

Sugran: The Maharashtrian Recipe Generator — An AI-Powered Serverless Multimodal Web Application for Regional Culinary Preservation

Prachi Devidas Bhombe¹, Sujal Chandrajit Baviskar², Nikita Moreshwar Bharambe³,

Amit Sanjay Khatal⁴, Prof. Rajni Hemade⁵

^{1,2,3,4} B.Tech Students, Department of Computer Science and Engineering, Nutan College of Engineering and Research (NCER), Talegaon Dabhade, Pune – 410507, Maharashtra, India

⁵ Assistant Professor, Department of Computer Science and Engineering, Nutan College of Engineering and Research (NCER), Talegaon Dabhade, Pune – 410507, Maharashtra, India

Abstract— This paper presents Sugran, an innovative AI-powered multimodal web application designed to identify traditional Maharashtrian cuisine from user-provided images and dynamically generate highly localised, authentic, and region-specific recipes. The system architecture is built upon a fully decentralised, serverless frontend using React.js v18, ensuring 100% user data privacy by processing images locally as Base64-encoded strings. By leveraging the Google Gemini 1.5 Flash Vision API, Sugran overcomes the fundamental limitations of standard Convolutional Neural Networks (CNNs) that frequently misclassify the complex visual textures of regional Indian food. A core technical innovation is the implementation of an algorithmic JSON Forcing prompt engineering methodology, which constrains the AI to return structured JSON payloads containing up to five distinct regional recipe variations — Kolhapuri, Puneri, Malvani, Varhadi, and Nashik — for each identified dish. The application promotes technological inclusivity via a bilingual interface in English and Marathi (Devanagari script), supplemented by the browser-native Web Speech API for hands-free Text-to-Speech (TTS) guidance. A state-driven portion calculator scales ingredient quantities in real-time. Experimental evaluation demonstrates 94% identification accuracy for Maharashtrian dishes across 50 test images, outperforming the 84.2% achieved by dedicated InceptionV3 classifiers. A 100% parseable JSON response rate was achieved across all 30 formal trials.

Key Words: Multimodal AI, Large Multimodal Models (LMMs), Maharashtrian Culinary Preservation, Gemini 1.5 Flash, Serverless Architecture, React.js, Regional Recipe Generation, Bilingual Accessibility, Text-to-Speech, Dynamic Ingredient Scaling, Prompt Engineering, JSON Forcing, Cultural Heritage Digitisation.

1. INTRODUCTION

The rapid advancement of Artificial Intelligence (AI), specifically the evolution of Large Multimodal Models (LMMs), has fundamentally transformed how computer systems interpret, process, and reason about complex visual and textual data simultaneously. Traditional computer vision models were largely restricted to simple bounding-box object detection or single-label image classification. Today, multimodal architectures can seamlessly bridge the gap between pixel-level visual data and deep, contextual natural-language generation.

Sugran is an innovative, AI-powered web application designed to harness this cutting-edge technology to identify traditional Maharashtrian cuisine from user-provided images and dynamically generate highly detailed, authentic, and region-specific recipes. The word "Sugran" is derived from Marathi, traditionally referring to a skilled and artistically accomplished homemaker renowned for her extraordinary culinary expertise — a name that embodies the project's spirit of cultural reverence, culinary precision, and technological respect for indigenous knowledge systems.

Developed as a Final Year Engineering Project at Nutan College of Engineering and Research, Talegaon, Pune, Sugran serves as a vital technological bridge between modern artificial intelligence and the rich, centuries-old culinary heritage of Maharashtra. The application is built on a decentralised, serverless frontend architecture using React.js v18 for component-based state management and Virtual DOM rendering, and employs Tailwind CSS v3 for a fluid, mobile-first interface.

A. Necessity

The necessity for a dedicated Maharashtrian culinary AI application arises from the profound geographic,

climatic, and cultural diversity of the state — diversity that is vastly underrepresented in global digital media and culinary technology sectors. Maharashtrian cuisine is not a monolithic entity; its flavour profiles, cooking methodologies, and staple ingredients change drastically across distinct geographic zones. The Konkan coastal belt (Malvani cuisine) relies heavily on fresh coconut and kokum. The Desh region (Puneri cuisine) is known for milder vegetarian flavours using jaggery, tamarind, and Goda Masala. Kolhapur is internationally recognised for its fiery Tambda Rassa and Pandhra Rassa. Standard global recipe applications completely fail to capture this nuanced regional depth.

B. Problem Statement

This project directly addresses four critical, unresolved shortcomings in existing culinary platforms:

- **Algorithmic Bias and Cultural Homogenisation:** Most commercial platforms utilise generic CNNs trained on Western datasets like Food-101 [2], which contains 101,000 images across 101 predominantly Western food categories with no Maharashtrian representation.
- **Linguistic and Technological Accessibility Barriers:** Severe lack of native vernacular support and hands-free assistive technology in existing culinary software.
- **High Architectural Overhead and Data Privacy Risks:** Traditional recipe applications rely on centralised, resource-intensive backend servers, leading to high costs and latency.
- **Loss of Intangible Cultural Heritage:** The oral tradition of recipe-passing from generation to generation is rapidly eroding.

C. Objectives

The Sugran application is guided by the following strict technical and functional objectives: (1) High-Fidelity Multimodal Identification — develop an AI-driven interface capable of accepting user-uploaded images and accurately identifying Maharashtrian food items using the Google Gemini 1.5 Flash Vision API with target accuracy exceeding 90%; (2) Algorithmic Regional Generation — engineer complex meta-prompts that force the LMM to generate structured JSON data containing up to five distinct regional recipe variations; (3) Native Bilingual Accessibility — implement a state-driven bilingual UI allowing the entire application to toggle between English and Marathi without page reloads; (4) Dynamic Portion Scaling — construct a React-based portion calculator that recalculates ingredient matrices using $Q_{\text{final}} = B \times M$,

where $M = S_{\text{user}} / S_{\text{base}}$; and (5) 100% Serverless Deployment — build a decentralised, zero-cost frontend architecture.

2. LITERATURE REVIEW

A comprehensive review of existing literature was conducted to understand the state of the art in food image recognition, recipe generation, multilingual AI, serverless web architecture, and culturally-aware AI systems. The survey spans foundational work from 2014 to the most recent publications in 2025.

TABLE I: Literature Survey Summary

Sr.	Author (Year)	Methodology	Key Findings	Gap Addressed
1	Bossard et al. (2014)	CNN + Random Forests	Food-101; 88.1% top-5 accuracy	Western-centric; no Indian cuisine
2	Salvador et al. (2017)	Neural + Joint Embedding	Cross-modal image-to-recipe retrieval	No regional Indian recipes
3	Sharma et al. (2021)	ResNet50 Transfer Learning	88.5% accuracy on Indian food	No cultural context or multilingual output
4	Deshpande (2022)	Fine-tuned InceptionV3	84.2% on 18 Maharashtrian dishes	Classification only; no recipe generation
5	Patel et al. (2023)	MobileNet V3 Transfer Learning	91% accuracy; lightweight	Mobile only; no recipe generation
6	Lo et al. (2025)	Gemini Vision API	Accurate macro-nutrient estimation	No regional or bilingual support
7	Patel et al. (2025)	Prompt Engineering	High accuracy in Marathi generation	Validates Sugran bilingual approach

A. Early CNNs and Western Dataset Bias

Foundational food recognition research relied entirely on CNNs trained on datasets like Food-101 [2]. The dataset comprises 101,000 images across 101 food categories and was inherently Western-centric — featuring pizza, hamburgers, and French fries while entirely excluding Indian or South-East Asian cuisines. Models trained on such corpora inherit absolute geographical and cultural bias. A Maharashtrian Thalipeeth shares superficial visual similarities with a Western pancake but differs fundamentally in ingredients (jowar, bajra, besan), preparation technique, texture, and deep cultural significance.

B. Paradigm Shift to Large Multimodal Models

The advent of LMMs such as Google's Gemini family represents a fundamental paradigm shift from simple image labelling to deep contextual reasoning about visual content. Unlike pure vision classifiers, LMMs are pre-trained on trillions of interleaved image-text pairs at internet scale, allowing them to simultaneously reason about the cultural, historical, and semantic content of an image while generating natural language descriptions [3]. Sugran leverages these capabilities by bypassing the need for custom dataset training entirely.

C. Research Gaps Addressed by Sugran

The systematic review reveals three primary research gaps that Sugran simultaneously addresses: (1) Identification + Generation Gap — no existing system combines multimodal dish identification with culturally aware, region-specific recipe generation for Maharashtrian cuisine; (2) Bilingual Accessibility Gap — complete absence of bilingual (English-Marathi) culinary AI applications with native TTS support; (3) Privacy and Cost Gap — existing food AI applications universally rely on centralised backend architectures, creating unnecessary privacy risks.

3. SYSTEM ANALYSIS AND DESIGN

A. Functional Requirements

The system must satisfy the following key functional requirements:

- FR-01 Secure API Key Management: Client-side settings modal for users to input their personal Gemini API key, stored exclusively in the browser's localStorage.
- FR-02 Multimodal Input Handling: Drag-and-drop file upload zone and live camera capture via

MediaDevices.getUserMedia. Supported formats: JPEG, PNG, WEBP (max 10 MB).

- FR-03 Base64 Image Encoding: HTML5 FileReader API to convert uploaded images into Base64 strings entirely without server-side processing.
- FR-04 Algorithmic Prompt Compilation: Frontend logic dynamically compiles the JSON-schema Culinary Historian meta-prompt alongside the Base64 image string before executing the API POST request.
- FR-05 Dynamic UI Rendering: Parse AI-generated JSON and map data into React components including a tabbed interface for regional recipe variations: Kolhapuri, Puneri, Malvani, Varhadi, and Nashik.
- FR-06 Real-Time Portion Scaling: State-driven portion calculator that recalculates all ingredient quantities for serving counts from 1 to 20 without additional API calls.

B. Data Flow Diagrams

The Context Level (Level 0) DFD treats the core web application as a single encapsulated "Black Box" process. Two primary external entities exist: (1) The End User, who provides food images, the personal Gemini API key, and dynamic UI control triggers; and (2) The Google Gemini API, which receives an HTTPS POST payload containing a Base64-encoded image string combined with a system-generated meta-prompt, and returns a structured JSON object.

The Level 1 DFD reveals six granular internal subprocesses: (P1) Input Processing and Encoding Module; (P2) Prompt Compilation Engine; (P3) API Fetch and Routing Utility; (P4) Data Parsing and State Management; (P5) Dynamic Portion Calculator; and (P6) Accessibility and Localisation Module for Marathi rendering and TTS.

C. System Requirements

TABLE II: Hardware Requirements

Component	Minimum Requirement
Processor	Intel Core i3 8th Gen / AMD Ryzen 3 / ARM equivalent
Memory (RAM)	4 GB minimum; 8 GB recommended
Camera	Integrated webcam or smartphone camera (≥ 2 MP)
Display	Any display with minimum 320 px width

Network	Stable broadband (≥ 5 Mbps) or 4G/5G
Storage	Minimum 150 MB for browser cache and Vite assets

TABLE III: Software Requirements

Component	Specification
Operating System	Windows 10/11, macOS 12+, Linux, Android 10+, iOS 14+
Web Browser	Chrome v90+, Firefox v88+, Safari v14+, Edge v90+
Core Framework	React.js v18.0+ (Functional Components with Hooks)
Styling Engine	Tailwind CSS v3.0+ (utility-first responsive CSS)
Build Tool	Vite v5.0 (ES module HMR; $\sim 10\times$ faster than CRA)
AI API	Google Gemini Vision API (gemini-1.5-flash)
Deployment	Vercel (automated CI/CD via GitHub integration)

4. PROJECT IMPLEMENTATION

A. System Architecture

The system architecture of Sugran is strictly client-side and component-driven, organised into three primary logical tiers:

Tier 1 — Input and Security Tier consists of the FileDropzone component, the Webcam Modal component, and the localStorage controller. It securely captures and stores the user's Gemini API key in the browser's localStorage, never transmitting it to any server other than the authorised Google AI endpoint. Binary image data is processed into payload-ready Base64 format using the FileReader API — entirely within the user's browser.

Tier 2 — AI Orchestration Tier handles the critical asynchronous communication to Google's Gemini servers. The ApiOrchestrator.js module concatenates the Base64 image string with the hardcoded JSON Forcing prompt schema, injects the user's API key from localStorage into the request header, dispatches an HTTPS POST request to the Gemini 1.5 Flash endpoint, awaits the JSON response with a configurable 30-second

timeout, strips any markdown fences from the response, and passes the clean JSON string to the parser with comprehensive error catching.

Tier 3 — Rendering and State Management Tier parses the payload and maps all regional variations to a clickable tabbed navigation interface (RegionTabs.jsx), binds all ingredient quantities to the Portion Calculator's multiplier state variable, and manages the SpeechSynthesis queue for auditory instruction playback.

B. Methodology and Algorithms

Algorithm A: JSON Forcing Methodology. The JSON Forcing methodology is the core algorithmic contribution of the Sugran project. It relies on three interdependent prompt constraints applied simultaneously: (1) Persona Constraint — the AI is assigned the role of a Maharashtrian Culinary Historian and Expert Chef; (2) Format Constraint — output is restricted exclusively to a raw JSON object, no markdown fences or conversational preamble; (3) Schema Constraint — the prompt defines exact mandatory JSON keys including dish_name, marathi_name, history, is_upvas_friendly, nutrition_per_serving, and regional_variations. This three-constraint approach achieved a 100% parseable response rate across all valid food image inputs.

Algorithm B: Dynamic Portion Scaling. Let B represent the base ingredient quantity returned by the AI model, S_{base} be the base serving size, and S_{user} be the user-defined serving count (1 to 20). The scaling multiplier M is calculated as: $M = S_{user} / S_{base}$. The final scaled quantity for each ingredient: $Q_{final} = B \times M$. Any update to S_{user} via the UI counter triggers an immediate React state update, recalculating all ingredient quantities simultaneously using parseFloat arithmetic.

Algorithm C: Asynchronous Audio Chunking for TTS. To prevent browser memory crashes during long TTS tasks involving multi-step Marathi cooking instructions, Sugran implements array-based chunking via the Speech Synthesis API. AI-generated cooking steps are received as a pre-segmented array (one element per step). On user trigger, a SpeechSynthesisUtterance object is created for each step with the appropriate language code (mr-IN for Marathi, en-US for English). Each utterance is pushed to the browser's speech queue, with a callback on the onend event triggering the next utterance.

C. Component Architecture

TABLE IV: React Component Architecture

Component	Responsibility
App.jsx (Root)	Global state management; conditional rendering between input and output phases
ImageUploader.jsx	File drag-and-drop zone; FileReader API integration; format and size validation
CameraModal.jsx	MediaDevices.getUserMedia integration; live video preview; frame capture
ApiOrchestrator.js	JSON Forcing prompt compilation; HTTPS fetch; response parsing with error handling
RecipeDisplay.jsx	Dish identification card; cultural history; nutritional data; Upvas badge
RegionTabs.jsx	Tabbed navigation for 5 regional recipe variations; active tab state
PortionCalculator.jsx	Real-time ingredient scaling; serving counter UI; parseFloat arithmetic
AudioPlayer.jsx	SpeechSynthesis TTS queue; play/pause/stop controls; step-by-step chunking

D. Deployment Strategy

Sugran utilises the Jamstack paradigm for zero-cost serverless hosting. The automated deployment pipeline executes in three stages: (1) Build Phase — project compiled using Vite, performing dead code elimination, minifying JavaScript chunks, and optimising static assets into a /dist directory (average build time: 12 seconds); (2) Static Hosting via CDN — build artefacts distributed to Vercel's global CDN infrastructure across multiple geographic edge nodes, ensuring sub-100 ms Time-to-First-Byte (TTFB) for users in India; (3) Automated CI/CD — every push to the main branch on GitHub triggers Vercel's automated build and deployment cycle, maintaining zero-downtime deployments with rollback capability.

5. SOFTWARE TESTING

A. Testing Strategy

Software testing for Sugran was conducted comprehensively to ensure the application is robust, user-friendly, and entirely free of critical defects prior to final deployment. The testing strategy prioritised three primary concerns: API Reliability, Data Transformation Integrity, and UI Consistency. The testing methodology followed the industry-standard V-Model. Both Black Box testing (based on input/output specifications) and White Box testing (verifying internal logic, particularly the mathematical scaling formula and JSON schema parsing) were employed throughout.

Compatibility Testing was conducted on four distinct browsers (Chrome v120, Firefox v121, Edge v120, Safari v17) and three hardware platforms including Android and iOS devices. Edge Case and Fault Tolerance Testing deliberately provided the system with non-food images, extremely low-light images, images of non-Maharashtrian dishes, corrupted image files, and invalid API key strings.

B. Test Cases and Results

TABLE V: Selected Test Cases and Results

Test Description	Input	Expected Output	Result
Valid food image upload (Poha)	Clear JPEG of Poha	Dish identified; regional variants generated	PASS
Live camera capture (Misal Pav)	Live camera feed	Dish identified with recipe output	PASS
Non-food image input	Image of a book	Error: NOT_MAHARASHTRIAN_FOOD	PASS
Invalid API key string	Wrong API key	Error modal: Invalid API Key (401)	PASS
Language toggle to Marathi	Click Marathi button	All text renders in Devanagari script	PASS
Portion scaling to 8 servings	Adjust counter to 8	All quantities multiplied by 4×	PASS

	(base=2)		
TTS playback in Marathi	Click audio (Marathi mode)	Marathi instructions spoken	PASS
Upvas flag: fasting-friendly dish	Image of Sabudana Khichdi	Upvas-friendly badge displayed	PASS
Mobile responsive layout	320 px viewport	UI stacks vertically; no overflow	PASS
Network disconnection mid-request	Disable internet	Network error message displayed	PASS

C. Bug Resolution Summary

Seven primary defects were documented and resolved during the testing phase. Key resolutions include: BUG-01 (High Severity) — Memory Leak in Camera Modal resolved by calling `track.stop()` on all active `MediaStreamTrack` objects when the camera modal unmounts; BUG-02 (Medium Severity) — Floating-Point Truncation in Portion Calculator resolved by migrating from `parseInt()` to `parseFloat()` with `toFixed(2)` formatting; BUG-03 (Low Severity) — Devanagari Text Overflow resolved via `overflow-wrap: break-word` CSS; BUG-04 (Medium Severity) — API Response Markdown Contamination resolved by a pre-parsing regex strip: `text.replace(/``json```g, "").trim()` before `JSON.parse()`.

6. RESULTS AND DISCUSSION

A. Results

The primary objective of Sugran — successful deployment of a serverless, multimodal application for Maharashtrian cuisine preservation — has been comprehensively achieved. The Google Gemini 1.5 Flash API processed visual cues and accurately identified dishes, returning a structured JSON payload within a mean API round-trip of 5,102 ms. Upon activating the Marathi language toggle, all text content was re-rendered in authentic Devanagari script without any page reload.

The tabbed navigation interface successfully displayed up to five distinct regional variations for each dish.

TABLE VI: Dish Identification Accuracy by Category

Food Category	Images Tested	Correct ID	Accuracy (%)
Breakfast Dishes	8	8	100.0%
Snacks / Street Food	8	8	100.0%
Main Course Non-Vegetarian	8	8	100.0%
Sweets and Desserts	6	5	83.3%
Main Course Vegetarian	8	7	87.5%
Breads (Bhakri, Thalipeeth)	6	5	83.3%
OVERALL	50	47	94.0%

TABLE VII: Stage-wise Latency Breakdown (Wi-Fi 50 Mbps)

Pipeline Stage	Mean (ms)	P95 (ms)	Share (%)
Base64 Encoding	187	268	3.4%
Prompt Compilation	12	18	0.2%
Gemini API Round-Trip	5,102	6,198	93.1%
JSON Parse + React Render	148	201	2.7%
CSS Animation + TTS Init	~29	~53	0.6%
Total End-to-End	5,449	6,685	100%

B. Discussion

The 94% identification accuracy achieved entirely without any custom-trained CNN model or traditional backend infrastructure surpasses the 84.2% achieved by Deshpande (2022) using a dedicated InceptionV3 model trained specifically on Maharashtrian dishes, and the 88.5% achieved by Sharma et al. (2021) using ResNet50 on generic Indian food. This demonstrates that prompt-engineered LMMs can outperform task-specific trained classifiers in specialised cultural domains without any labelled training data collection.

The three misclassifications (6%) were concentrated in categories where two dishes share similar visual characteristics — cream-coloured dairy desserts and round fried/steamed dumplings. The 100% parseable

JSON response rate across all 30 formal trials validates the JSON Forcing methodology as a reliable, reproducible technique for structured data extraction from LMMs.

Observed limitations include: (1) single point of failure on internet connectivity and valid Gemini API key; (2) single-dish analysis when multiple dishes appear in one frame; (3) AI-estimated nutritional values unsuitable for clinical dietary management; (4) significant API key onboarding friction for elderly and non-technical users; and (5) variable mr-IN Marathi TTS voice quality across browser vendors.

7. CONCLUSION AND FUTURE SCOPE

A. Conclusion

The successful implementation of Sugran validates the transformative potential of Large Multimodal Models when applied to hyper-local cultural preservation tasks. Three primary engineering conclusions have emerged:

(1) Superiority of Client-Side Privacy Architecture: By adopting a decentralised, serverless architecture, Sugran proves that AI applications can be simultaneously powerful and privacy-preserving, establishing a replicable standard for privacy-first AI application design.

(2) Prompt Engineering as a Reliable Algorithmic Discipline: Strict schema enforcement through persona-based meta-prompts can reliably transform conversational AI into a precise, deterministic data-generation engine. The JSON Forcing methodology achieved 100% parse success and is highly generalisable to other cultural documentation tasks.

(3) Democratisation of Specialised AI: Sugran conclusively demonstrates that undergraduate engineering teams working with limited resources and no proprietary datasets can leverage general-purpose LMMs to outperform commercial CNN-based alternatives by significant accuracy margins (94% vs. 84.2%) in specialised cultural domains.

B. Future Scope

- National Scalability and Multi-Cuisine Expansion: The component-driven architecture and prompt schema are cuisine-agnostic. Future iterations can scale to support the regional cuisines of all 28 Indian states with no structural code changes required.

- Edge Computing and Offline Identification: Future versions could integrate lightweight, on-device food recognition models using TensorFlow.js or ONNX Runtime Web (quantised MobileNetV3) for basic dish identification without internet connectivity.
- Advanced Multi-Dish Computer Vision: Integrating YOLOv8 object detection pre-processing would allow the system to identify and crop multiple individual food items within a single photograph, enabling comprehensive analysis of a full Maharashtrian Thali.
- Augmented Reality (AR) Cooking Assistant: Utilising the WebXR Device API to overlay step-by-step cooking instructions directly onto the user's kitchen workspace via smartphone cameras.
- Community-Driven Cultural Archive Platform: Expanding Sugran into a participatory, crowd-sourced platform where home cooks and regional culinary experts can contribute and validate their own family recipes.

ACKNOWLEDGEMENT

The authors express heartfelt gratitude to the Director, Dr. Pramod Patil, and the Head of Department, Dr. Mahesh Wankhade, of the Department of Computer Science and Engineering, Nutan College of Engineering and Research (NCER), Talegaon, Pune, for their unwavering institutional support. We sincerely thank our project guide, Prof. Rajani Hemade, for her patience, technical mentorship, and dedicated guidance throughout every phase of this project. We also thank Prof. S. B. Mehta, Project Co-ordinator, for organising all project phases and ensuring proper academic direction throughout the year.

REFERENCES

- [1] Bhombe, P., Bharambe, N., Baviskar, S., Khatal, A., & Hemade, R. (2025). A Comprehensive Survey on AI-Based Maharashtrian Recipe Image Recognition and Regional Culinary Personalisation Using Multimodal Deep Learning. Dept. of CSE, NCER, Pune (Under Review).
- [2] Bossard, L., Guillaumin, M., & Van Gool, L. (2014). Food-101 - Mining Discriminative

- Components with Random Forests. Proceedings of ECCV, Zurich, 446-461.
- [3] Chen, J., Zhu, D., Li, X., Wang, L., & Liu, Y. (2023). A Comprehensive Survey of Large Multimodal Models: Architecture, Training Paradigms, and Applications. arXiv preprint arXiv:2306.13549.
- [4] Deshpande, M. (2022). Deep Learning for Maharashtrian Cuisine Recognition Using InceptionV3 Architecture. Journal of Emerging Technologies in Engineering Applications, 5(2), 34-41.
- [5] Gomez, L., & Perez, R. (2022). Accessibility Benefits of the Web Speech API in Culinary Mobile Applications. Proceedings of ACM CHI, New Orleans, LA, 1-12.
- [6] Google AI for Developers. (2024). Gemini 1.5 Flash Model Documentation and API Reference. Google LLC. <https://ai.google.dev/docs>
- [7] Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., & Iwasawa, Y. (2022). Large Language Models are Zero-Shot Reasoners. NeurIPS, 35, 22199-22213.
- [8] Lo, F., Wang, X., & Huang, M. (2025). Nutritional Analysis from Food Images Using Large Multimodal Models. Proceedings of ACM Multimedia.
- [9] Patel, A., Joshi, R., & Kulkarni, S. (2025). Multilingual LLMs for Indic Languages: Marathi Benchmark Evaluation. International Journal of NLP Research, 12(1), 45-62.
- [10] Patel, J., Shah, V., & Mehta, N. (2023). MobileNetV3-Based Lightweight Indian Food Recognition. Proceedings of MISIP, 441-449.
- [11] React Community (Meta Open Source). (2024). React.js Documentation: Hooks, State Management, and Functional Components. <https://react.dev>
- [12] Salvador, A., et al. (2017). Learning Cross-Modal Embeddings for Cooking Recipes and Food Images. Proceedings of IEEE CVPR, Honolulu, HI, 3020-3028.
- [13] Sharma, S., Kumar, P., & Singh, A. (2021). ResNet50-Based Transfer Learning for Indian Food Classification. IJERA, 10(3), 112-119.
- [14] Smith, T. (2023). Serverless Web Architectures with React: Security, Privacy, and Cost Analysis. IEEE Internet Computing, 27(4), 55-62.
- [15] Tailwind Labs. (2024). Tailwind CSS Documentation. <https://tailwindcss.com/docs>
- [16] ViteJS Organization. (2024). Vite v5.0: Next Generation Frontend Tooling. <https://vitejs.dev>
- [17] W3C. (2023). Web Speech API Specification. W3C Community Group. <https://www.w3.org/TR/speech-api/>
- [18] Wei, J., et al. (2022). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. NeurIPS, 35, 24824-24837.