ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VII, July 2026

Smart Food Recommendation System Based on Age, Mood and Weather

Ch. Sravanthi Sowdanya[1] , Anisetty Gnana Sai[2] , Patnana Pavan Sai[3] , Korada Jithin Sai Kamal[4] , Batchu Mona Sahasra[5]

[1]Assistant Professor Dept. of CSE (AI & ML) ANITS Visakhapatnam, India

[2,3,4,5]Student Dept. of CSE (AI & ML) ANITS Visakhapatnam, India

DOI: https://doi.org/10.51583/IJLTEMAS.2026.150700096

Received: 02 August 2026; Accepted: 07August 2026; Published: 15 August 2026

ABSTRACT

Food recommendation systems share a common dependency: they require an interaction history before they can offer anything useful. First-time users receive no useful recommendations, while recommendations for returning users rely on the previous week's orders as an approximation of current preferences. Both cases lack important contextual information: weather, mood, and time of ordering are rarely considered, while age and hunger are almost never incorporated.

This paper introduces Smart Dine Pro, a system designed to avoid dependence on interaction history. At each request, the system captures the user's facial expression using DeepFace, retrieves live weather data from the OpenWeather API, and combines these with the user's age, hunger level, and stated food preference to score menu items using a weighted algorithm. Users who prefer not to use the webcam can instead submit their mood through a short questionnaire, and the same scoring engine processes both input routes.

We evaluated the system across 140 input combinations, spanning seven emotional states, five weather conditions, and four age groups. Contextually aligned suggestions were produced in 95.7% of cases, with zero age-restriction violations. A user study involving 30 participants reported an overall satisfaction score of 3.9 out of 5 and a recommendation acceptance rate of 73.3%.

Index Terms—Food Recommendation, Emotion Recognition, Deepface, Cold-Start Problem, Affective Computing, Context-Aware Computing, Weighted Scoring Algorithm, Flask, React**.Js**

INTRODUCTION

When asked what they want to eat, most people give an answer that has little to do with what they ordered last week. Rain changes it, as does stress. So do hunger level, age, and time of day. Food preference is shaped by contextual factors that a ratings history cannot capture, yet nearly all recommendation systems treat historical order data as their primary—often their only—input.The resulting effect is decision fatigue. Food delivery apps now present hundreds of options, which is associated with longer decision times, lower satisfaction, and higher abandon-ment rates [2]. A recommendation engine is intended to reduce this burden. Collaborative filtering, content-based filtering, and integrations of the two reduce the choice space moderately well [1], [2], but they still share one hard requirement: a past history of behavior. Without it, a new user receives no useful suggestions. This is the cold-start problem, which remains unsolved in production food systems more than a decade later [2].

The gap does not extend only to new users. Even for users with extensive histories, most systems remain entirely unaware of the present moment. A customer who ordered biryani on a hot Tuesday afternoon may want soup when it is raining. Comfort foods appeal to anxious users [5]. Nutritional requirements vary across age groups in ways that go beyond personal taste—they constitute dietary constraints [4]. All of this data is available the moment a user opens the menu. Almost no deployed system makes use of it.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VII, July 2026

Smart Dine Pro addresses this gap. The system detects emotional state using DeepFace, retrieves live weather from the OpenWeather API, and combines these signals with the user's age, hunger level, and dietary preference to rank menu items that have no interaction history. A dual input mode supports users who prefer not to use the camera— they enter their mood through a questionnaire and receive recommendations of equal quality. The Agri-Connect module also provides a small incentive for ordering locally sourced items and maintains a per-order donation ledger.

The contributions of this work are:

- An emotion-, weather-, and age-driven context-fusion scoring engine that requires no prior user history
- Dual emotion capture: automatic facial recognition via DeepFace, with a fallback questionnaire that feeds the same scoring engine
- Live rating updates: post-order feedback updates item ratings through a running average
- An Agri-Connect social impact module integrated directly into the scoring function, not applied as a post-processing step
- An explicit weighted multi-factor scoring algorithm
- A deployed React.js and Flask application with an admin dashboard
- A proposed ML improvement roadmap for Smart Dine Pro v2

LITERATURE REVIEW

Food recommendation has received significantly increased research attention over the past decade. Kumar et al. [1] compared these methods and found that hybrid approaches typically outperform either individual strategy, since each compensates for the other's blind spots. Gunawardena's review of all three system types [2] concluded that cold-start handling and real-time contextual adaptation remain the two most consistently unaddressed limitations. The work most closely related to the present system is Melese [3], which incorporated mood and weather into a hybrid filtering model but required users to manually input their emotional state and the weather, introducing friction and self-reporting bias. None of the existing systems capture ambient context during ordering time, and none offer a cold-start solution without prior interaction. Table I maps the gap in each system to the corresponding design decision in Smart Dine Pro.

SYSTEM OVERVIEW

Smart Dine Pro is structured in three layers — presentation, application, and data — that communicate through a REST API. Fig. 1 shows the architecture of the whole system. Three candidate designs were considered during development; the hybrid approach was chosen for its ability to capture real-time context, cold-start handling, age-appropriate filtering, dynamic rating updates, and social impact tracking. What differentiates this system from a regular recommendation backend lies in the application layer: no item is scored until the system retrieves the user's emotional state, current weather, and applies age-group dietary constraints as hard exclusions.

Presentation Layer

It is a single-page React.js application built around two design objectives: minimizing the number of steps between opening the application and receiving a recommendation, and providing clear feedback at every step. The home page offers users Camera Mode — automatic emotion detection via webcam — and Questionnaire Mode, a mood dropdown for users who prefer not to use the camera. Both paths feed into the same recommendation engine. The landing screen also displays food-card information (name, type, cost, calories, and rating), a slide-out cart panel, an ever-present navbar with a live cartcount, and an administrator-only dashboard displaying order queues, social impact data, and menu administration controls.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VII, July 2026

Application Layer

When a request is made to the Flask backend, it executes four modules:

- **Emotion Detection:** Accepts a webcam frame in base64 format, turns it into grayscale, downsizes it, restores the RGB channels, then sends it to DeepFace to be categorized into one of seven moods.
- **Weather Retrieval:** Makes requests to the OpenWeather IP-based geolocation API. If no answer is obtained in two seconds, the time-of-day fallback is automatic.
- **Preference Filtering:** Diet and age-group restrictions are applied before scoring; items outside the eligible set are removed entirely — no penalty, simply dropped.
- **Recommendation Engine:** Performs the weighted scor-ing function (Section IV) and yields six items through a two-step diversity sampling pass.

Active orders are stored in a Python dictionary. A before_request hook purges records older than 24 hours, keeping memory usage bounded without requiring a separate background scheduling process. Charity and order counts are summed and written to a dedicated JSON file that persists across reboots and is not included in the daily cleanup.

Data Layer

The menu is stored as a flat CSV file containing 77 annotated records — 40 vegetarian and 37 non-vegetarian— divided across 7 food categories. Each record contains the following fields: id, name, type, category, mood_tag, weather_tag, age_group, spice_level, calories, description, rating, and price. After each order, the accumulated rating is recalculated using a running average and written back immediately. impact_stats.json is a second JSON file with permanent charity and order counters outside the 24-hour cleanup cycle. This flat-file design avoids the need for database infrastructure while remaining mappable to a relational schema if scaling becomes necessary.

System Interaction Flow

All recommendation requests follow the sequence below:

(1) the user can select either Questionnaire Mode or Camera Mode; (2) the mood is read or a webcam frame is taken by the frontend; (3) a POST request is sent to /api/ai/analyze or /api/ai/manual; (4) Camera Mode only: the backend preprocesses the picture and invokes DeepFace; (5) the server fetches up-to-date weather; (6) the scoring engine filters, scores, and samples 6 items; (7) ranked results are returned as JSON and render as food cards.

Algorithm: Weighted Multi-Factor Scoring Function

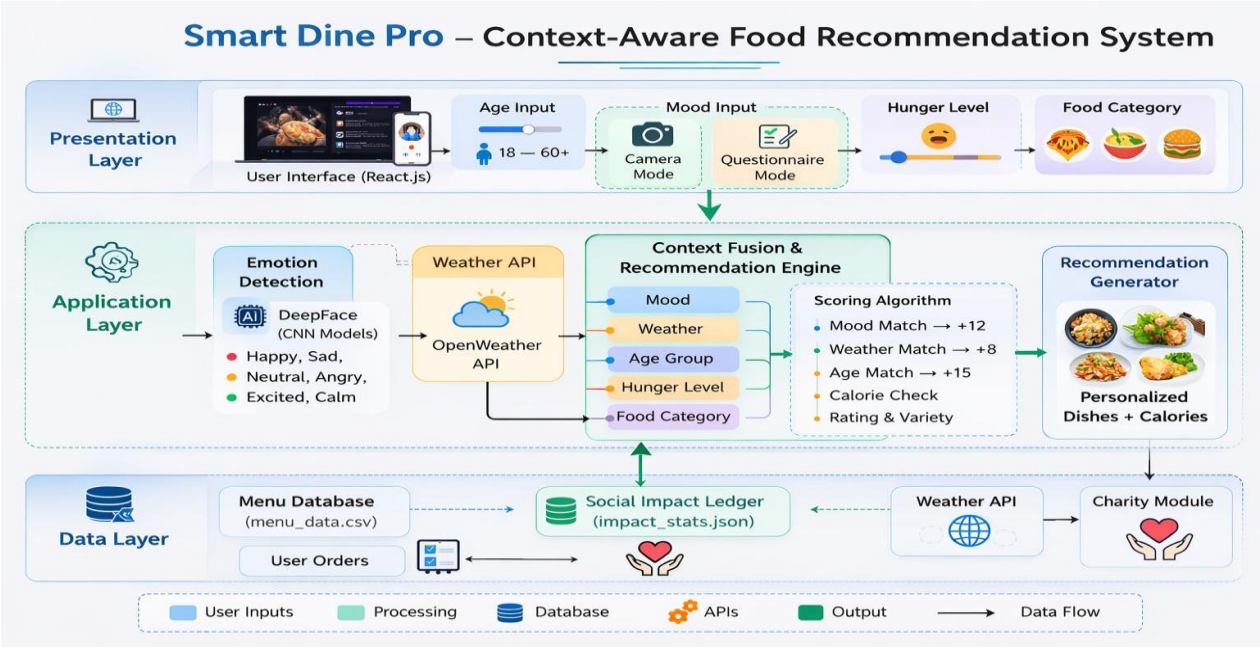
The recommendation engine rates all qualified menu items in a five-dimensional context vector space using a fixed weighted function. No collaborative filtering, matrix factorization, or

TABLE I Comparison of Related Work and How Smart Dine Pro Addresses Their Limitations

Ref.	Methodology	Objectives	Limitations	How Smart Dine Pro Addresses the Gap
[1]	Collaborative filtering (collab-orative + content-based)	Enhance accuracy of food and restaurant recommendations	Based on past ratings; no con-textual signals used	Emotion and weather are captured fresh on demand, dropping the dependency on order history

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VII, July 2026

[2]	Survey of recommendation techniques	Determine obstacles through system types	Cold-start and dynamic adaptation identified as gaps but not solved	Contextual parameters and emotion detection replace the need for prior rating data
[3]	Digital menu recommendation interface	Enhance customer interaction and personalization	UI-focused recommendation logic; no behavioral context	Backend scoring combines emotion detection with dietary and environmental signals for deeper personalization
[4]	Health-aware recommender models	Promote healthy nutrition using nutritional guidance	Nutrition-only scope; emotional and weather factors absent	Age-group dietary constraints applied as hard filters alongside mood and weather signals
[5]	Behavior and social-data-based recommendation	Personalize meal plan using behavioral data	No real-time emotional detection mechanism	DeepFace classifies facial expression of the user on each request
[6]	Sentiment analysis on user reviews	Enhance relevance by opinion mining	Retrospective; cannot respond to current user state	Context vector is constructed at request time; no review history required
[7]	Augmented reality restaurant interaction	Enhance engagement with immersive interfaces	Recommendation logic remains bound to stored preferences	Live emotion, weather and user parameters are blended on a request basis



Dietary Filter:

$$\mathbf{M}' = \{m \in \mathbf{M} \mid m.\text{type} \in \text{eligible}(d)\} \quad (2)$$

Age-Group Filter:

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VII, July 2026

$$\mathbf{M}' = \{m \in \mathbf{M}' \mid m.\text{age\_group} \in \text{agePool}(a)\} \quad (3)$$

{Child, All} $a < 13$

Fig. 1. Comprehensive System Architecture of Smart Dine Pro: three-layer structure with inter-module communication flow.

trained model is involved. This was a deliberate design choice: with only 77 menu items and 30 pilot participants, the dataset was too small to train a reliable classifier. The deterministic approach also makes it possible to trace every recommendation

— any output can be reconstructed from its inputs.

A. Input Vector

Every request forms a context vector **C**:

$$\mathbf{C} = \langle e, w, a, h, d \rangle \quad (1)$$

$$\text{agePool}(a) = \begin{cases} \{\text{Teen, All}\} & 13 \le a < 18 \\ \{\text{Adult, All}\} & 18 \le a \le 60 \\ \{\text{Senior, All}\} & a > 60 \end{cases} \quad (4)$$

Items that are not in the qualifying set are not penalized or down-weighted. The distinction matters: a penalized item can still be seen where all the contextually appropriate alternatives are scored lower. The hard exclusion removes such items from consideration entirely, so they cannot appear in the results at all.

C. Scoring Function

Once the qualified set $\mathbf{M}'$ has been established, each remaining item is scored along five weighted dimensions.

The composite score of each $m \in \mathbf{M}'$ is:

$$S(m, \mathbf{C}) = \alpha_1 f_e + \alpha_2 f_w + \alpha_3 f_a + \alpha_4 f_h + \alpha_5 f_g + r(m) \quad (5)$$

Mood Match ($\alpha_1 = 12$):

Mood Match rewards a menu item when the detected or self-reported emotion overlaps with the item's tagged moods, contributing 12 points to the composite score.

$$f_e = \begin{cases} 1 & e \in m.\text{mood\_tag} \\ 0 & \text{otherwise} \end{cases}$$

where e is the perceived emotion; w is the weather condition string; $a \in \mathbb{Z}^+$ is age; $h \in$ {light, moderate, heavy, starving} is hunger level; and $d \in$ {veg, non-veg, both} is dietary preference.

Weather Match ($\alpha_2 = 8$):(6)

*B. Hard Filtering*Weather Match rewards a menu item when the current

weather condition matches one of the item's tagged weather conditions, contributing 8 points to the composite score.

The full menu M has two restrictions before scoring to create a qualified set $\mathbf{M}'$:

$$f_w = \begin{cases} 1 & w \in m.\text{weather\_tag} \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VII, July 2026

Age-Group Match (α_3 = 15):

Age-Group Match rewards a menu item when its target age group matches the user's age category, contributing 15 points to the composite score._ ☾

Algorithm 1 Smart Dine Pro Weighted Context Scoring

Require: $\mathbf{C} = \langle e, w, a, h, d \rangle$, menu $\mathbf{M}$ **Ensure:** Recommendation set R (6 items) 1: $\mathbf{M}' \leftarrow$ DietaryFilter(M, d)

2: $\mathrm{M}' \leftarrow$ AgeFilter(M', a)

3: **for** each $m \in \mathbf{M}'$ **do**

$f\ a = m$.age_group = target(a)

0 otherwise

(8)

4: $S \leftarrow r(m)$

5: **if** m.age_group = target(a) **then** S += 15

6: **end if**

Hunger-Calorie Match (α_4 = 10):

Hunger-Calorie Match rewards items whose calorie content fits the user's stated hunger level, contributing 10 points to the composite score.

1 h = light, m.cal < 3007: **if** $e \in m$.mood_tag **then** S += 12

8: **end if**

9: **if** (h = light $\wedge$ m.cal < 300) $\vee$ ($h \in$ {heavy, starving} $\wedge$

m.cal > 500) **then** S += 10

10: **end if**

11: **if** $w \in m$.weather_tag **then** S += 8

12: **end if**

13: **if** "Locally Sourced" $\subseteq m$.description **then** S += 2

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VII, July 2026

$f\ h = h \in$ {heavy, starving}, m.cal > 500

0 otherwise(9)14: **end if**

15: score[m] ← S

16: **end for**

17: T12 ← TopK(score, k=12)

Agri-Connect Boost (α_5 = 2):

The Agri-Connect Boost adds a small tie-breaking bonus when a dish's description indicates it is locally sourced.

Component	Technology / Version
Frontend	React 18 + Vite 4
Backend	Flask 2.x + Flask-CORS
Emotion Detection	DeepFace 0.0.80
Image Processing	OpenCV (cv2) 4.x
Data Processing	pandas, NumPy
Weather Integration	OpenWeather API v1
Data Storage	CSV + JSON (flat-file)
Iconography	Lucide React 0.383.0
Runtime	Python 3.x, Node.js

1 "Locally Sourced" ⊆ m.description18: R ← RandomSample(T12, k=6)

return R

TABLE IITECHNOLOGY STACK$f\ g = 0$ otherwise(10)**Dynamic Rating** $r(m) \in [3.8, 4.9]$, updated after each order:

$$r'(m) = \text{round}\ \frac{r(m)+r\ \text{new}}{,\ 1}\ -\qquad (11)$$

D. Weight Rationale

The priority ordering — α_3=15 > α_1=12 > α_4=10 > α_2=8 > α_5=2 — was not set arbitrarily. Each weight reflects a judgment about which type of mismatch would be most harmful. The age-group suitability weight is the greatest since it is unreasonable to prescribe nutritionally unsuitable food to a Child or Senior — it is not only a mismatch of preferences but a functional error [4]. Mood carries the second-highest weight, as it is the primary indicator of personalization and is captured automatically without requiring an additional task from the user [5]. Hunger reflects an immediate caloric need that is more decision-relevant than ambient weather. Weather, by contrast, is a matter of comfort and situational fit, and is a weaker predictor of what one ought to consume than the prevailing hunger condition. The Agri-Connect boost at α_5=2 is very small on purpose — it tips the balance between otherwise identical items without overriding a contextually better match. During development, we tested a flat-weight variant in which all five signals were scored equally. Average performance on the test set was reasonable, but nutritionally unsuitable items appeared in the Child and Senior profiles in edge cases. That problem was overcome by the weighted ordering.

E.Formal Pseudocode

The complete procedure is given in Algorithm 1.

IMPLEMENTATION

Smart Dine Pro can run on any Python 3.x compatible computer with a webcam and internet access. No cloud

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VII, July 2026

infrastructure is required.

Frontend and Backend

The frontend is built with React 18 and Vite 4. We chose Vite over Create React App because build times during development were much shorter, and the production bundles came out smaller

— significant for a system intended to run on modest hardware. The backend is developed with Flask 2.x and Flask-CORS for cross-origin requests between the Vite development server (port 5173) and Flask (port 5000); in production both are served from the same origin.

Table II lists the full technology stack.

A. Emotion Detection

Before the captured frame is passed to DeepFace, three preprocessing stages are carried out: grayscale conversion using cv2.COLOR_BGR2GRAY to minimize noise from un-even lighting; downsample to 640×480 to equalize input dimensions; and conversion back to 3-channel RGB via cv2.COLOR_GRAY2RGB, which is the format DeepFace expects. DeepFace is run with actions=['emotion'],

TABLE III CONTEXTUAL COVERAGE RESULTS (140 TEST CASES)

Alignment Category	**Count**	**%**
Fully Aligned (3/3)	81	57.9%
Partially Aligned (2/3)	53	37.9%
Misaligned (<2)	6	4.3%
Total	**140**	**100.0%**

Note: Age hard-constraint violations (of 140 runs): 0 (0.0%).

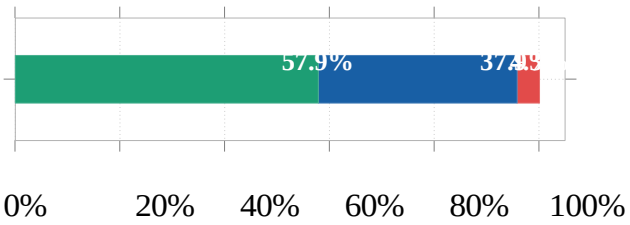


Fig. 2. Contextual alignment across 140 test combinations. Combined aligned rate: **95.7%**. Age hard-constraint violations: zero.

Evaluation

METHODOLOGY

Three methods of assessment were involved: systematic testing of every combination of inputs to test scoring accuracy; end-to-end latency measurement in repeated trial mode; and a pilot user study of acceptance rate and satisfaction.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VII, July 2026

Contextual Coverage Testing

The dataset includes 7 emotion conditions and 4 age groups. Of the full weather taxonomy, the 5 most common conditions were selected for combinatorial testing, yielding $7 \times 5 \times 4 = 140$ test combinations. Each output was compared with three criteria: mood correspondence, weather congruence and age hard-constraint satisfaction. Table III shows the counts; Fig. 2 shows the distribution graphically.

A combined alignment rate of 95.7% was achieved across the whole input space. The 6 non-congruent cases were all based in the same small corner: the *Disgust* and *Fear* emotional states combined with *Snow* or *Thunderstorm* weather in the Child age bracket, where the dataset has very few tagged items. In such instances the engine returned the highest-rated eligible items instead of generating an error. This is a data gap, not an algorithm failure. No age-constraint violations occurred

(Table III).

Scoring Range Analysis

Table **??** indicates the variation in composite scores in four scenarios. A completely unmatched item achieves its raw rating (3.8–4.9) only, and a full match item with all the bonuses applied scores 46.8–51.8. The gap is so broad that a highly-rated but context-mismatched item cannot outperform a lower-rated context-matched one. The ranking is dominated by contextual signals.

The latency of Camera Mode is nearly entirely DeepFace running on CPU. The average of 1,820 ms is within the range of acceptability in the case of the restaurant ordering scenario butTABLE IV

SYSTEM LATENCY (30 TRIALS PER MODE, MEAN VALUES)

Mode	**DeepFace**	**Weather API**	**Total**
Camera Mode	1,140 ms	480 ms	1,820 ms
Questionnaire Fallback	N/A	480 ms	510 ms
Active	N/A	N/A	<95 ms

Note: Total latency includes additional system overhead beyond the measured DeepFace and Weather API components. (Please verify this matches the actual implementation before final submission.)

TABLE V USER STUDY SATISFACTION RESULTS (N = 30, 5-PT LIKERT)

Dimension	**Mean**	**SD**
Recommendation Relevance	3.8	0.74
Interface Usability	4.2	0.61
System Speed	3.6	0.82
Overall Satisfaction	3.9	0.68

GPU acceleration would bring it below one second and is the most significant v2 infrastructure enhancement. Questionnaire Mode at 510 ms is quick enough to accommodate any ordering scenario.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VII, July 2026

User Study

All 30 participants between age 18 and 58 (mean 27.4, SD 9.2) used Camera Mode, checked the six proposed items, placed a simulated order, and responded to a five-point Likert questionnaire. Table V reports the scores.

Of the 30 respondents, 22 selected at least one item from the recommended set, giving a recommendation acceptance rate of **22/30 = 73.3%** (defined as the proportion of participants who selected at least one recommended item). The highest-rated dimension was Interface Usability at 4.2/5. System Speed received the lowest rating (3.6/5), consistent with the 1,820 ms Camera Mode latency, suggesting participants perceived the delay. The 8 declined participants all mentioned dietary specificity: they wanted sub-category choices like South Indian cuisine, rather than the vegetarian/non-vegetarian dichotomy that the current system offers. No one said they were given an age-inappropriate recommendation.

DISCUSSION

The 95.7% alignment rate indicates that the scoring function produced contextually relevant output across the tested combi-nation space. The discrepancy between matched and unmatched items is broad enough that a highly rated item that does not match the context cannot displace a partially matched one

— that was the intended behavior and it succeeded during testing. The Agri-Connect module performed as expected: with α_5=2, locally produced items were given a tie-breaking advantage without interfering with results where a stronger contextual signal was present. Live rating updates were reflected in recommendations immediately during each test session, confirming that the write-back mechanism functioned correctly. The misalignment of 4.3% can entirely be explained by the scarcity of data coverage for specific emotion-weather combinations, not by any flaw in the scoring logic. The highest-priority improvement is expanding the dataset. The 73.3%

acceptance rate is a fair starting point; the most obvious path to improvement, based on participant feedback, is sub-category dietary filtering.

Limitations

Five limitations are outlined below.

First, the user study (N = 30) was done under controlled lighting on a single test device. The conditions of reality

— changing light, incomplete face masking, and a range of webcam distances — were not applied and outcomes can vary in such environments.

Second, the flat-file storage layer is okay at the present scale but would need migration to a relational database before the system could support relevant simultaneous load.

Third, IP-based geolocation gives erroneous results for users on VPNs and on some mobile networks.

Fourth, DeepFace was not validated for demographic fairness in this study; emotion-recognition accuracy is known in the literature to vary across skin tone, lighting conditions, and age groups, and this system inherits that risk.

Fifth, the age-based hard constraint relies on a single self-reported value with no verification mechanism; borderline or misreported ages could result in inappropriate inclusion or exclusion of menu items, a fairness concern that future versions should address.

CONCLUSION

Smart Dine Pro was developed to address a specific problem: food recommendation systems typically rely on

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VII, July 2026

interaction histories and therefore cannot assist first-time users. We addressed this by replacing interaction history with three context signals that are taken at the time of ordering — emotion recognized on a webcam frame using DeepFace, live weather provided by the OpenWeather API and age announced by the user — together with the level of hunger and choice of diet through a weighted scoring method. The system generated contextually aligned recommendations in 95.7% of cases across 140 test combinations, with zero age-constraint violations. A 30-subject user study reported 73.3% acceptance and 3.9/5 general satisfaction. The choice of a deterministic scoring algorithm over a trained classifier was driven by data volume constraints: a 77-item menu and 30 participants would have made a trained model unreliable. That is a tradeoff that is explicitly recognized

— a trained model is possible after sufficient interaction information is gathered, and the scoring algorithm would be retained as a cold-start fallback during that transition. Smart Dine Pro v2 defines four explicit targets, informed directly by user study feedback and evaluation findings: dataset expansion to cover low-frequency emotion-weather combinations; GPU acceleration to lower Camera Mode response to less than one second; browser-based geolocation to replace the IP-based lookup; and sub-category dietary filtering to address the most common reason participants declined recommendations.

REFERENCES

1. S. Kumar, S. Singh, and A. Gupta, "Survey and Evaluation of Food Recommendation Systems and Techniques," in Proc. 3rd Int. Conf. Computing for Sustainable Global Development (INDIACom), IEEE, 2016, pp. 2744–2749.
2. H. Gunawardena, "BestDish: A Digital Menu and Food Item Recom-mendation System," in Proc. 2nd Int. Conf. Advancements in Computing (ICAC), IEEE, 2020, pp. 168–173.
3. T. Melese, "Food and Restaurant Recommendation System Using Hybrid Filtering Mechanism," NAAR Journal, vol. 3, no. 1, pp. 73–81, 2021.
4. S. Trapp, C. Leech, D. Evenson, and L. Lytle, "Nutritional Quality of Children's Menus in Restaurants: Does the Type of Cuisine Matter?," Public Health Nutrition, vol. 26, no. 2, pp. 384–392, 2023.
5. A. Kumar and P. Gupta, "Behavior-Based Food Recommendation Using User Preference Modeling," Int. J. Computer Applications, vol. 178, no. 7, pp. 12–18, 2019.
6. J. Smith and R. Brown, "Sentiment Analysis Based Recommendation System for Restaurants," in Proc. Int. Conf. Data Mining and Applications, 2018.
7. M. Patel and S. Shah, "Augmented Reality Based Restaurant Recom-mendation System," in Proc. IEEE Conf. Emerging Technologies, 2021.
8. R. Zhang, Y. Liu, and H. Wang, "Context-Aware Food Recommendation Using Environmental Factors," J. Artificial Intelligence Research, vol. 63,
9. pp. 221–240, 2020.
10. K. Lee and J. Park, "Emotion Recognition Using Deep Learning for Human-Computer Interaction," IEEE Access, vol. 8, pp. 172311–172320, 2020.
11. OpenWeather, "OpenWeather API Documentation," [Online]. Available: https://openweathermap.org/api