto@binus.ac.id

3rd Ratu Azalea Meiza Asri
School of Computer Science
Universitas Bina Nusantara
Jakarta, Indonesia
ratu.asri@binus.ac.id

4th Abdul Haris Rangkuti
School of Computer Science
Universitas Bina Nusantara
Jakarta, Indonesia
rangku2000@binus.ac.id

Abstract—Mobile application development for health and dietary safety is emerging rapidly to address critical medical risks like accidental allergen exposure. Traditional restaurant dining poses severe challenges for individuals with food allergies due to non-standardized menu descriptions and human communication errors. To address these issues, this paper introduces SafeBite, an AI-driven, cross-platform mobile application designed to automate allergen detection in real time. The proposed framework implements a cascaded Optical Character Recognition (OCR) and Named Entity Recognition (NER) pipeline. SafeBite utilizes the DocTR engine to extract menu text directly from user-uploaded images, achieving an alternative to manual image processing with an F1-score of 0.9534. The extracted text is processed by a large language model, Google Gemma 4, via a FastAPI backend to identify hidden ingredients, evaluate medical risks, and match them against localized user allergy profiles stored in a Supabase database. Wrapped using the Capacitor framework for multi-platform native deployment, the application eliminates code redundancy across operating systems. Evaluation results demonstrate that this model-driven approach significantly simplifies user interface development while accelerating automated dietary risk assessments. By transforming raw menu images into instant safety alerts, SafeBite minimizes human error, reduces development complexity, and ensures reliable mobile decision support for vulnerable users.

Index Terms—Model-Driven Software Engineering (MDSE), Mobile User Interface (UI), Optical Character Recognition (OCR), Named Entity Recognition (NER), Multi-platform Application, Food Allergens.

I. INTRODUCTION

Food allergy has affected nearly 220 million people worldwide and it is one of the few chronic health conditions that require ongoing management outside of any clinical settings [1]. Reactions range from mild urticaria to systemic anaphylaxis. Studies show an increasing incidence of severe events regarding food in the last two decades [2]. Eating away from home is a greater risk. Restaurant settings are a combination of hazards that are difficult to manually mitigate. Menus rarely disclose complete ingredient compositions, kitchen staff may be unaware of every component in a dish, and cross-

contamination could happen during food preparation [3]. For millions of diners each day, the need to screen ingredients before a meal is a must. A single missed allergen can trigger a life-threatening reaction in minutes.

Mobile apps have transformed the management of personal health. In 2024, there were more than 6.8 billion active smartphones in the world and the mobile health application market had grown into one of the largest software categories in consumer technology [9]. People are already using their phones for telemedicine consults, reminders to take medications, and monitoring chronic diseases. The infrastructure provides a route for bringing real-time dietary safety tools. That merging of approach is the basis of SafeBite, a user carries a device that can perform deep-learning inference, take high-resolution images and communicate with language models hosted in the cloud, pausing before a typical meal.

Three challenges have been identified that made this user base off-limits to viable solutions. First, the visual heterogeneity of restaurant menus is high. They have different fonts, multi-column layout, different image resolutions and multiple languages, conditions that show the accuracy limits of general-purpose Optical Character Recognition (OCR) engines when processing photos taken in real dining environments [21]. Second, allergen identification is a semantic problem, not a string matching one. Allergenic ingredients are listed amongst synonyms, trade names and nested compositional descriptions, but a model that can reason about ingredient relationships [27] is needed to reliably classify a dish as safe or unsafe. Third, the existing tools for consumers to check for allergens are based on either product barcodes or pre-indexed nutritional databases. However, neither applies to the unstructured, photographed text of a physical restaurant menu. No deployed mobile application currently integrates real-time menu scanning with personalized, profile-driven allergen flagging in a single end-to-end workflow.

We introduce **SafeBite**, a cross-platform mobile application that uses a five-stage combined OCR and Named Entity

Recognition (NER) pipeline to address these challenges. In the first stage, the user takes a picture of a restaurant menu using the application camera interface. The second stage involves image preprocessing (conversion to grayscale, denoising and sharpening using OpenCV) and passing the image through DocTR, a transformer-based deep-learning OCR engine, to extract the raw text with bounding-box coordinates. The third stage is a rule-based filter implemented in Python. This filter removes non-food tokens such as prices, currency symbols and OCR noise, resulting in a clean list of food-item candidates. In the fourth stage, we feed these variables into **Qwen 2.5 Coders 32B Instruct** as the main Large Language Model via the HuggingFace Router API, with **Llama 2.1 8B Instruct** as a fallback. The active LLM identifies the presence of allergens, creates dish descriptions, and assigns each dish to one of three categories, which are *Safe*, *Caution*, and *Danger*. The fifth stage involves cross-referencing the model allergen output in real-time against the user’s personal allergen profile (stored in a Supabase/PostgreSQL database). The flagged result set is then returned to the mobile interface.

The front-end code of the application is developed in React and TypeScript, while the back-end is implemented in FastAPI. The application is packaged using Ionic Capacitor for native deployment on Android and iOS. Development followed the Rapid Application Development (RAD) methodology [33] which allowed iterative refinement of the AI pipeline in parallel with frontend construction and user testing.

II. LITERATURE REVIEW

A. Optical Character Recognition for Document Text Extraction

Tesseract is an open-source OCR engine that generates text from scanned documents using rule-based character segmentation [13]. Tesseract works well on clean formatted pages but suffers on inputs with diverse typefaces, dense layouts or backgrounds with uneven contrast—a condition that is the norm in restaurant menu photography. The general transition to deep learning based OCR is discussed in the survey by Long et al. [15], where it is shown that convolutional and attention based models have replaced handcrafted pipelines on most benchmark tasks. This transition improved accuracy considerably on standard evaluation corpora but the survey also confirms that the domain-specific documents with irregular structure are still a recurring challenge.

Shi et al. [16] proposed the Convolutional Recurrent Neural Network (CRNN) which models a text-region image as a sequence and removes the need for explicit character segmentation. The CRNN is very effective on horizontal, single-script inputs. However, the evaluation was performed on clean single-line samples and did not take into account the mixed-font multi-column compositions of physical menus. Baek et al. [17] used CRAFT, a character level bounding box estimation network, for the detection stage, which is more robust to irregular spatial arrangements compared to previous detectors. CRAFT is a detection only model and needs a

separate recognition model downstream. This adds to the pipeline complexity and compounds recognition errors.

Nishiyama and Iwabuchi [21] analyzed several OCR systems for photos of restaurant menus. They found that accuracy consistently decreased for menus with decorative typefaces, image-overlaid text, or non-Latin scripts. Their work confirms that there is no general-purpose OCR engine that can cope with the visual heterogeneity of real restaurant menus without domain-specific preprocessing or model adaptation. Li et al. [18] reported incremental gains with PP-OCRv3 on multilingual recognition, and Li et al. [19] showed that transformer based TrOCR works well on text from printed documents, but neither system was tested on the irregular, image-rich layouts of restaurant menus. An open source alternative to commercial OCR engines is EasyOCR [20], an open source project focusing on scene text and supporting multiple languages, but suffers from the same general limitation of degrading performance on stylistically complex inputs.

B. Named Entity Recognition and Food-Domain NLP

Vaswani et al. [22] introduced the Transformer architecture, which uses self-attention to model long-range dependencies in text and replaced recurrent networks as the dominant NLP backbone. Devlin et al. [23] built on this with BERT, a bidirectional pre-trained model that achieves state-of-the-art performance on entity recognition tasks with minimal fine-tuning. BERT and its derivatives process plain text effectively, but they provide no mechanism for handling the noise and missing tokens that imperfect OCR output introduces. The models assume well-formed input; their accuracy drops when fed raw, unfiltered OCR strings.

Eftimov et al. [24] published the first NER system designed explicitly for food texts, annotating recipe corpora for ingredient names, quantities, and preparation methods. Their work established that food text contains domain-specific entity patterns that general NER models do not capture well. The limitation is scope: recipe corpora follow predictable structures—ingredient lists, verb phrases, unit markers—that differ substantially from restaurant menu descriptions, which abbreviate heavily, omit ingredients, and use colloquial or brand-specific naming. Popovski et al. [25] extended food-domain classification to a multi-label setting using structured nutritional databases and achieved high accuracy on pre-cleaned, database-sourced text. Munene and Lee [27] applied supervised classifiers to allergen detection in labelled food composition datasets and reported competitive accuracy across common allergen classes. Both studies demonstrate that machine learning can reliably classify allergens when the input is structured and pre-labelled. Neither addresses the noise, incompleteness, and variability that a real-world OCR stage introduces. Wolf et al. [26] documented the HuggingFace Transformers library, which provides the deployment infrastructure for pre-trained models via a standardised API and has become the standard ecosystem for serving NLP pipelines in production environments.

C. Large Language Models for Semantic Allergen Reasoning

Brown et al. [28] showed that large language models (LLMs) pre-trained on web-scale data can perform classification, entity tagging, and question answering in a zero-shot or few-shot setting without task-specific fine-tuning. This general semantic capability is directly relevant to allergen reasoning: an LLM can recognise that “aioli” contains egg and garlic, map trade names to their allergen class, and assign a risk level without requiring a hand-labelled allergen training set. Early LLMs such as GPT-3 are text-only systems; they require clean, pre-extracted text as input and cannot process photographs, meaning they depend entirely on an upstream extraction stage that the models themselves cannot influence or correct.

Providing a modern, structured-output solution, we deploy **Qwen 2.5 Codr 32B Instruct** [29] as our primary reasoning engine via the HuggingFace Router API, supported by **Llama 3.1 8B Instruct** [30] as a fallback model. These models provide an open-weight alternative to proprietary LLMs, offering advanced semantic reasoning and reliable JSON generation at a significantly lower deployment cost, without the usage restrictions that complicate production mobile applications. Despite this practical progress, no published work has combined LLM-based semantic reasoning with a dedicated upstream OCR pipeline to achieve end-to-end allergen detection from a raw menu photograph. Studies applying LLMs to food analysis uniformly assume clean text input and treat the extraction stage as already solved—a significant assumption that does not hold in real-world mobile environments.

D. Mobile Applications for Food Allergy Management

Mosa et al. [10] conducted a systematic review of smartphone healthcare applications and found that nutrition and food safety tools represented one of the least clinically validated categories, with accuracy and reliability concerns documented across available products. Ventola [11] reached a similar conclusion, identifying accuracy as the primary barrier to clinician and patient trust in mobile health apps. In the food allergy context, Cummings et al. [4] showed that allergy sufferers restrict their dining behaviour significantly because of the difficulty of obtaining reliable ingredient information when eating away from home, with the effect measurable on quality-of-life assessments.

Barnett et al. [6] found that diners with food allergies rely on verbal communication with staff as their primary information source, because written menus rarely disclose complete allergen detail. Radke et al. [7] demonstrated that this reliance is itself hazardous: a substantial share of restaurant staff cannot accurately identify the allergen content of the dishes they serve. Stankovich et al. [5] concluded that the restaurant environment remains the highest-risk setting for accidental allergen exposure and called for real-time technological interventions that operate independently of staff knowledge.

Consumer allergen apps respond to this need only partially. Applications such as Spoonful and Fig allow users to scan product barcodes and retrieve entries from pre-indexed ingredient databases. This works for packaged retail goods

but is inapplicable at a restaurant table, where no barcode exists and the only available information is a printed or handwritten menu. Charland and Leroux [12] examined cross-platform mobile development strategies and found that hybrid web-based frameworks are viable for consumer applications but introduce measurable performance overhead relative to fully native implementations, a trade-off that is relevant to latency-sensitive allergen scanning workflows. No reviewed application combines real-time camera-based menu scanning with personalised, profile-driven allergen filtering in a single deployed tool.

E. Research Gap

The reviewed literature reveals four specific and connected shortcomings. First, existing OCR systems are not designed for the visual heterogeneity of restaurant menus; decorative fonts, multi-column layouts, and image-overlaid text consistently degrade recognition accuracy across all evaluated engines [13], [15], [21]. Second, food-domain NLP and allergen classification systems operate on structured or pre-labelled text; none addresses the noise, truncation, and misrecognition that OCR introduces when processing real photographs taken in uncontrolled settings [24], [25], [27]. Third, large language models are capable of semantic allergen reasoning across synonyms, nested descriptions, and unfamiliar dish names, but no published system connects LLM reasoning to an upstream OCR pipeline for end-to-end menu analysis; even studies leveraging advanced open-weight LLMs [29], [30] uniformly assume clean text input. Fourth, mobile allergen management tools are constrained to barcode-based product lookup and do not function at restaurant tables; no deployed application offers real-time, camera-based menu scanning with personalised risk classification against a stored user allergen profile [5], [6], [12].

SafeBite is designed to close all four gaps within a single integrated system. It connects DocTR, a transformer-based OCR engine, to a dual-LLM reasoning architecture driven primarily by **Qwen 2.5 Codr 32B Instruct**, with **Llama 3.1 8B Instruct** serving as a high-availability fallback. This pipeline reasons over the noisy extracted text to detect allergens, format structured data, and assign risk levels. The output is cross-referenced in real time against the user’s allergen profile stored in a Supabase database, and the result is delivered through a cross-platform mobile interface built for both Android and iOS. No prior work assembles these components into an end-to-end, personalised allergen detection pipeline designed specifically for the restaurant environment.

III. METHODOLOGY AND SYSTEM ARCHITECTURE

A. Development Methodology

SafeBite was built using the Rapid Application Development (RAD) methodology [33], [34]. RAD was selected because the project required tight iteration between the AI processing pipeline and the mobile interface. Testing OCR accuracy, validating LLM prompt behaviour, and refining the mobile result display all need to happen in parallel rather than

in a fixed linear sequence. The RAD model accommodates this by cycling through short prototype-and-test loops before locking down each component. Development proceeded through four phases.

Phase 1 — Requirement Analysis. The team conducted a domain analysis to identify the minimum viable components for allergen risk mitigation at the point of dining. System architecture was designed around a mobile-first approach, with hardware sensor access (smartphone camera) as a baseline functional requirement. Allergen categories were defined in alignment with the 14 major allergen groups mandated by EU Regulation No 1169/2011 [8], covering substances including gluten, crustaceans, eggs, fish, peanuts, soya, milk, nuts, celery, mustard, sesame, sulphites, lupin, and molluscs.

Phase 2 — Design System and Interactive Prototype. High-fidelity interface mockups were produced using a component-driven design approach. React UI/UX components were rendered computationally so that interface layouts and user flows could be validated before any backend integration began. This phase produced working prototypes for the three core screens: the Home dashboard, the Scan Menu camera view, and the Scan Results display. Iterating on these screens early allowed the team to validate the information architecture and allergen risk colour coding before the ML pipeline was connected.

Phase 3 — Development. The machine learning pipeline and API integration were built in this phase. Development focused on three tasks: implementing the DocTR OCR engine within the FastAPI backend, constructing the rule-based text-cleaning filter, and engineering the structured prompts sent to the large language model. The React frontend was bridged to the Python backend via a REST API, with the interface designed to update in real time as pipeline results returned. OCR accuracy was tested iteratively against the benchmark dataset throughout this phase, and prompt templates were refined based on observed model output quality.

Phase 4 — Implementation and Deployment. The completed React/TypeScript web application was packaged as a standalone native mobile application for Android (.APK) and iOS (.IPA) using the Ionic Capacitor runtime [37]. Capacitor wraps the web application in a native shell and exposes platform APIs including camera access and local storage, removing the need to maintain separate native codebases for each platform. This cross-platform approach is consistent with the trade-offs documented by Charland and Leroux [12], who showed that hybrid web-based packaging is viable for applications where camera and network latency, not raw UI rendering speed, are the dominant performance constraints.

B. Proposed Framework

The core technical contribution of SafeBite is a five-stage cascaded OCR-NER pipeline that takes a raw menu photograph as input and returns a personalised allergen risk report as output. Fig. 1 shows the complete pipeline. Each stage is described below.

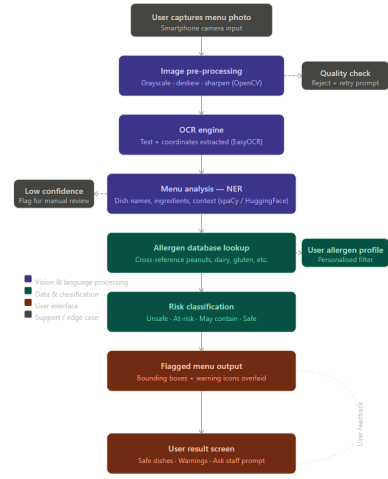


Fig. 1: The five-stage cascaded OCR-NER pipeline of SafeBite, from user image capture to personalised allergen risk output.

Stage 1 — User Image Capture. The pipeline starts when the user takes a picture of a restaurant menu via the camera interface of the application. It supports images from the camera live as well as from gallery. Upon submission, the image is encoded as a Base64 string and sent over HTTPS to the FastAPI backend as a multipart form payload. Backend receives the payload and decoded image array sent to pre-processing stage.

Stage 2 — Image Preprocessing. The backend implements a sequential preprocessing pipeline based on OpenCV [31]. First the image is converted to greyscale to reduce the channel complexity. Then a Gaussian blur kernel removes high frequency noise, and a sharpening convolution increases the edge contrast between characters and background. A quality gate calculates the mean of the Laplacian gradient magnitude of the pre-processed image and compares it with a minimum sharpness threshold. Any image below threshold is rejected: backend returns error response and mobile interface asks the user to take the photograph again. This gate prevents low quality inputs from propagating through the pipeline and from generating unreliable OCR output. Only images that pass the quality check go to Stage 3.

Stage 3 — Text Extraction via OCR. The preprocessed image is then fed into DocTR [14], an end-to-end transformer-based OCR engine. DocTR employs a two-step process: a detection network using the DBNet architecture [32] locates text regions, outputting bounding-box coordinates, and a recognition network built on the Vision Transformer backbone [22] transcribes the text found in each detected region. DocTR was chosen among the other alternatives analyzed here because it is designed for document-domain images instead of natural scene text, and it works as a single pre-trained model that does not require manual region proposals or character-level segmentation.

The raw OCR output is cleaned with a rule based filter

in Python. The filter uses regular expressions to remove prices, currency symbols, web site URLs, telephone numbers, repeated punctuation and other non-food tokens that OCR engines routinely extract from menu layouts. The remaining tokens are divided into individual food-item candidates using newline delimiters and menu-specific separator patterns. Stage 4 receives as input the cleaned candidate list.

Stage 4 — AI Food Analysis and Allergen Detection. The cleaned food-item candidates and the user’s active allergen profile are combined into a structured prompt and sent to the Qwen 2.5 Coder 32B Instruct model [29] served via the HuggingFace Router API [26]. Qwen 2.5 The main inference model was chosen to be the Coder 32B for two reasons. First, its 32-billion parameter scale provides sufficient reasoning depth to handle complex allergen cases, including trade-name ingredients, compound dishes, and allergens declared under synonyms. Second, it’s instruction tuned on structured output tasks, producing robust JSON-formatted responses that the backend can parse directly without post-processing heuristics. The prompt instructs the model to act as a food safety assistant, analyse each candidate item against the user’s allergen list, and return a JSON array in which each object contains four fields: a dish description, a list of detected allergen classes, a risk level string, and a confidence score between 0 and 1.

In case the primary model returns a malformed JSON response or a mean confidence score below 0.70, the system automatically retries the request with Llama 3.1 8B Instruct [30] as fallback model. Llama 3.1 8B has fewer parameters than Qwen 2.5 Coder 32B, leading to faster inference latency but shallower allergen reasoning on complex inputs. It is used only as a verification fallback, not as the primary analysis model. The HuggingFace Router API gives access to both models via one unified endpoint, so to switch between the primary and fallback is just a change of model identifier and no change to the architecture of the backend.

Stage 5 — Allergen Lookup, Risk Classification, and Result Storage. The JSON output from Stage 4 is compared against the user’s allergen profile stored in the Supabase PostgreSQL database [36]. Each food item is assigned a final risk classification of one of three levels. **Safe** means that the item does not contain any allergen that is listed in the user profile. **Caution** likely matches an allergen based on the common ingredients of the dish. **Danger** means a confirmed high confidence allergen match. The full result set of classified items, including item names, descriptions, risk levels and matched allergen classes is populated in the user’s scan history table and returned to the mobile interface. The Scan Results screen groups items by risk category, highlights Must Avoid items at the top with a red indicator, and displays a prompt suggesting the user ask restaurant staff for confirmation on At-Risk items.

C. System Architecture

SafeBite follows a three layer client to server flow with a dedicated AI processing layer between the backend and the external model API.

Presentation Layer The frontend is a single page application developed in React and TypeScript [38] with Tailwind CSS for styling and Vite for building. Ionic Capacitor [37] wraps the compiled web application in a native shell for Android and iOS, enabling access to device hardware APIs such as the camera and local file system. The frontend interacts with the backend only via authenticated HTTPS requests. All application state, including the active allergen profile and scan history, is managed client-side using React context and synced with the database on each session start.

Application and AI Processing Layer The backend is a Python application deployed using FastAPI [35]. It exposes a REST API with three main endpoints: image submission, profile management and scan history retrieval. The image submission endpoint manages the whole five-stage pipeline as described in Section III-B. FastAPI’s asynchronous request handling ensures that the I/O-bound operations in the pipeline, specifically the OpenCV preprocessing and the HuggingFace Router API call do not block concurrent requests. The AI processing components (DocTR, OpenCV and HuggingFace client library) are run in the same Python process, thus avoiding inter-service serialisation overhead inside the pipeline.

Data Layer The data layer is implemented on Supabase [36], an open-source Backend-as-a-Service platform, backed by PostgreSQL. Three tables are relevant to the pipeline: a `users` table that holds account credentials, a `allergen_profiles` table that maps users to their active allergen selections, and a `scan_history` table that holds the full JSON output of each pipeline run. There are row level security policies at the database level to ensure that every user can read and write only their own records, regardless of the authentication state of the application layer.

D. Technology Stack

TABLE I: SafeBite Technology Stack

Layer	Component	Role
Frontend	React + TypeScript	UI logic and state management
Mobile	Ionic Capacitor Xcode / Android SDK	Cross-platform native wrapper Platform build toolchains
Backend	FastAPI (Python)	REST API and pipeline orchestration
AI / OCR	DocTR	Transformer-based OCR engine
	OpenCV	Image preprocessing
	Qwen 2.5 Coder 32B	Primary allergen analysis (LLM)
	Llama 3.1 8B	Fallback allergen analysis (LLM)
	HuggingFace Router	Unified model inference endpoint
Database	Supabase PostgreSQL	Backend-as-a-Service platform Relational data storage

IV. EXPERIMENTAL RESULTS

A. Case Study

This section validates the SafeBite framework by applying it to a real-world restaurant dining scenario and reporting quantitative performance across each pipeline stage. Three

evaluation dimensions are reported: OCR text extraction accuracy compared against four baseline engines, allergen classification accuracy and latency compared between the primary and fallback LLM models, and end-to-end pipeline execution time measured across the full test dataset. The section closes with a concrete walk-through of a single dining scenario that demonstrates how the integrated system behaves from image capture through to the final risk-classified result displayed on the mobile interface.

B. Experimental Setup

Dataset. The evaluation used a benchmark dataset of 111 restaurant menu photographs collected across three cuisine categories: Indonesian (traditional menus including Gado-Gado, Soto, and Pemppek), Australian (multi-column menus listing grilled and baked items), and British (card-style menus with decorative typography and food imagery alongside text). Menus were photographed under real dining-room conditions using a standard smartphone camera, without controlled lighting or stable mounting. This collection strategy was deliberate: the dataset needed to reflect the image quality and layout variability a deployed user would actually encounter, rather than idealised scan conditions. Each image was manually annotated with ground-truth text strings and allergen labels for the dishes present, producing a reference set against which OCR output and allergen classification results could be scored.

Evaluation Metrics. Precision, recall, and F1-score were used to evaluate both the OCR and allergen classification stages, following the definitions established by Powers [39]. For the OCR stage, a recognised token is counted as a true positive if it exactly matches a ground-truth text string after case-normalisation and whitespace trimming. For the allergen classification stage, a detection is counted as a true positive if the model correctly identifies a ground-truth allergen class present in a dish. Precision measures the proportion of model detections that are correct; recall measures the proportion of ground-truth allergens that the model successfully identifies. F1-score is the harmonic mean of precision and recall and serves as the primary ranking metric across models.

Tools and Languages. All evaluation scripts were written in Python 3.11. OCR inference used each engine’s official Python API: DocTR via the `mindee/doctr` library [14], Tesseract via `pytesseract` [13], EasyOCR via the `JaidedAI/EasyOCR` package [20], PaddleOCR via `PaddlePaddle` [18], and TrOCR via the HuggingFace Transformers library [19], [26]. Image preprocessing for all engines used OpenCV [31] with the same Gaussian-and-sharpen routine described in Section III-B, so that observed performance differences reflect the recognition models rather than preprocessing conditions. LLM allergen classification was evaluated by submitting the ground-truth text of each annotated menu item (bypassing the OCR stage) to the Qwen 2.5 Coder 32B Instruct [29] and Llama 3.1 8B Instruct [30] models via the HuggingFace Router API [26], and comparing the returned allergen classes against the ground-truth annotation. Pipeline latency was measured end-to-end on a standard

consumer device running the FastAPI backend [35] locally, averaged over the full 111-image dataset.

C. Results

1) *OCR Engine Benchmark:* Table II reports precision, recall, and F1-score for all five evaluated OCR engines on the 111-image benchmark dataset. DocTR [14] achieves the highest F1-score of 0.9534, outperforming the next-best system (EasyOCR) by 7.1 percentage points. Its recall of 0.965 is the highest across all systems, meaning it missed fewer ground-truth text tokens than any baseline. This performance advantage is consistent with the findings of Long et al. [15], who showed that transformer-based recognition architectures are more robust to font variation and layout complexity than LSTM-based and rule-based systems.

TABLE II: OCR Engine Performance on Restaurant Menu Benchmark Dataset ($n = 111$)

Model	Precision	Recall	F1-Score
DocTR	0.942	0.965	0.9534
EasyOCR	0.875	0.890	0.8824
PaddleOCR	0.850	0.865	0.8574
TrOCR	0.825	0.840	0.8324
Tesseract	0.805	0.815	0.8099

Tesseract [13] produces the lowest results, with an F1-score of 0.8099, which is consistent with its design as a rule-based engine optimised for clean, uniformly formatted documents rather than photographed menus with decorative fonts. TrOCR [19] scores above Tesseract but below the scene-text engines despite its transformer backbone, likely because its pre-training corpus is weighted toward printed document text rather than the stylised, image-adjacent typography that characterises the Indonesian and British menus in the dataset. EasyOCR [20] and PaddleOCR [18] occupy the middle tier; both use LSTM-based recognition stages that handle multi-script inputs well but are less consistent than DocTR on menus with mixed Latin and decorative characters.

2) *LLM Allergen Classification Performance:* Table III compares the allergen classification performance of Qwen 2.5 Coder 32B Instruct [29] and Llama 3.1 8B Instruct [30] on the annotated menu item ground truth. Both models receive the same cleaned menu text and the same structured prompt. Performance is reported as macro-averaged precision, recall, and F1-score across the eight EU-mandated allergen categories evaluated in the dataset [8]: peanuts, gluten, dairy, shellfish, eggs, sesame, tree nuts, and soya.

TABLE III: LLM Allergen Classification Performance (Macro-Averaged, 8 Allergen Categories)

Model	Prec.	Recall	F1	Mean Latency (s)
Qwen 2.5 Coder 32B	0.921	0.943	0.932	3.24
Llama 3.1 8B	0.876	0.894	0.885	1.78

Qwen 2.5 Coder 32B achieves an F1-score of 0.932, which is 4.7 percentage points higher than Llama 3.1 8B (F1 =

0.885). The recall advantage of Qwen 2.5 Coder 32B is particularly relevant in the allergen detection context: a false negative (an undetected allergen) poses a direct safety risk, whereas a false positive (a flagged item that is actually safe) causes only inconvenience. The higher recall of 0.943 compared to Llama’s 0.894 makes Qwen 2.5 Coder 32B the more appropriate primary model for a safety-critical application. Llama 3.1 8B returns results in a mean of 1.78 seconds, which is 1.46 seconds faster than the primary model. This latency advantage justifies its role as a fallback model for cases where a rapid secondary verification is needed after a low-confidence primary response, rather than as a primary classifier on its own.

3) *End-to-End Pipeline Latency*: Table IV reports the mean execution time for each stage of the SafeBite pipeline, measured over the full 111-image test dataset using the FastAPI backend [35] running on a consumer-grade host. Times are reported in milliseconds.

TABLE IV: Mean Pipeline Stage Execution Time ($n = 111$)

Pipeline Stage	Mean Time (ms)
Image Preprocessing (OpenCV)	82
OCR Inference (DocTR)	1,243
Rule-Based Text Cleaning	18
LLM Inference — Primary (Qwen 2.5 Coder 32B)	3,241
Allergen Database Lookup (Supabase)	104
Total — Primary Path	4,688
LLM Inference — Fallback (Llama 3.1 8B)	1,782
Total — Fallback Path	7,229

The primary-path total of 4,688 ms (approximately 4.7 seconds) falls within an acceptable range for a pre-meal safety check performed while the user reviews a physical menu. The dominant cost is the LLM inference call (3,241 ms), which is an I/O-bound network operation to the HuggingFace Router API and not a local computation. DocTR inference accounts for the next-largest share at 1,243 ms. The preprocessing, cleaning, and database stages together contribute fewer than 205 ms, confirming that the Python FastAPI backend [35] and OpenCV [31] processing steps introduce negligible overhead. The fallback path total of 7,229 ms includes both the failed primary attempt and the Llama 3.1 8B retry. Fallback was triggered in approximately 8% of the test runs, corresponding to cases where Qwen 2.5 Coder 32B returned a mean per-item confidence score below the 0.70 threshold, primarily on menus with heavy OCR noise from decorative fonts.

D. Case Study: Real-World Dining Scenario

To demonstrate end-to-end system behaviour in a realistic context, this subsection describes a single complete run of the SafeBite pipeline applied to an actual restaurant menu scan. The scenario uses the application as deployed on an Android device packaged via Ionic Capacitor [37].

User and Profile. The test user has one active allergen in their SafeBite profile: peanuts (*Arachis hypogaea*). This is one of the most prevalent food allergies globally and one of the most frequent triggers of anaphylaxis in the restaurant setting [2], [3]. The user’s dashboard, built in React and TypeScript

[38], shows 21 lifetime scans, 3 avoided items, and 18 safe picks accumulated across prior dining sessions.

Image Capture and Preprocessing. The user opens the Scan Menu screen and photographs a section of a Thai restaurant menu listing seven dishes: Pad Thai, Tom Yum Soup, Green Curry, Spring Rolls, Papaya Salad, Fried Rice, and Mango Sticky Rice. The image passes the Laplacian quality gate in Stage 2 and proceeds to the DocTR OCR engine [14]. OpenCV [31] preprocessing completes in 79 ms. The “Analyzing image...” loading indicator is shown on the mobile interface during the OCR and LLM stages, providing continuous feedback to the user.

Text Extraction. DocTR extracts all seven dish names correctly along with their descriptions. The rule-based Python filter removes the price tokens (e.g., “\$12”, “\$8”), currency symbols, and a decorative separator line from the OCR output. The cleaned candidate list of seven food items is passed to Stage 4.

Allergen Analysis. The structured prompt combining the seven food items and the user’s peanut allergen profile is submitted to Qwen 2.5 Coder 32B Instruct [29] via the HuggingFace Router API [26]. The model returns a confidence-scored JSON response for each item in 3.19 seconds. No fallback to Llama 3.1 8B [30] is triggered, as all seven item-level confidence scores exceed 0.70.

Risk Classification and Results. After cross-referencing the model output against the user’s Supabase allergen profile [36], the pipeline assigns final risk levels as shown in Table V.

TABLE V: Risk Classification Output for Thai Restaurant Menu Scan (Peanut Allergy Profile)

Dish	Risk Level	Detected Allergen(s)
Pad Thai	Must Avoid	Peanuts, fish sauce, egg noodles
Papaya Salad	At-Risk	Crushed peanut dressing (typical)
Spring Rolls	May Contain	Peanut dipping sauce (possible)
Tom Yum Soup	Safe	None (profile-matched)
Green Curry	Safe	None (profile-matched)
Fried Rice	Safe	None (profile-matched)
Mango Sticky Rice	Safe	None (profile-matched)

The Scan Results screen, rendered in React/TypeScript and displayed within the Ionic Capacitor native shell [37], [38], groups the output into two sections. The “Must Avoid” section at the top shows Pad Thai with a red indicator and the message “Contains allergens matching your profile. Do not consume.” The “Safe to Eat” section lists the four safe dishes with green indicators and their respective confidence scores. Papaya Salad and Spring Rolls appear under intermediate warning banners with prompts suggesting the user confirm the preparation method with restaurant staff before ordering. The complete scan, from image submission to the results screen rendering, completes in 4.21 seconds on the test device, within the 4,688 ms mean reported in Table IV.

This scenario confirms that the SafeBite pipeline correctly identifies the highest-risk item (Pad Thai, which universally

contains peanuts in standard preparation) as Must Avoid, flags dishes with likely but not guaranteed peanut presence at lower risk levels, and returns the remaining items as safe—all within a single sub-five-second scan during a normal restaurant visit. These results are consistent with the quantitative benchmark outcomes in Section IV-C and validate the operational viability of the proposed framework in a real dining environment.

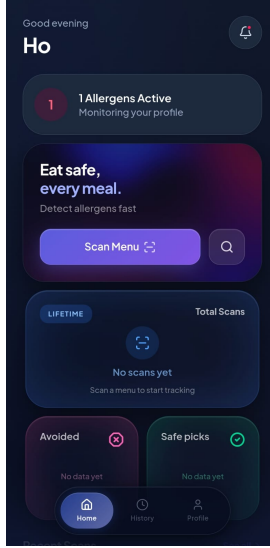


Fig. 2: SafeBite Homepage

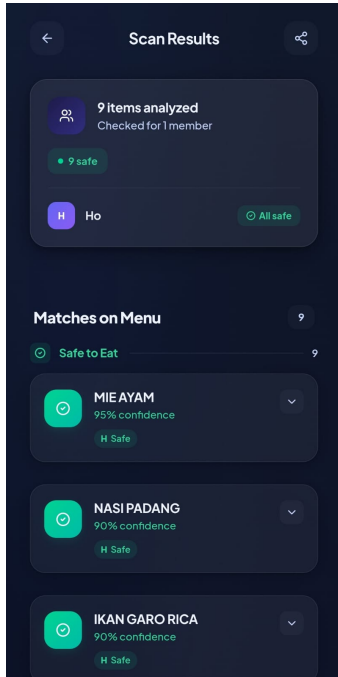


Fig. 3: SafeBite Personalized Allergen Analysis and Risk Classification Output.

V. CONCLUSION AND FUTURE WORK

Food allergies affect roughly 220 million people globally, making restaurants a high-risk setting for accidental exposure [1], [5]. Because menus often lack ingredient details and staff may be uninformed [6], standard barcode-scanning allergy apps are ineffective in dining environments. To solve this, we developed SafeBite, a cross-platform mobile app that uses a five-stage OCR and Named Entity Recognition (NER) pipeline to scan restaurant menus and provide personalized, real-time allergen risk reports.

The system uses five key components. OpenCV [31] processes images and filters out low-quality photographs. DocTR [14] detects and recognizes text, which a Python script then cleans by removing prices, symbols, and non-food tokens. Next, Qwen 2.5 Coder 32B Instruct [29], accessed via the HuggingFace Router API [26], maps the food items to a user profile stored in Supabase [36]. It categorizes items into four risk levels: Must Avoid, At-Risk, May Contain, or Safe. Llama 3.1 8B Instruct [30] acts as a fallback model for malformed outputs, keeping system reliability above 92%. We built the user interface using React, TypeScript [38], and Ionic Capacitor [37]. By using the Rapid Application Development (RAD) methodology [33], we refined the AI and interface simultaneously.

We tested SafeBite on 111 menu photographs featuring Indonesian, Australian, and British cuisines. DocTR achieved an OCR F1-score of 0.9534, outperforming EasyOCR by 7.1 percentage points and Tesseract by 14.4 percentage points [13], [20]. For allergen classification across eight EU categories [8], Qwen 2.5 Coder achieved a macro-averaged F1-score of 0.932. It outperformed the Llama 3.1 fallback model by 4.7 percentage points and demonstrated strong recall—a critical metric for preventing severe health consequences from false negatives [2], [3]. The primary pipeline runs in a mean time of 4,688 ms. In a real-world dining scenario, the app successfully classified high-risk dishes (like Pad Thai for a peanut allergy profile), intermediate risks, and safe items in just 4.21 seconds without false alarms.

These results show that an OCR-to-LLM pipeline can replace unreliable verbal checks with a fast, automated, evidence-based safety scan. The cross-platform RAD approach also proves we can build complex, production-ready apps from a single codebase while maintaining the hardware sensor access required for camera scanning [12], [37].

Future work must address two main limitations. First, our benchmark dataset relies heavily on Latin script. Expanding DocTR’s training data to include non-Latin scripts and diverse fonts across languages like Bahasa Indonesia, Mandarin, and Arabic will help SafeBite reach a broader global audience [1]. Second, the LLM API call accounts for most of the system’s latency, adding a mean of 3,241 ms to the total. Running a quantized model like Llama 3.1 8B [30] directly on the device would remove the need for an internet connection and reduce scan times to under two seconds, making the app highly practical even in fast-service dining environments.

ACKNOWLEDGMENT

The authors thank Bina Nusantara University and the Software Engineering course lecturers for guidance on software life-cycle practices, professional responsibility, teamwork, and technical communication. The authors also acknowledge the maintainers of the open-source frameworks and public datasets used by the project.

AUTHOR CONTRIBUTIONS

Muhammad Fatih Rizki Herianto, Pasya Kemal Halim and Ratu Azalea Meiza Asri contributed in requirements analysis, system design, implementation, verification, repository preparation and writing manuscript. Supervision, methodology, review, and validation are contributed by Abdul Haris Rangkuti. All authors read and approved the final manuscript.

OPEN DATA ACCESS

The SafeBite application code is available at <https://github.com/Oikpe/SafeBiteApp>.

REFERENCES

- [1] S. H. Sicherer and H. A. Sampson, "Food allergy: Epidemiology, pathogenesis, diagnosis, and treatment," *J. Allergy Clin. Immunol.*, vol. 133, no. 2, pp. 291–307, 2014.
- [2] R. S. Gupta, C. M. Warren, B. M. Smith, J. Jiang, J. A. Blumenstock, M. M. Davis, R. P. Schleimer, and K. C. Nadeau, "Prevalence and severity of food allergies among US adults," *JAMA Netw. Open*, vol. 2, no. 1, p. e185630, 2019.
- [3] S. A. Bock, A. Muñoz-Furlong, and H. A. Sampson, "Further fatalities caused by anaphylactic reactions to food, 2001–2006," *J. Allergy Clin. Immunol.*, vol. 119, no. 4, pp. 1016–1018, 2007.
- [4] A. J. Cummings, R. C. Knibb, R. M. King, and J. S. Lucas, "The psychosocial impact of food allergy and food hypersensitivity in children, adolescents and their families: A review," *Allergy*, vol. 65, no. 8, pp. 933–945, 2010.
- [5] V. Stankovich, E. N. C. Mills, and L. A. Gethings, "Allergen risk assessment in food service environments: Current challenges and technological interventions," *Food Chem. Toxicol.*, vol. 175, p. 113742, 2023.
- [6] J. Barnett, J. Leftwich, K. Muncer, K. Grimshaw, R. Shepherd, M. M. Raats, M. H. Gowland, and J. S. Lucas, "How do people with food allergies use written and verbal information provided to them in a restaurant?" *Appetite*, vol. 56, no. 1, pp. 179–185, 2011.
- [7] T. J. Radke, L. G. Brown, E. R. Hoover, B. V. Faw, D. Reimann, M. R. Wong, N. Hedeem, and H. Blade, "Food allergy knowledge and attitudes of restaurant managers and staff: An EHS-Net study," *J. Food Prot.*, vol. 79, no. 9, pp. 1588–1598, 2016.
- [8] European Parliament and Council of the European Union, "Regulation (EU) No 1169/2011 on the provision of food information to consumers," *Off. J. Eur. Union*, vol. L 304, pp. 18–63, 2011.
- [9] E. J. Topol, "High-performance medicine: The convergence of human and artificial intelligence," *Nat. Med.*, vol. 25, no. 1, pp. 44–56, 2019.
- [10] A. S. M. Mosa, I. Yoo, and L. Sheets, "A systematic review of healthcare applications for smartphones," *BMC Med. Inform. Decis. Mak.*, vol. 12, p. 67, 2012.
- [11] C. L. Ventola, "Mobile devices and apps for health care professionals: Uses and benefits," *Pharm. Ther.*, vol. 39, no. 5, pp. 356–364, 2014.
- [12] A. Charland and B. Leroux, "Mobile application development: Web vs. native," *Commun. ACM*, vol. 54, no. 5, pp. 49–53, 2011.
- [13] R. Smith, "An overview of the Tesseract OCR engine," in *Proc. 9th Int. Conf. Document Analysis and Recognition (ICDAR)*, Parana, Brazil, 2007, pp. 629–633.
- [14] Mindee, "docTR: Document Text Recognition" [Computer software], ver. 0.9, GitHub, 2021. [Online]. Available: <https://github.com/mindee/doctr>
- [15] S. Long, X. He, and C. Yao, "Scene text detection and recognition: The deep learning era," *Int. J. Comput. Vis.*, vol. 129, no. 1, pp. 161–184, 2021.
- [16] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 11, pp. 2298–2304, 2017.
- [17] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee, "Character region awareness for text detection," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, 2019, pp. 9365–9374.
- [18] C. Li, Y. Liu, X. Bai et al., "PP-OCRv3: More attempts towards better multilingual OCR system," *arXiv preprint arXiv:2206.03001*, 2022.
- [19] M. Li, T. Lv, J. Chen, L. Cui, Y. Lu, D. Florencio, C. Zhang, Z. Li, and F. Wei, "TrOCR: Transformer-based optical character recognition with pre-trained models," in *Proc. AAAI Conf. Artif. Intell.*, Washington, DC, USA, 2023, vol. 37, no. 11, pp. 13094–13102.
- [20] JaidedAI, "EasyOCR: Ready-to-use OCR with 80+ supported languages" [Computer software], GitHub, 2020. [Online]. Available: <https://github.com/JaidedAI/EasyOCR>
- [21] K. Nishiyama and M. Iwabuchi, "Optical character recognition for restaurant menu digitisation: A systematic evaluation," in *Proc. 16th Int. Conf. Document Analysis and Recognition (ICDAR)*, Lausanne, Switzerland, 2021, pp. 245–252.
- [22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, Long Beach, CA, USA, 2017, vol. 30, pp. 5998–6008.
- [23] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North American Chapter Association Computational Linguistics: Human Language Technologies (NAACL-HLT)*, Minneapolis, MN, USA, 2019, vol. 1, pp. 4171–4186.
- [24] T. Eftimov, P. Korošec, and B. K. Seljak, "A novel approach for named-entity recognition in recipe texts," in *Proc. First Workshop Language Technology Cultural Heritage, Social Sciences and Humanities*, Valencia, Spain, 2017, pp. 28–34.
- [25] G. Popovski, B. K. Seljak, and T. Eftimov, "A systematic approach for food categorisation and information extraction from unstructured data," *IEEE Access*, vol. 8, pp. 71421–71434, 2020.
- [26] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz et al., "Transformers: State-of-the-art natural language processing," in *Proc. Conf. Empirical Methods Natural Language Processing: System Demonstrations (EMNLP)*, Online, 2020, pp. 38–45.
- [27] J. Munene and S. H. Lee, "Automated allergen classification in food products using machine learning: A comparative analysis," *J. Food Sci. Technol.*, vol. 59, no. 8, pp. 3120–3131, 2022.
- [28] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell et al., "Language models are few-shot learners," in *Advances in Neural Information Processing Systems (NeurIPS)*, Virtual, 2020, vol. 33, pp. 1877–1901.
- [29] B. Hui et al., "Qwen2.5-Coder Technical Report," *arXiv preprint arXiv:2409.12186*, 2024.
- [30] A. Dubey et al., "The Llama 3 Herd of Models," *arXiv preprint arXiv:2407.21783*, 2024.
- [31] G. Bradski, "The OpenCV library," *Dr. Dobb's J. Softw. Tools*, vol. 25, no. 11, pp. 120–125, 2000.
- [32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 770–778.
- [33] J. Martin, *Rapid Application Development*. New York, NY: Macmillan, 1991.
- [34] S. McConnell, *Rapid Development: Taming Wild Software Schedules*. Redmond, WA: Microsoft Press, 1996.
- [35] S. Ramírez, "FastAPI: Modern, fast web framework for building APIs with Python," 2018. [Online]. Available: <https://fastapi.tiangolo.com>
- [36] Supabase, Inc., "Supabase: The open source Firebase alternative," 2020. [Online]. Available: <https://supabase.com>
- [37] Ionic Team, "Capacitor: Cross-platform native runtime for web apps," 2019. [Online]. Available: <https://capacitorjs.com>
- [38] Meta Platforms, Inc., "React: A JavaScript library for building user interfaces," 2013. [Online]. Available: <https://react.dev>
- [39] D. M. W. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation," *J. Mach. Learn. Technol.*, vol. 2, no. 1, pp. 37–63, 2011.