Solution Architecture Model for MyMacroTracker

Kartikeya Arvind Yadav — kay54

May 2025

The architecture of *MyMacroTracker* is intended to promote rapid development, efficient scaling, and long-term maintainability. The system uses a modular monolith design for the backend, with FastAPI as the main framework. The backend is logically divided into modules for authentication, user profile management, meal tracking, and AI integration. These modules are initially deployed as a single service, but are designed in such a way that allows them to be separated into independent services down the road as the platform grows.

The frontend here is done using React Native for mobile and React for web, providing the opportunity to share code and thus lessen development effort, while maintaining consistency across platforms. This resolves the time-to-market versus cost-of-development balance, which proves especially important for an MVP.

The backend connects to an AI API (such as OpenAI) to generate individual meal suggestions. This prevents us from going through the expenses of developing AI from scratch and instead allow us to provide these advanced features at a very minimal infrastructure cost. Auto-syncing of fitness data occurs through APIs like Google Fit and Apple Health, thereby giving real-time elevation or alteration to the nutrition plans of the users.

Whenever there is a need for structured information storage, PostgreSQL is employed and maintained for its strong consistency, with support for complex queries. In case of a data-tier failure, the architecture is geared for automatic backup and recovery within an acceptable time frame of few minutes, depending on hosting provider configurations.

This architecture supports key quality attributes:

- **Cost Efficiency**: Leveraging third-party services and shared frontend reduces dev and operational costs.
- **Scalability**: Modular backend design and async API support make it easy to scale horizontally as usage grows.
- **Resilience**: Structured data storage, API separation, and modular logic improve reliability and error isolation.
- **Security**: OAuth 2.0, encrypted data handling, and limited AI data sharing maintain user privacy.

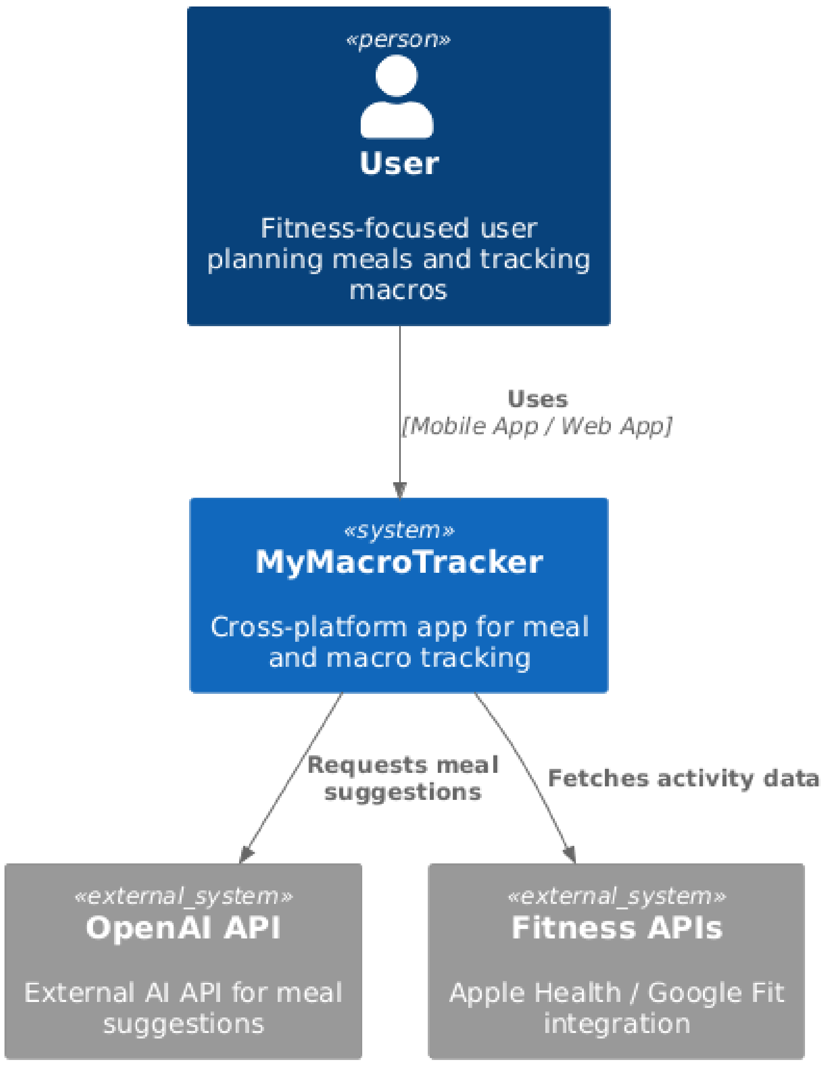
«person»
User
Fitness-focused user planning meals and tracking macros
Uses
[Mobile App / Web App]
«system»
MyMacroTracker
Cross-platform app for meal and macro tracking
Requests meal suggestions
Fetches activity data
«external_system»
OpenAI API
External AI API for meal suggestions
«external_system»
Fitness APIs
Apple Health / Google Fit integration

Figure 1: System Context Diagram

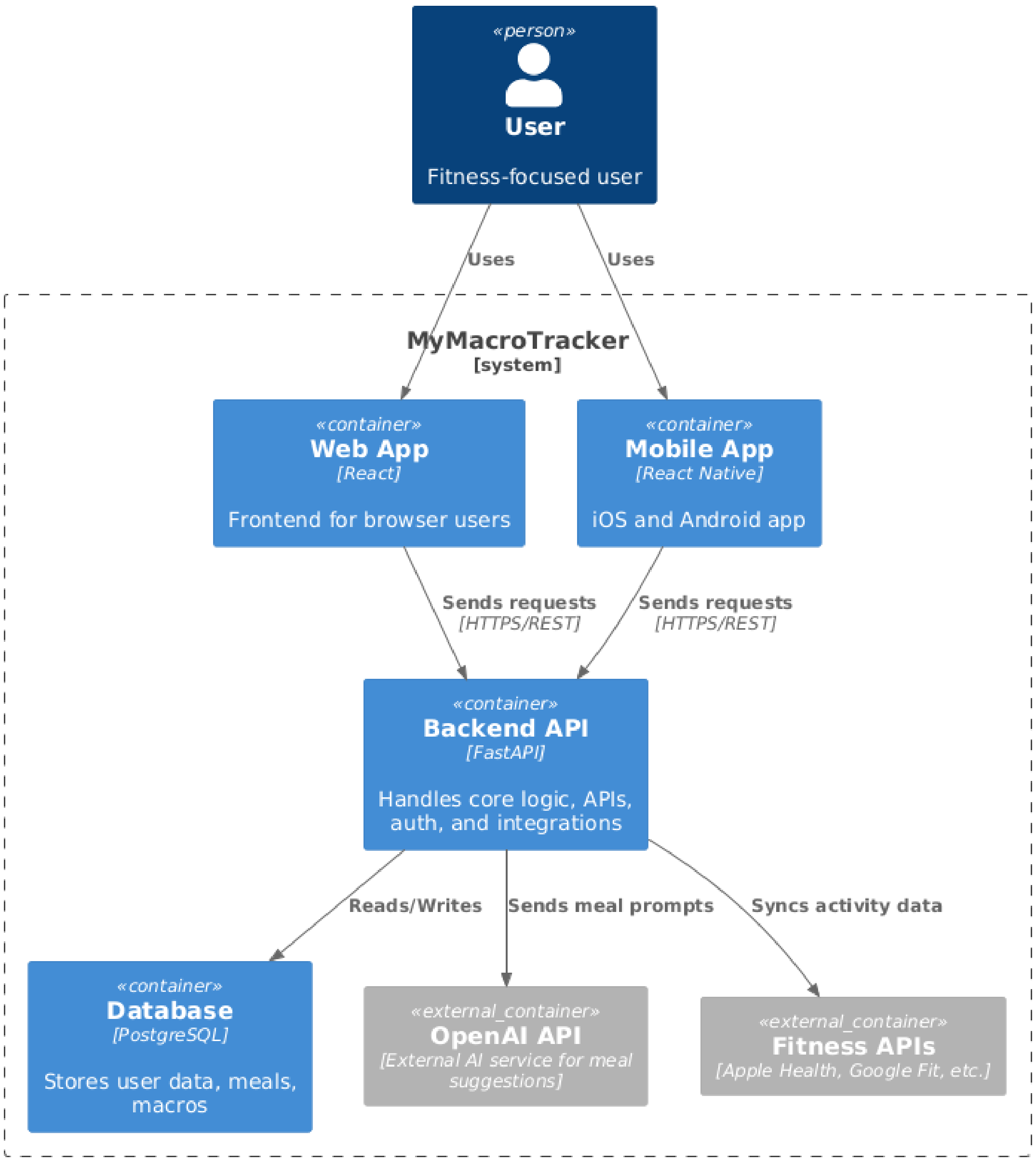

Figure 2: Container Diagram

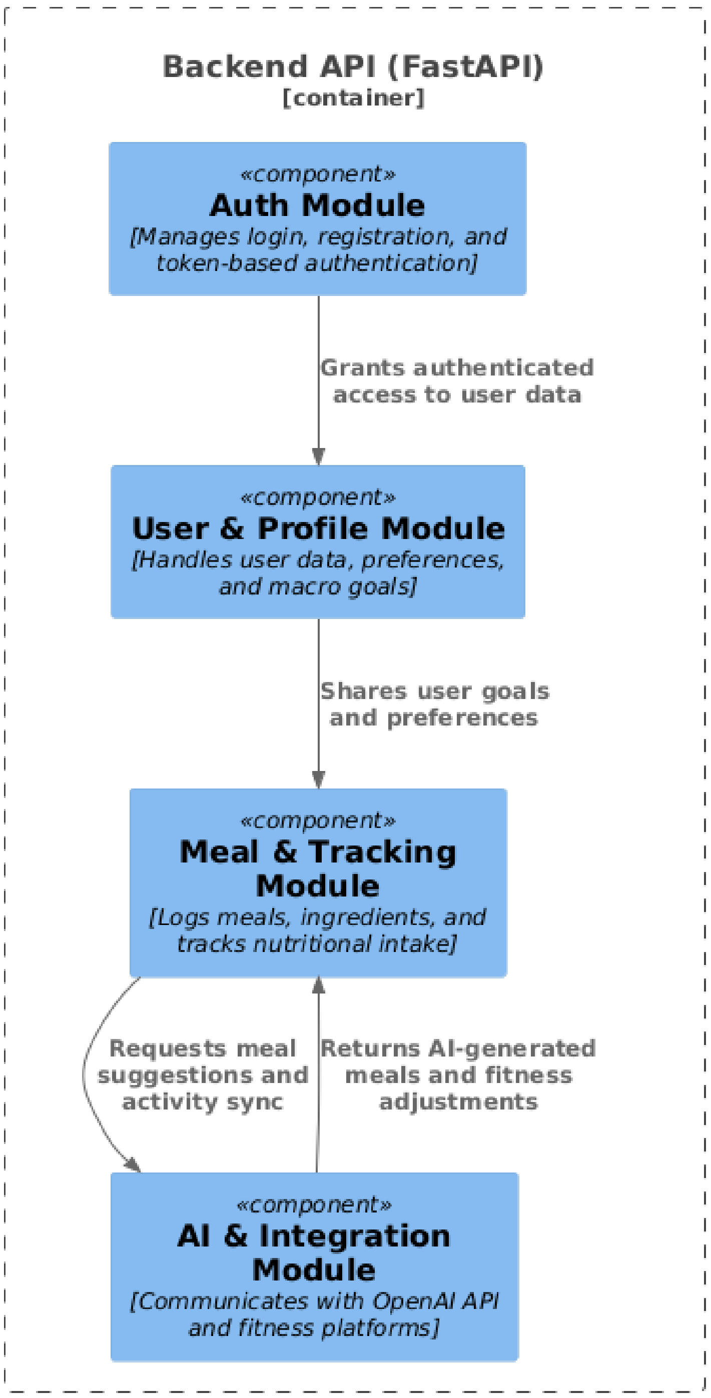
Backend API (FastAPI)
[container]
«component»
Auth Module
[Manages login, registration, and token-based authentication]
Grants authenticated access to user data
«component»
User & Profile Module
[Handles user data, preferences, and macro goals]
Shares user goals and preferences
«component»
Meal & Tracking Module
[Logs meals, ingredients, and tracks nutritional intake]
Requests meal suggestions and activity sync
Returns AI-generated meals and fitness adjustments
«component»
AI & Integration Module
[Communicates with OpenAI API and fitness platforms]

Figure 3: Component Diagram - Backend API