DEMOCRATIZING NUTRITIONAL KNOWLEDGE: A NATIVE ANDROID ARCHITECTURE FOR REAL-TIME DIETARY MANAGEMENT VIA BARCODE SCANNING

Farhan Iftekhar Shaikh[1], Dr. Shiv Kumar Goel[2]
[1]Student, Final Year MCA, Vivekanand Education Society's Institute of Technology, Mumbai, India
[2]Associate Professor of MCA, Vivekanand Education Society's Institute of Technology, Mumbai, India

***Abstract*:** Most food labels confuse people. Fewer diet applications have the most important information about allergy issues available to users for no cost. We chose to develop our application in an open way with the idea of developing an android based solution. Rather than use proprietary products we used free products like Open Food Facts API and Google's barcode scanner using ML kit. User preference is stored in a secure local database. In addition to these features the application was developed with a structured architecture pattern called MVVM. Users will receive instant answers and display nutrient information through a visual color scheme. Users who identify as having specific allergies will also receive warning messages related specifically to their allergy or allergen(s). Additionally the application will display environmental foot print associated with each item at the point of decision making. We anticipate that developers will be able to disassemble this application, analyze its parts and then build upon them.

IndexTerms - mHealth, Usability, Barcode Scanning, Open Source, Nutritional Literacy, MVVM Architecture, REST API, Android, Blueprint, Reference Architecture.

1. INTRODUCTION

Mobile phones will be able to make it easier for individuals to monitor their diet. Since mobile phone cameras can read the barcode of packaged foods, as well as provide information from a database of nutrition content for thousands of items, finding the proper information regarding food intake will become much faster than checking food labels by hand. A system such as this provides a relatively simple example of how one could create an android based version of a program like this.

Rather than creating an application for sale, this project includes a plan (blueprint) using open source software development tools. This allows users to get visual feedback of what they are consuming through easy-to-read format without having to decipher ambiguous labeling terms. In other words, this project is not focused on developing polished applications for consumers; rather, the focus is on providing an open source platform that can be copied and modified into different applications with various features for future developers/researchers working in related areas such as dietary management/mhealth.

The main contributions of this project were not producing a final, commercially viable application; instead, it has provided documentation of a method for assembling free, open-source technologies to form an application that any developer/ researcher interested in dietary management or closely related health-related fields may modify/adapt/duplicate.

2. LITERATURE REVIEW

To provide an example of how today's apps for mobile devices could be improved (with regard to nutrition), consider all the applications that exist - but most lack a shared set of rules to follow when sending data to one another [8], handle allergic reactions poorly, or support users equally.

2.1. The Transparency Gap

Another major issue with many of today's digital health tools is that they apply proprietary algorithms to assess the nutritional value of foods. For example, rating apps like Yuka [2] rate foods quickly, however since their formulas are proprietary, they hide their underlying code from view. As a result, consumers will have no ability to audit the individual components of the formula and therefore no way to determine what certain ingredients may mean for their health. This will erode consumer confidence in the app over time. Conversely, systems developed using public domain methods expose each and every element for review, testing, and even modification by other developers interested in making improvements.

2.2. The Information Specificity and Accessibility Gaps

For individuals suffering from severe food allergies, many existing apps simply do not adequately protect them. For instance, while Fooducate provides extensive breakdowns of ingredients, typically full access to allergen information requires the user to pay for a premium version of the application. Similarly, tools like MyFitnessPal require users to pay for "smart" features, including automatic scanning of barcodes, before they can use them. By charging for basic functionality, users experience two additional burdens; they incur costs associated with accessing the feature, and the process creates confusion, particularly among vulnerable populations who require rapid and clear information regarding potential threats related to their diets.

The alternative introduced by this research is an approach based on completely publically accessible data sets and software that can be audited or modified by anyone. This study outlines a specific architecture for eliminating these barriers by utilizing only free, open-source databases and libraries.

3. PROBLEM DEFINITION

Individuals with adverse physical reactions to specific food items have difficulty identifying harmful components. Many digital tools restrict access to essential medical data through subscription fees. And other platforms present complicated scientific facts in a manner that is difficult for a person to interpret. It is common for consumers to state that the contents of their meals are important - but many people do not examine the product data when they select items from a store.

In a small study of twenty individuals, the data demonstrates this inconsistency. As an example 75% of the participants stated that they do not inspect the nutritional table prior to a purchase. On the shelves most shoppers look only at the front design of a container. Or they neglect to read the small text entirely. To understand the difficulty of interpreting modern containers, participants provided feedback on the process. Nine out of ten people described the task as one that requires significant effort. As Figure 2 indicates, 85% of the group selected a difficulty rating of 3 or more on a scale of 5. There is a high frequency of uncertainty among the public rather than occasional errors [9].

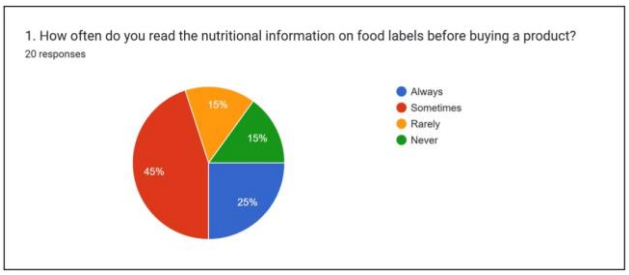

Fig. 1: Frequency of reading nutritional labels prior to purchase (n=20).

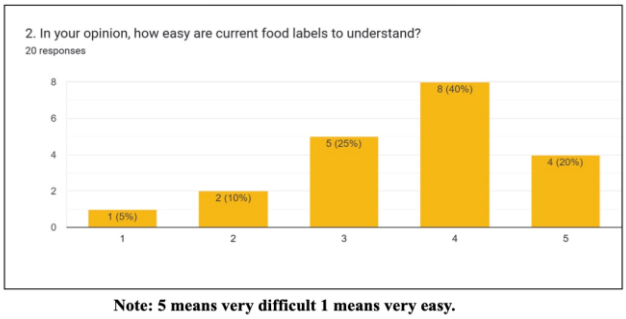

Fig. 2: Perceived difficulty of understanding food labels on a 5-point scale (n=20).

If the small group size makes it difficult to apply those results to the entire population, the data provides a clear path for the project. By using the results, it is possible to establish the technical requirements for the structure of the system.

4. OBJECTIVE AND SCOPE

To achieve the objectives specified within this research paper, I am developing an architecture and functions (goals) that will allow a replicable, open-source native android framework. The purpose of this project is to show how these free/available technologies may be integrated into solutions for real world food/dietary management applications, and how the developed architecture can be used by other developers for creating their own mHealth apps/tools.

4.1. Research Objectives

The primary objectives of this research include:

1. To build a single reproducible architecture for one barcode scan, which will unlock complete information about a products nutritional content instantly while providing users a clear visual view of exactly what is contained in each product they decide to purchase.
2. To create a pattern for representing nutritional/caloric data using clean and easy to use card-based layouts. Using publically available data this layout was designed to provide maximum clarity and space, and minimize technical complexity.

3. To document how color-coded allergen/additive flags can be easily added using publicly available data bases. Since we do not want to be locked into a proprietary solution, our goal was to ensure that safety markers would always be displayed properly regardless of the device being used.
4. To build a system that continues to function and provide users with access to their personal data and/or security features whether they have a consistent connection to the Internet, or if connectivity is spotty, utilizing a combination of off-line first cache capable local database technology.

4.2. Scope of the System

This project's scope consists of the design and prototype development of a native Android app written in the Kotlin programming language as a proof-of-concept example of the blueprint described. The app will also include real-time interaction with the Open Food Facts REST API [1], a service that provides unbiased, open-source data. Additionally, it will include Room Database architecture to persistently store users' personally identifiable allergens/data, along with a history of previously viewed/cached product information. The documentation provided here for the architectural elements identified in this blueprint (MVVM layering, REST API DTO mapping, on-device machine learning, offline-first caching), and are applicable to any mHealth domain requiring real-time data at the point of need.

5. SYSTEM ARCHITECTURE AND METHODOLOGY

One key factor that ties the whole system together is a design based on an open-source model. The proposed framework will be created using the Kotlin programming language running on top of Android. There is no need for excessive layers of complexity as there is enough room for separation of concerns with Model-View-ViewModel (MVVM) (Fig. 3) to fit into. What keeps it all from falling apart? A combination of products that do not require a fee to use. Each piece fits together perfectly due to their free nature, their level of community support, and the fact that they are suitable for use on devices located at the network's edge.

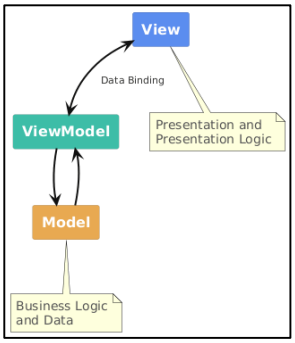

Fig. 3: MVVM Architecture of the Proposed Blueprint.

5.1. Open-Source Technology Stack and Selection Rationale

A major decision that influenced the overall strategy of the project involved selecting a variety of free, open-source tools. In addition to selecting a tool for each layer of the MVVM architecture (Table 1), I have also identified why I selected them over the common closed and/or commercial alternatives. Furthermore, this open-access prevents vendor lock-in, simply allowing the whole system to evolve independently.

TABLE 1: Open-Source Technology Stack: Components and Selection Rationale

Component	Chosen Solution	Rationale vs. Alternative
Food Database	Open Food Facts REST API	Free, open, community-verified; Edamam and Spoonacular require paid API keys
Barcode Scanning	ML Kit Barcode Scanning	On-device, no network call needed; ZXing is deprecated and lacks active maintenance
Authentication	Firebase Auth	Free tier sufficient; eliminates need to build and maintain a custom auth backend
Local Storage	Room Database	Type-safe SQLite abstraction; native Kotlin coroutines support; enables offline-first design
Architecture	MVVM + Repository Pattern	Enforces separation of concerns; simplifies unit testing; recommended by Android Jetpack guidelines

5.2. MVVM Layering and Responsibilities

The use of the MVVM architecture in the model presented here spans three levels and each level has specific roles:

1. View Layer (Activities and Fragments): responsible for displaying UI state and transmitting any user actions or input (triggering a scan, entering an allergen preference) to the ViewModel.
2. ViewModel Layer: Functions as the sole repository for all UI state. Provides the Views with access to a number of LiveData objects. The ViewModel does not contain any code for retrieving data; instead, it delegates data retrieval tasks to the Repository.
3. Repository and Data Layer: Coordinates requests for data from two different sources — the Open Food Facts REST API (a remote source) and the Room database (a local cache). The Repository maps the raw payload returned from the API into filtered versions of only those fields that have relevance to making dietary choices.

5.3. System Workflow

In addition, Fig. 4 shows the overall activity diagram illustrating how the entire workflow for a consumer's experience flows from scanning a product's barcode to obtaining a tailored nutritional result. The flow has been streamlined so that when consumers need to make dietary decisions at point of sale they do not have to manually enter data themselves.

5.4. On-Device Barcode Detection with ML Kit

The ML Kit Barcode Scanning tool reads the barcodes locally without any network calls, so this is "instant" response on scan as the local handling has been set up from very early on. Even when the network is entirely out you still have working functionality. Standard EAN-13 and QR codes come readily available from the library so they will scan almost any item that is currently on grocery packaging. Feel free to use a different barcode scanning libraries instead (these will run locally too) with no alteration of the above framework needed.

5.5. Offline-First Design with Room Database

A Room Database is used to locally store two types of data: allergen profiles created by the user and the product details of previously scanned products. Data storage allows for use without Internet during scanning at a later stage since initial users indicated clearly they require this functionality. While the user is checking a product again, the result that was saved during the first check is taken into consideration first without going to remote servers. Thus response time is shortened with less network traffic.

5.6. Open Food Facts API Integration

The nutritional information used in this project was obtained through interface to the Open Food Facts REST API [1]. After a successful barcode scan, the Repository layer creates a formatted HTTP GET request to the API, specifying the scanned barcode as a query parameter and limiting the fields returned based on explicit inclusion/exclusion criteria specified via query parameters. The resulting JSON is parsed and transformed into one or more local Data Transfer Objects (DTOs), which discard many of the hundreds of metadata fields — such as contributor edit history and packaging recyclable materials — that are completely irrelevant to food purchase decisions made at point of sale.

The process of transforming the raw data into DTOs is a fundamental part of this model and will be analyzed in detail in Section 6. A sequence diagram showing communication among all the various system components is shown in Fig. 5.

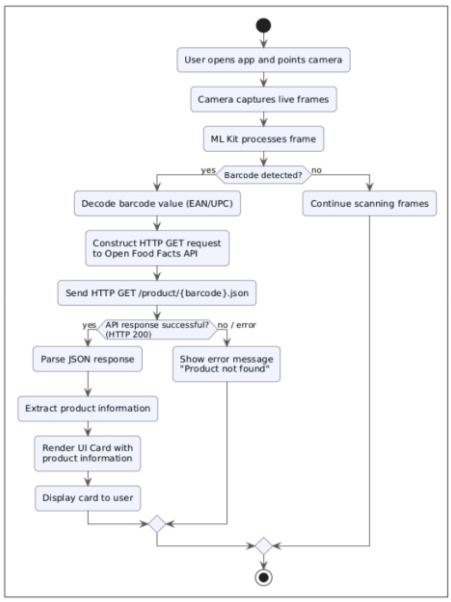

Fig. 4: Activity Diagram illustrating the user interaction flow from barcode scan to nutritional result display.

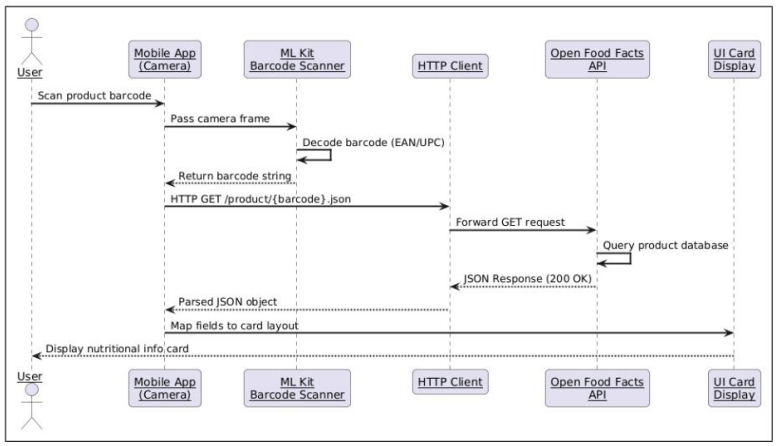

Fig. 5: Sequence Diagram depicting the message exchange between application components and the Open Food Facts REST API.

6. ANALYSIS AND FINDINGS

The objective of these evaluations is to assess the technical feasibility of creating mHealth applications that employ an open-source stack. The findings respond to the question a developer using this blueprint would reasonably have: can these free-to-use technologies perform reliably in the wild?

6.1. API Response Time Evaluation

We performed 5 scans of the Open Food Facts REST API in various product categories to assess it as an entity with a zero-cost data source. The REST client recorded the response times and these are displayed in Table 2.

TABLE 2: Open Food Facts API Response Time Across Multiple Scans

Scan	Product Category	Response (ms)	Status
1	Oreo Chocolate Biscuit	677	200 OK
2	Coca-Cola Beverage	615	200 OK
3	Kit Kat Chocolate Bar	976	200 OK
4	Cheerios Breakfast Cereal	931	200 OK
5	Maggi 2-Minute Noodles	763	200 OK
Avg.	—	792.4	100% OK

The findings reveal that the Open Food Facts REST API provides consistently reliable performance, with an average response time of 792.4 ms and a 100% success rate in scans for all product categories. Mobile consumer applications that are real-time in nature are generally considered acceptable if their response times remain less than a second [10]. Thus, our choice of using the Open Food Facts REST API as a zero cost, reliable data source for dietary management at point of sale is validated.

6.2. Network Payload and Bandwidth Optimization

The transmission and the local storage large data payloads which mobile applications need to handle pose a significant architectural challenge. Open Food Facts database offers many product profiles which contain hundreds of metadata fields that are irrelevant to consumer dietary choices at the point of sale. In order to save bandwidth and edge-device memory, strict DTO mapping and use-case-specific API query filtering is employed. There are structural constraints for HTTP GET requests as end users only need the most critical safety and nutritional fields such as product_name, nutriments, allergens_tag and nutriscore_grade. To measure the effectiveness of this design choice, we have compared the sizes of raw, unfiltered JSON payloads to sizes of filtered payloads as processed by the Repository layer for five test products in Table 3.

TABLE 3: Data Payload Optimization: Raw vs. Filtered API Responses

Product Category	Raw (KB)	Filtered (KB)	Reduction
Oreo Chocolate Biscuit	115.0	2.10	98.2%
Coca-Cola Beverage	21.8	0.49	97.8%
Kit Kat Chocolate Bar	20.45	1.40	93.2%
Cheerios Breakfast Cereal	45.2	2.70	94.0%
Maggi 2-Minute Noodles	21.8	1.67	92.3%
Average	44.85	1.67	96.3%

The building process of propose query parameter and DTO mapping reduce payloads by respective for 96.3% from average data size of 44.85KB (before data) to 1.67KB per scan (after data).

As a result, this minimizes the bandwidth consumption of the network and speeds up the JSON parsing times at the edge device which is essential for deployment at low bandwidth configurations.

6.3. Open Food Facts Database Coverage Analysis

The developer adopting this blueprint should take note of the coverage characteristics of the Open Food database since the blueprint relies solely on this source. To evaluate this, we scanned 20 packaged products from three categories: internationally designed products, nationally distributed Indian products, and regional/local Indian products.

TABLE 4: Open Food Facts Database Coverage by Product Category (n=20)

Category	Scanned	Full Data	Partial/No Data
International Brands	8	8 (100%)	0 (0%)
National Indian Brands	7	5 (71%)	2 (29%)
Regional/Local Brands	5	1 (20%)	4 (80%)
Overall	20	14 (70%)	6 (30%)

In Table 4, we see complete coverage for international products while coverage for regional Indian products is fairly low. Developers that are primarily targeting the Indian consumer should expect this gap and help curate the missing product data by submitting it to the Open Food Facts community database or by using a secondary source to patch it up. This will be a priority area in the future scope.

6.4. Comparative Analysis Against Proprietary Alternatives

To illustrate the practicality and accessibility of the proposed architectural design, the native Android versions of three leading proprietary food applications in mHealth Domain (Yuka, MyFitnessPal and Fooducate) were compared to each other. By evaluating the Android deployments, a direct comparison can be made with the framework. This assessment will look at the availability of crucial real-time features, data transparency, and offline capabilities. Through direct investigation of their current implementations, as well as their premium subscription documents from publishers, the feature sets, offline limitations and paywall limitations of proprietary applications were settled [5]-[7].

The open-source framework suggested in this study not only matches the core functionalities of these systems, as shown in Table 5, but also renders them practically affordable while avoiding opaque, black-box scoring algorithms.

TABLE 5: Comparative Analysis: Proposed Blueprint vs. Proprietary Alternatives

Capability	Proposed Blueprint	Leading Alternatives
Barcode Scanning	✓ (Free)	Mixed (Often Premium)
Nutritional Analysis	✓ (Free)	✓ (Free)
Allergen Alerts	✓ (Free)	$ (Premium / Manual)
Offline Caching	✓ (Local DB)	× (Network Required)
Code Availability	✓ (Open Source)	× (Closed Source)
Scoring Logic	Transparent (Open)	Opaque (Black Box)

Consolidated Overview of Yuka, MyFitnessPal and Fooducate. Key: ✓ = Included Natively, $ = Paywalled, × = Unavailable

7. LIMITATIONS AND FUTURE SCOPE

7.1. Limitations

Though the plan looks good, it has flaws. A small sample of 20 respondents were taken for preliminary pilot survey means it may not display the trends. Thus, it is understood as an early signal and not the proof. Open Food Facts data contains many absence gaps, particularly in the case of local Indian packaged foods, as the audit shows. At this time, the app is only compatible with Android devices, leaving out iPhone users. It can be checked offline only if you have previously seen that item or else it won't load unless you connect to the internet.

7.2. Future Scope

Later versions of this blueprint will correct these shortcomings in several ways. With Flutter or Kotlin Multiplatform, the architecture will be made available to iOS users without repeating business logic. Wearable health devices will allow continuous, passive dietary monitoring, beyond point-of-sale scanning. A personalized meal recommendation engine powered by AI that is trained on anonymized scan histories will provide proactive advice. According to the report, an OCR-based label scanning of regional Indian languages packaging is a high-priority addition. Alongside, formal database verification with FSSAI will be done to enhance domestic product coverage. Soon people will be allowed to upload and suggest missing Indian products to Open Food Facts.

8. CONCLUSION

The primary contribution of the current study was not the development of proprietary software, however it was the documentation of how existing free technologies can be used together as an architectural model for real time dietary management using bar code scans. These free technologies include the Open Food Facts REST API for accessing nutrition information, Google's MLKit Barcode Scanner, and Room Database for storing local user preference and allergy settings.

Together these technologies were integrated into a Model View ViewModel (MVVM) pattern to provide users with immediate access to both color coded nutritional analyses and personalized allergen alerts at the point of purchase in addition to providing

such information without the costs associated with subscription services or proprietary data. Technical validation of this open-source stack demonstrated its technical feasibility: the Open Food Facts API had less than one second response time and a 100 percent successful search rate for all categories of products that were tested; and data object mapping with selective query filtering resulted in an average of a 96.3 percent reduction in payload size which greatly improved the usability and efficiency of the application on edge devices as well as on lower bandwidth networks.

Additionally, a database coverage analysis provided developers with actionable expectations concerning whether there would be sufficient amounts of international versus local Indian product data available in the database. Findings from preliminary surveys also validate that there exists a sizeable knowledge-action gap among consumers who would benefit from dietary interventions like the one being described. As noted previously, the architectural patterns documented here (MVVM layering, REST API DTO mapping, on device machine learning, offline first caching) can be applied to any type of mobile health application that requires real time data at the point of need. Therefore, the proposed architectural blueprint serves as a meaningful reference architecture for developers and researchers working on mobile health tools designed to address the needs of vulnerable populations living with dietary restrictions and food allergies.

REFERENCES

[1] Open Food Facts REST API Documentation. https://openfoodfacts.github.io/openfoodfacts-server/api

[2] Yuka Application – Social Impact. https://yuka.io/en/socialimpact

[3] Fooducate – Nutrition Tracker. https://www.fooducate.com

[4] MyFitnessPal – Calorie Counter. https://www.myfitnesspal.com

[5] Yuka, "Yuka Premium Version Features," Official Website. https://yuka.io/en/premium-member/

[6] MyFitnessPal, Inc., "MyFitnessPal Premium Features: Barcode Scanner," Official Website. https://support.myfitnesspal.com/hc/en-us/articles/360032271452

[7] Fooducate, "Fooducate Pro Subscription and Premium Features," Official Website. https://appbundles.com/best-apps-bundle?app=fooducate

[8] Franco, R.Z., Fallaize, R., Lovegrove, J.A., Hwang, F.: Popular Nutrition-Related Mobile Apps: A Feature Assessment. JMIR mHealth and uHealth 4(3), e85 (2016). https://doi.org/10.2196/mhealth.5846

[9] Maringer, M., Wisse-Voorwinden, N., Veer, P.V., Geelen, A.: Food Identification by Barcode Scanning: A Quality Assessment of Labelled Food Product Databases Underlying Popular Nutrition Applications. Public Health Nutrition 22(7), 1215–1222 (2019). https://doi.org/10.1017/S136898001800157X

[10] Klasen, L. et al.: NutriDiary, a Smartphone-Based Dietary Record App: Description and Usability Evaluation. JMIR Human Factors 12, e62776 (2025). https://doi.org/10.2196/62776

[11] Khazen, W., Jeanne, J.F., Demaretz, L., Schäfer, F., Fagherazzi, G.: Rethinking the Use of Mobile Apps for Dietary Assessment in Medical Research. Journal of Medical Internet Research 22(6), e15619 (2020). https://doi.org/10.2196/15619