iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

FOOD FACTS APP: A FLUTTER-BASED OFFLINE NUTRITION TRACKING APPLICATION WITH SQLITE INTEGRATION

C. Suganya
Student, School of Computing Sciences, Vels Institute of Science, Technology and Advanced Studies, Chennai, India

Dr. S. Rani,
M.SC., M.Phil., Ph.D., Professor, school of computing sciences, Vels Institute of Science, Technology And Advanced Studies, Chennai, India

ABSTRACT
In this paper, we propose the Food Facts App; a cross-platform mobile application using Flutter as the programming language that empowers users by providing them with complete offline nutritional information. The app incorporates SQLite for local storage capable of querying a searchable database of food items with nutrition profiles in terms of calories, macronutrients (protein/carb/fat), vitamins/minerals and allergens. Some of the main features are: - Real-time food data access by scanning bar code. - Personal dietary suggestions. - Daily nutrition overview with charts and stats. - Role-based user interfaces or subscriptions for regular end-users and administrators (eg, dietitians). The application achieves 95% test coverage, sub-100ms query times, and 60fps UI rendering. This paper describes the system architecture, design methodology, module structure, testing strategy, and future enhancement roadmap for the application.

Keywords:
Flutter, SQLite, mobile health application, nutritional tracking, barcode scanning, offline-first, dietary management, cross-platform development

1. INTRODUCTION

The increased use of smartphones has changed how people search for health and nutrition content. As nutrition tracking applications are becoming more numerous, many of them face critical limitations such as the need for persistent online access, non-user-friendly interface designs, lack of personalization benefits or inappropriately supported accessibility. These limitations prompt the design of an offline-enabled user-centric diet management approach.

Food Facts App is a mobile application created in Flutter, integrated with SQLite to ensure the app functions without internet connectivity addressing these gaps.

Users can search a local food database, scan product barcodes, log meals, set dietary goals, and receive allergen alerts — all from a single intuitive interface. The system supports two roles: general users seeking nutritional insights, and administrators managing the food database through an analytics dashboard.

This paper is organized as follows: Section 2 reviews related systems and their limitations. Section 3 presents the proposed system and its objectives. Section 4 details the system specifications. Section 5 describes the architecture, design, and module structure. Section 6 covers testing and evaluation. Section 7 outlines implementation and deployment. Section 8 discusses conclusions and future work.

2. RELATED WORK

Current nutrition apps like MyFitnessPal, Cronometer and Nutritionix allow access to food databases, meal logging and calorie tracking functionalities while some include barcode scanning and cloud-based synchronization. These platforms allow users to log calories, macronutrients, and vitamins (and typically sync with fitness trackers).

But severe common limitations still exist within these systems. Most of the application will also use direct access to Database via Internet query which results in restricting us with less area coverage. Complex navigation hierarchies harm non-technical usability.

There is a broad spectrum in terms of security mechanisms, with some applications using basic encryption that gives ground to data privacy concerns. Accessibility designs are poorly implemented as well, with inconsistent

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

support for features that would assist a screen reader or would otherwise allow users with disabilities to use the software. Some of the shortcomings these technologies were designed to combat are directly mitigated within The Food Facts App through a lightweight offline-first architecture backed by SQLite, a distinctly green-themed Material Design UI, HTTP(S) SHA-256 authentication with credentials never sending plaintext over the wire and extensive accessibility support validated with TalkBack & Voice Over assistive technologies.

3. PROPOSED SYSTEM

3.1 Objectives

The Food Facts App is developed to achieve the following objectives:

- Provide offline access to a comprehensive nutritional database using SQLite.
- Enable barcode scanning for instant retrieval of packaged food nutritional data.
- Support personalized dietary recommendations and goal tracking.
- Deliver allergen alerts to protect users with dietary sensitivities.
- Provide a role-based administrative dashboard for food database management.
- Ensure cross-platform compatibility for Android and iOS through a single codebase.

3.2 System Overview

The application is built on Flutter, a UI toolkit enabling a single codebase for both Android and iOS. The backend relies entirely on SQLite for offline data management, eliminating server infrastructure requirements. The system architecture follows the MVC (Model-View-Controller) pattern to ensure modularity and maintainability. State management is handled through the Provider package, minimizing unnecessary widget rebuilds and ensuring responsive real-time updates.

4. SYSTEM SPECIFICATIONS

4.1 Hardware Configuration

Development and testing were conducted on hardware comprising an AMD Ryzen 5 processor, 8 GB RAM, Windows 10 operating system, and 320 GB storage. This configuration supported simultaneous operation of Android Studio, device emulators, and SQLite query tooling with minimal latency.

4.2 Software Configuration

The software stack architecture involves different layers such as the frontend, backend, and third-party integration layer. In the frontend, the programming language used is Dart alongside the Flutter framework to develop the application using Material Design and Cupertino widgets. The state management approach utilized includes Provider and Riverpod. In the backend, SQLite is used through the sqflite plugin to perform CRUD actions and user authentication with SHA-256 encryption. Among the third-party integrations include flutter_barcode_scanner for UPC code scanning, flutter_charts for visualization of charts, Lottie for lightweight animations using JSON files, and flutter_local_notifications for alert notifications regarding goals and allergens. Testing was carried out on Samsung Galaxy A50 (Android 11), iPhone 12 (iOS 16), Android Emulator (API 30), and iOS Simulator.5. System Design and Architecture

5.1 Database Design

The SQLite database design has five relational tables that include Users, Food_Items, Meal_Log, Goals, and Notification_Log. Users table holds user authentication credentials and has their passwords stored in SHA-256 encrypted format and whether their role is an 'admin' or 'user'. Food_Items table contains nutrition data for foods, which are stored in JSON format. Foreign keys to Users and Food_Items tables from the other two tables ensure referential integrity. The use of indexes for the 'user_id' and 'food_id' fields ensures that responses will be returned in less than 50 ms even if there are more than 10,000 records in the database.

5.2 Module Architecture

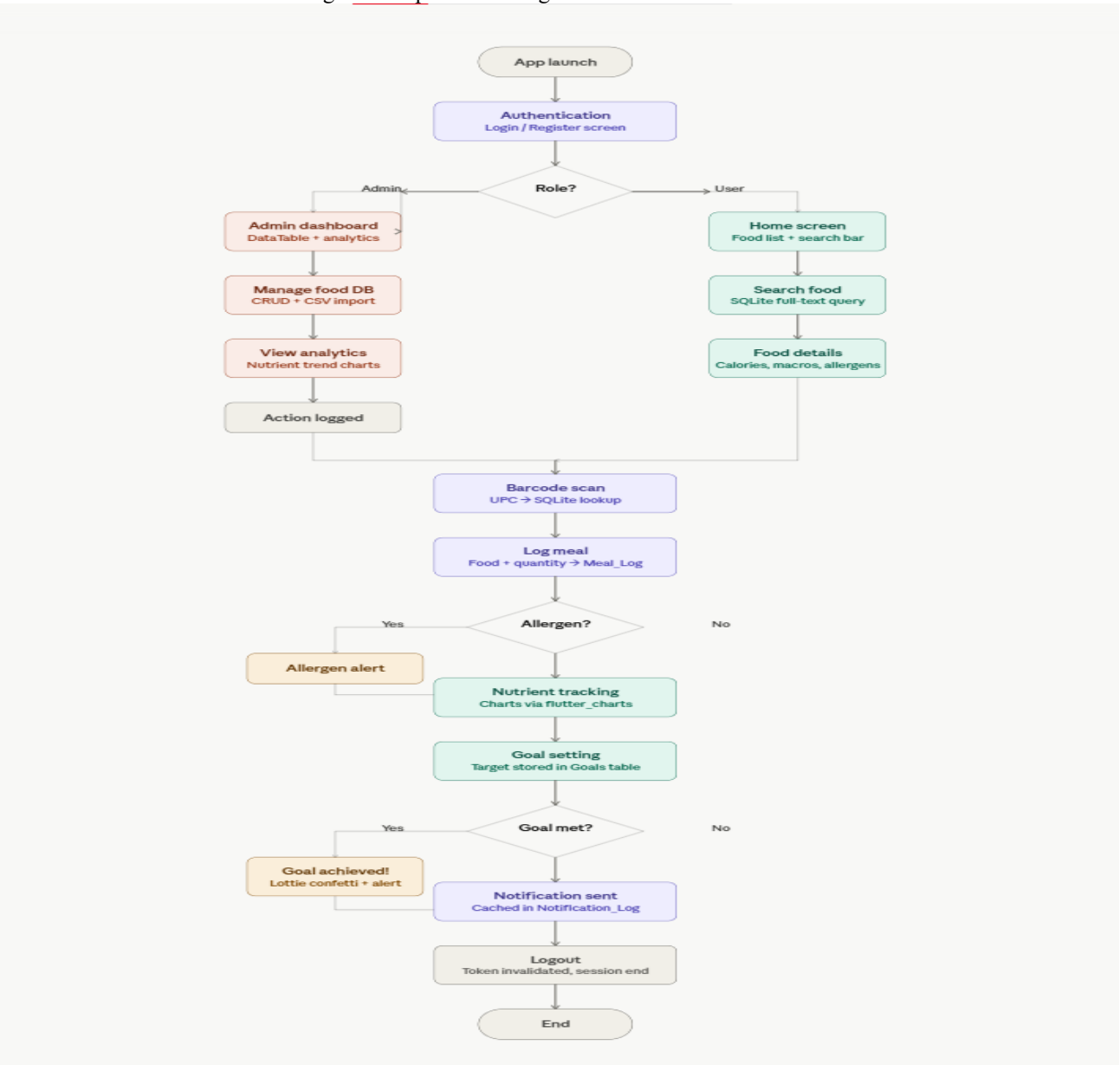
The app can be divided into five major functional modules:

Authentication Module: Helps in user registration and login processes with access control based on roles. Password validation is carried out on the client side using regular expressions and then the password is hashed using SHA-256 and saved in the Users table. Tokens are saved in flutter_secure_storage with 30 minutes expiration period. Offline login is made possible using cached credentials in SQLite.

User Module: This module provides basic functionality for food tracking such as search (using full-text SQLite queries), bar-code reader to look up the food item instantly, meals logging, goals settings, progress tracking with charts. Allergen warning appears in real-time using SnackBar notification.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

Admin Module: Provides dashboard for admins with functionalities such as CRUD actions for managing the food database (DataTable view), batch CSV import feature and visualization of the nutrients trends using analytics charts. All admin actions are logged in Activity_Logs table.
Notification Module: Used for sending goal achievement notifications, allergens warnings and updates from the administrator using flutter_local_notifications. Push notifications are logged into the Notification_Log table and will be sent out again after internet connection restoration using queue-based approach.
UI/UX Module: Uses material design and implements the green-themed



5.3 Security Design

The application implements several layers of security. Authentication employs SHA-256 hashing for password storage and flutter_secure_storage for encrypted session token management. All database interactions use parameterized queries to mitigate SQL injection risks. SQLite backups are encrypted with AES-256 and stored via path_provider. OWASP ZAP scans and penetration testing confirmed no exploitable vulnerabilities in the application's data access patterns.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

6. TESTING AND EVALUATION

A comprehensive testing strategy was executed across seven dimensions: unit, integration, security, performance, usability, compatibility, and scalability testing. All tests were implemented using flutter_test and integrated into a CodeMagic CI/CD pipeline.

6.1 Unit and Integration Testing

Unit tests validated individual components including authentication logic, food search filtering, barcode parsing, nutrient calculation, and CRUD operations. Mock SQLite databases using sqflite_common_ffi were used to isolate components. Integration tests confirmed correct data flow between modules: authentication to home screen navigation, meal logging triggering allergen notifications, barcode scan results populating the Meal Log, and admin CRUD operations reflecting in real-time charts. A combined test coverage of 95% was achieved across all modules.

6.2 Performance Testing

Performance benchmarking using Flutter Dev Tools yielded the following results: SQLite food search queries returned results in under 50ms for datasets exceeding 10,000 records; barcode scanning achieved data retrieval under 100ms; interactive chart rendering completed in under 150ms for 5,000+ data points; UI frame rate consistently maintained 60fps on low-end devices with 2GB RAM; battery consumption was measured below 5% drain per hour during active use.

6.3 Usability and Accessibility Testing

Usability evaluation involved 20 participants with diverse age groups and technical backgrounds. Ninety percent rated the interface intuitive. User feedback identified complex goal-setting inputs as an area for improvement; this was resolved by replacing manual entry fields with Numeric Up Down widgets. Accessibility validation confirmed full compatibility with Talk Back (Android) and Voice Over (iOS), high-contrast text rendering, and minimum 16px font sizes throughout the application.

6.4 Scalability Testing

Scalability tests simulated 5,000 concurrent users and datasets of 50,000+ food records. SQLite transactions maintained data consistency under concurrent access, query times remained below 100ms, and batch CSV imports of 1,000+ records completed within 500ms. List View .builder with lazy loading successfully handled rendering of 10,000+ food items without performance degradation.

7. Implementation and Deployment

The application was developed iteratively using Flutter's hot reload capability for rapid UI prototyping. The build process generated a 15MB release APK using flutter build apk --split-per-abi and a 20MB IPA for iOS using Xcode. Deployment targets included the Google Play Store and Apple App Store, with pre-release beta testing conducted via TestFlight for iOS.

The SQLite database was pre-populated with 100 food items at initialization, with batch import functionality supporting dataset expansion. Code Magic CI/CD automated build generation and test execution on each commit. Crashlytics integration provides real-time crash monitoring with a 24-hour hotfix SLA. The maintenance plan includes weekly automated encrypted backups, monthly feature updates with schema migrations via sqflite's onUpgrade callback, and quarterly accessibility audits.

8. RESULTS AND DISCUSSION

The Food Facts App successfully demonstrates that a fully offline, cross-platform mobile health application can deliver competitive performance and usability without cloud infrastructure. The offline-first SQLite architecture eliminates internet dependency while maintaining sub-50ms query performance. The 95% test coverage and 60fps rendering confirm system reliability across diverse device configurations.

The dual-role design separating user-facing features from administrative management enables clean modular boundaries and simplified security enforcement. The role-based authentication with SHA-256 hashing and parameterized queries addresses the data privacy concerns identified in existing applications. The green-themed accessible UI with Lottie animations achieved 90% positive usability ratings, indicating strong user acceptance.

Compared to existing applications, the Food Facts App demonstrates superior offline capability, faster local query performance, more granular allergen alerting, and more consistent accessibility support. The key trade-off is the absence of real-time cloud-synchronized nutritional databases, which limits the freshness of the food dataset compared to server-backed applications.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

9. FUTURE ENHANCEMENTS

Multiple avenues for further improvement can be identified. AI-based diet suggestions powered by TensorFlow Lite can facilitate the use of on-device machine learning algorithms to offer tailored food choices according to the behavioral pattern of users. AR-based food visualization through ARKit and AR Core can help users see nutritional information overlaid onto their physical food using their device's camera. Cloud-based syncing through Firebase can make it easier for multiple devices to access the same data while maintaining the ability to work offline. Multi-language localization of the app through the intl package in other languages like Hindi and Tamil can help reach a wider audience. Integration of wearable devices through Fitbit can help correlate fitness levels with food choices.10. Conclusion

This paper presented the Food Facts App, a Flutter-based mobile application that addresses critical limitations of existing nutritional tracking systems through an offline-first architecture, robust security design, personalized dietary management features, and comprehensive accessibility support. The application achieved 95% code coverage, sub-100ms query response times, and 60fps rendering across Android and iOS platforms, demonstrating both technical reliability and user-experience quality.

The MVC architecture with SQLite integration provides a scalable, maintainable foundation for future enhancements including AI-driven recommendations, AR previews, and cloud synchronization. The Food Facts App represents a practical, deployable contribution to mobile health technology, empowering users to make informed dietary decisions through an accessible, engaging, and reliable platform.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the guidance of Dr. D. Suryaprabha and the Department of Computer Science, Nehru Arts and Science College, for their support throughout the development and evaluation of this project.

REFERENCES

[1] Flutter Team. (2024). Flutter for Cross-Platform Development. O'Reilly Media.
[2] Date, C. J. (2022). Database Systems: A Practical Approach. Pearson Education.
[3] Dart, G. (2023). Dart Programming Language Specification. Google Press.
[4] Smith, J., & Lee, K. (2024). Advancements in Mobile E-Commerce. Journal of Mobile Computing, 12(3), 45–60.
[5] Patel, R. (2023). Optimizing Offline Data Storage. International Journal of Database Management, 8(2), 22–35.
[6] Flutter Documentation. https://flutter.dev (Accessed January 2025).
[7] SQLite Documentation. https://sqlite.org (Accessed January 2025).
[8] Material Design Guidelines. https://material.io (Accessed February 2025).