Main Application Architecture

The CoreFoods application will be developed as a native Android application with a clear separation between the user interface, application logic, and data storage layers. This layered approach helps keep the project well organised, easier to maintain, and simpler to test, which is particularly important for a group-based software project. Separating responsibilities also allows different team members to work on different areas of the system at the same time without causing unnecessary conflicts.

At a high level, the application architecture is divided into three main layers:

Presentation Layer (User Interface)

Application Logic Layer

Data Layer (Local Storage and External Services)

Each layer has a clearly defined role and communicates with the other layers in a controlled way.

Presentation Layer (UI)

The presentation layer is responsible for everything the user sees and interacts with within the app. This includes:

Screens for registration, login and user profile management.

The main dashboard displaying daily calorie balance, progress bars and key statistics.

Screens for logging exercise activity and food intake.

The AI chatbot screen where users can type questions and read AI-generated responses.

These screens will be implemented using Android activities and fragments, along with XML layout files. The main purpose of the presentation layer is to collect user input and display results, rather than performing calculations or making decisions itself.

To support good software design, the UI will remain as “dumb” as possible. This means it will pass user actions (such as button presses or form submissions) to the application logic layer and then display whatever data it receives in return. Keeping logic out of the UI helps reduce duplication, makes the interface easier to update, and improves overall testability.

Application Logic Layer

The application logic layer contains the core behaviour of the CoreFoods app. This layer is responsible for processing data, applying business rules, and preparing information for both the user interface and the AI features.

Key responsibilities of this layer include:

Calculating calories burned from exercise entries based on user inputs such as exercise type, duration and intensity.

Calculating calorie intake from logged meals and comparing it to the user's daily calorie target.

Determining whether a user is in a calorie deficit, surplus or at maintenance for a given day.

Generating summaries of user activity and nutrition data (for example, daily or weekly summaries).

Preparing structured data that can be safely sent to the AI system, such as "today's overview" or recent progress trends.

Enforcing validation and business rules, such as preventing invalid values (e.g. negative exercise duration) and ensuring required fields are completed before data is saved.

This layer will be implemented using separate classes or modules that can be reused across the app. By centralising calculations and logic here, we avoid duplicating code across multiple screens and make it easier to test individual components in isolation.

Data Layer

The data layer is responsible for storing, retrieving, and updating all information used by the application. It acts as the single source of truth for the app's data and ensures consistency between features such as tracking, analytics, and AI interaction.

The data layer is split into two main parts:

Local Storage (SQLite Database)

All core app data, including user profiles, exercise logs, food logs, and historical progress data, will be stored locally using an SQLite database.

Data access will be handled through a structured set of data access functions or a repository-style approach, so that the rest of the app does not need to interact directly with SQL queries.

This approach allows the app to function even without an internet connection, at least for logging data and viewing previous records.

Using local storage improves reliability and responsiveness while keeping the architecture relatively simple and suitable for the scope of the project.

External Services (Authentication and AI API)

User authentication and notifications may be handled using external services such as Firebase, in line with the project’s resource plan.

The AI chatbot feature will communicate with an external AI API over a secure HTTPS connection.

A dedicated module will manage all communication with the AI service, meaning the rest of the application only needs to send structured requests and receive responses.

This separation ensures that external dependencies are isolated from the rest of the system, making it easier to update or replace them in the future if requirements change.

Overall Application Flow

A typical flow through the CoreFoods architecture will follow these steps:

The user interacts with the user interface, for example by logging a meal, adding an exercise session, or opening the dashboard.

The presentation layer passes this input to the application logic layer for processing.

The application logic layer performs calculations, validation, or data preparation as needed.

The data layer is accessed to store new information or retrieve existing data from local storage or external services.

The processed result (such as updated calorie totals, progress statistics, or AI advice) is returned to the presentation layer and displayed to the user.