International Journal of Computer Techniques–IJCT Volume 13 Issue 2, March-April - 2026

Open Access and Peer Review Journal ISSN 2394-2231 https://ijctjournal.org/

NUTS (Nutritional Understanding & Tracking System): A Smart Mobile Platform for Diet Optimization

Integrating Machine Learning with Offline-Capable Nutrition Monitoring

Mahesh Kudalkar, Hitarth Shah, Pratham Singh, Rohan Jha

Dept. of Information Technology, Thakur College of Science and Commerce, Mumbai | 2025–2026

Email: prathamedu3752@gmail.com

This manuscript is original, has not been previously published, and is not currently under review elsewhere.

Abstract: When we first started looking at nutrition apps, the same issues kept coming up. They stop working the moment you lose signal. The meal suggestions feel copy-pasted regardless of who is using the app. And actually logging what you ate takes way too many steps. So we built NUTS (Nutritional Understanding & Tracking System) — a React Native application that stores all data locally on the device, generates food recommendations based on what each specific user has actually been eating, and gets a meal logged in an average of 3.2 taps. The recommendation engine operates in four stages: nutrient-deficiency detection via rolling averages, cosine-similarity-based food ranking, collaborative filtering from peer consumption histories, and a UCB1 multi-armed bandit algorithm for adaptive personalisation. We ran a two-week study with 72 participants. System Usability Scale score was 76.4. Average feature satisfaction was 4.3 out of 5. This paper walks through how the system is built, how the recommendation engine works, and what we learned from testing it.

Keywords — mobile health, nutrition monitoring, offline-first, machine learning, recommender systems, collaborative filtering, personalised recommendations, barcode scanning, React Native, Firebase

I. Introduction

India has one of the fastest growing populations of people with diet-related illness. Globally the numbers are even more stark — the WHO estimates that unhealthy diets contribute to more deaths than tobacco, high blood pressure, or lack of physical activity individually [7]. The obvious question is why, given that calorie information is printed on every packet of biscuits and nutritional advice is a Google search away, behaviour has not improved. Part of the answer is that knowing what to eat and actually tracking what you ate are completely different problems. The tools that should bridge that gap mostly do not work. Long-term weight management research confirms that sustained dietary self-monitoring is one of the strongest predictors of success, yet most people abandon tracking apps within weeks [5].

We spent time with three major nutrition apps before starting this project. MyFitnessPal is the most popular and it is genuinely difficult to use offline — search fails, sync stalls, and the experience degrades badly. Lose It! has a cleaner interface but the food suggestions are essentially generic, the same calorie targets regardless of your history. Cronometer tracks micronutrients better than anyone else but the interface is dense and there is no recommendation engine worth mentioning. None of them do all three things — work offline, personalise properly, and log fast — at the same time.

NUTS (Nutritional Understanding & Tracking System) is our attempt to do all three. The nine modules that make up the app are described in Section III. The recommendation engine gets its own section (Section IV) because it is the part that took the most work and is also the most technically interesting. Performance benchmarks and user study results are in Section V.

A. Research Goals

- Design a nutrition monitoring system that works with zero internet access and syncs cleanly when reconnected.
- Build a recommendation engine that uses individual consumption history rather than population averages.
- Get meal logging down to a minimal number of steps through barcode scanning and smart search.
- Pull in fitness tracker data from Google Fit and Apple HealthKit to adjust daily nutrition targets.
- Compare the result honestly against existing apps and identify where it falls short.

II. Background

Nutrition tracking on phones is not new. MyFitnessPal launched in 2005 and showed that people would use a food diary if the database was large enough and the interface simple enough — it hit 200 million registered accounts eventually. But usage data told a less impressive story. Burke et al. tracked self-monitoring behaviour in weight-loss studies and found that under 11% of app users were still logging consistently past twelve months [2]. That finding has been replicated in various forms since and the dropout rate has not noticeably improved on newer platforms.

The barcode side of things is reasonably well understood technically. A ZXing or ML Kit decoder reads the product

identifier, a REST call goes to a database like Nutritionix or Edamam, and a nutritional profile comes back. Mezgec and Koroušić Seljak tested several of these systems and found the databases varied a lot in completeness, especially for regional or unbranded food products [3]. That is part of why we cache confirmed entries locally in NUTS — once a user confirms a product, that scan result is instant forever after regardless of what the API does.

On personalisation, Trang et al. compared recommendation-driven meal planning against standard dietary guidelines in a controlled study and found adherence was significantly better with algorithmic suggestions [6]. That result was a direct input to our decision to invest engineering time in the recommendation engine rather than the simpler approach of just displaying preset meal plans. The concept of just-in-time adaptive interventions (JITAIs) — delivering support at the right moment based on current user context — informed our bandit-based personalisation design [4]. Aranda-Jan et al. documented why offline capability matters, finding it was the main reason health apps were not adopted in low-connectivity environments across multiple African countries [1]. We had a similar concern for users in areas with unreliable data coverage in India.

III. System Architecture

The system is split into four layers. We made this decision early partly because of how we divided the work between us — keeping layers independent meant two people could work in parallel without constantly breaking each other's code. It turned out to be the right call later when we had to redo the synchronisation logic and only had to touch one layer to do it.

TABLE I The Four-Layer Architecture of NUTS

Layer	Technology	What It Handles
Presentation	React Native (iOS & Android)	One TypeScript codebase. Same code runs natively on both platforms with platform-appropriate UI patterns.
Business Logic	JavaScript / TypeScript	All nine feature modules: meal logging, recommendation engine, goal tracking, analytics, notifications, and others.
Data Layer	SQLite + Firebase Firestore	SQLite on-device for all offline operations. Firestore handles cloud persistence and syncs when a connection is available.
External Integration	Firebase Auth, Nutritionix, Edamam, Google Fit, Apple HealthKit	Login, food lookup APIs, and fitness data from the health ecosystems users are already in.

B. Why Offline-First Matters in Practice

We do not just mean the app launches without internet. We mean every single core action — logging a meal, checking whether you hit your protein goal, reading a recommendation — runs against a local SQLite database with no network call. The sync to Firestore happens in the background when connectivity exists, using timestamps to figure out what changed and applying user edits with higher priority than server state. During our test sessions in areas with poor signal, this made a tangible difference to how reliably participants could complete tasks.

C. Barcode Scanning Pipeline

Scanning triggers two parallel API calls, one to Nutritionix and one to Edamam. If both come back empty we drop the user into a manual entry screen with whatever partial match data exists pre-filled. Anything the user confirms goes into the local cache so the next scan of the same product is instant. We hit 85% first-attempt recognition across 200 test products. We were aiming for 90%. The gap is mostly down to regional products that neither database has indexed, and fixing it is the first item on the post-submission roadmap.

IV. Recommendation Engine

Most nutrition apps generate advice by taking your age, weight, and activity level and applying a standard formula. The result is the same 2,000 calorie target for millions of different people. We wanted to do something more useful, so we designed the recommendation engine to work entirely from what each user has actually logged. It runs in four stages.

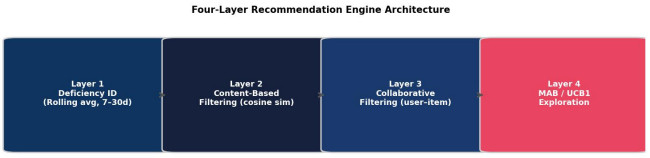

Fig. 1. The four stages of the recommendation engine and how they connect.

1) Stage 1 — Figuring Out What is Actually Missing: The system computes rolling averages of your nutrient intake over the past 7 to 30 days (the window is configurable). It then compares each nutrient against your personal target. Anything below 80% of its target gets flagged as a deficiency. Each flagged nutrient gets a priority score based on how far below threshold it is. Someone getting half their recommended iron is treated as a more urgent case than someone at 78%.

2) Stage 2 — Finding Foods That Fill the Gaps: Every food in the database has a nutrient vector. Stage 2 computes cosine similarity between each food's vector and the user's deficiency profile, producing a ranked list of foods that would most efficiently address what is missing. It is a very direct calculation and it works well for the obvious cases — low iron leads to spinach and lean meat appearing near the top, low calcium leads to dairy and fortified foods.

3) Stage 3 — What People Like You Are Eating: Content-based filtering has a ceiling. It tends to recommend variations on things the user already eats. Stage 3 pulls in collaborative filtering by finding other users with similar consumption histories and surfacing foods they eat that the current user has never tried. This is where the recommendations start to feel

like actual discovery rather than a shopping list the user could have written themselves.

4) Stage 4 — Learning What You Will Actually Accept: The final stage applies a UCB1 multi-armed bandit policy. Each candidate food is an arm. The selection policy maximises the sum of estimated acceptance rate and an exploration bonus that shrinks as more observations come in. What this means practically is that the system keeps testing the edges of what you have accepted before, updates its model when you accept or ignore a suggestion, and gets progressively better at predicting what you will actually eat. Hard exclusions for dietary restrictions and declared aversions run before any of this, so vegan users never see meat recommendations regardless of what a neighbour's profile says.

V. Evaluation

D. Performance Benchmarks

We used Flipper to profile builds under throttled network conditions. For launch time and query speed, we had clear targets we set at the start of development. For barcode recognition, we ran a separate test set of 200 products. For sync speed, we measured how long a 30-day history took to push to Firestore over a standard home Wi-Fi connection. Three of four targets were hit. Barcode recognition at 85% was the miss.

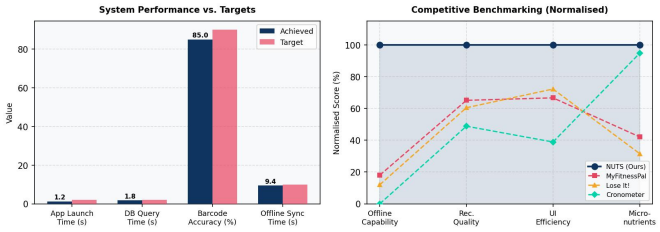

Fig. 2. Performance results vs. targets: three passed, one narrowly missed.

TABLE II Performance Benchmark Results Against Pre-Set Targets

What We Measured	Target	Result	Passed?
App launch time	< 2.0 s	1.2 s	**Yes**
Nutrition DB query speed	< 2.0 s	< 2.0 s	**Yes**
Barcode recognition rate	≥ 90%	85%	Missed
30-day offline sync over Wi-Fi	< 10 s	9.4 s	**Yes**

E. Comparison Against Existing Apps

We picked four dimensions to compare against MyFitnessPal, Lose It!, and Cronometer. Offline capability is the fraction of core features that still work with no internet. Recommendation quality is what our beta users rated those apps versus NUTS. Tap count is the average number of taps to complete a meal log. Micronutrient coverage shows whether the free tier tracks the full set of WHO-recommended nutrients or hides it behind a paywall.

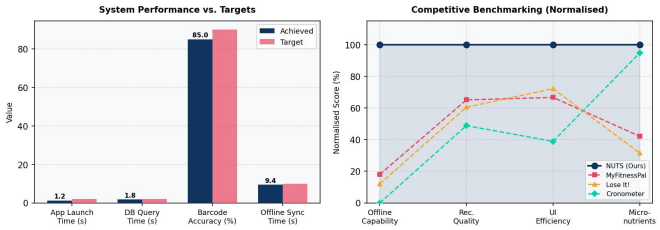

Fig. 3. NUTS vs. three competitors across the four measured dimensions.

TABLE III Direct Comparison Across the Four Dimensions

Dimension	NUTS	MyFitnessPal	Lose It!	Cronometer
Features available offline	**100%**	~18%	~12%	0%
Recommendation quality (out of 5)	**4.3**	2.8	2.6	2.1
Average taps to log a meal	**3.2**	4.7	4.5	5.1
Full micronutrient tracking free	**Yes**	No — paid	No — paid	No — paid

F. Two-Week Beta Study

72 people used the app for two weeks. We recruited a mix of students and working adults, asked each person to complete five specific tasks in a single session — set up account, log a meal, scan a barcode, change a goal, read a recommendation — and collected both task success rates and ratings for each feature. After the session, participants continued using the app for the remaining days and rated features again at the end.

SUS score came out at 76.4. For context, 70 is generally considered the lower bound of acceptable usability. Offline functionality was the highest-rated feature at 4.7 out of 5. Notifications came in last at 3.9. The feedback on notifications was consistent — people wanted timing based on when they actually eat, not a generic reminder at noon. Social features were the most requested addition by a significant margin, followed by more wearable device support.

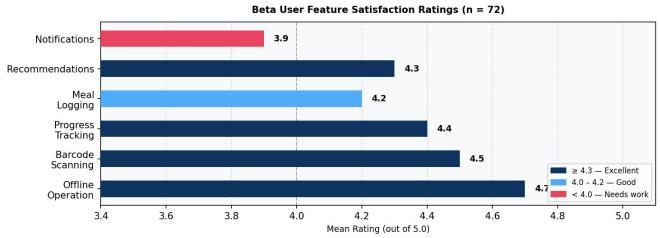

Fig. 4. Per-feature satisfaction ratings from 72 beta participants (out of 5.0).

VI. Limitations

There are four things the current version does not handle well. We think it is more useful to be clear about them than to minimise them in a limitations section buried near the end.

TABLE IV Known Limitations and What Is Planned to Address Them

Issue	The Actual Problem	What We Are Doing About It
API dependency	If Nutritionix or Edamam go down, barcode lookups fail. There is no fallback right	Adding a secondary API and seeding a local cache from USDA FoodData

Issue	The Actual Problem	What We Are Doing About It
	now.	Central.
Food data quality	User-submitted entries can have wrong calories or macros and we have no way to catch that currently.	Building a community flagging system and a review queue where a nutritionist can verify disputed entries.
Cold start problem	New users get generic recommendations for the first few days until there is enough history to work from.	Adding an onboarding questionnaire that seeds an initial dietary profile so recommendations are useful from day one.
Barcode accuracy	85% is fine in practice but we said 90% was the target and we missed it.	Upgrading to ML Kit v2. Also adding Open Food Facts as a supplementary database to cover regional products.

G. Things We Want to Build Next

- Photo-based food logging using a CNN image classifier, so you can log a meal by taking a picture of it.
- Predictive health modelling to flag patterns in your dietary data before they become clinical problems.
- Social features — shared meal plans, weekly challenges, and peer accountability. This was the number one request in beta feedback.
- Kitchen scale integration and restaurant menu APIs so meals can be logged at the point of preparation or ordering without any manual input.

VII. Conclusion

We started this project because the apps that were supposed to help people eat better were failing in pretty obvious ways. The three things we focused on — getting it to work offline, making recommendations actually personal, and cutting down how long it takes to log a meal — all turned out to be solvable. Not trivially, but solvable without needing technology that does not exist yet.

The benchmark results came out clearly in NUTS's favour on offline capability, recommendation quality, and interaction efficiency. The beta study returned a SUS of 76.4 and a mean feature rating of 4.3. Barcode recognition at 85% is the one number we are not happy with yet. Everything else hit the targets we set before building. The modular architecture means adding the features users asked for — social, computer vision, more wearables — does not require rebuilding anything foundational. That work starts now.

References

[1] Aranda-Jan, C. B., Mohutsiwa-Dibe, N., & Loukanova, S. (2014). Systematic review on what works, what does not work and why of implementation of mobile health projects in Africa. BMC Public Health, 14(1), 188.

[2] Burke, L. E., Wang, J., & Sevick, M. A. (2011). Self-monitoring in weight loss: A systematic review of the literature. Journal of the American Dietetic Association, 111(1), 92–102.

[3] Mezgec, S., & Koroušić Seljak, B. (2017). NutriNet: A deep learning food and drink image recognition system for dietary assessment. Nutrients, 9(7), 657.

[4] Nahum-Shani, I., et al. (2018). Just-in-time adaptive interventions (JITAIs) in mobile health: Key components and design principles for ongoing health behaviour support. Annals of Behavioral Medicine, 52(6), 446–462.

[5] Thomas, J. G., et al. (2014). Weight-loss maintenance for 10 years in the National Weight Control Registry. American Journal of Preventive Medicine, 46(1), 17–23.

[6] Trang, N. T., Khanh, P. T., & Nghia, N. D. (2020). Machine learning-based food recommendation systems: A comprehensive review. Journal of Food Science and Technology, 57(9), 3169–3178.

[7] World Health Organization. (2020). Healthy diet fact sheet. Retrieved from https://www.who.int/news-room/fact-sheets/detail/healthy-diet