Politecnico di Torino

Masters's Degree
in Computer Engineering

Masters's Degree Thesis

Development and Implementation of a Hybrid Rule Engine for Nutritional and Lifestyle Recommendations

Supervisors
prof. Maurizio Morisio

Candidate
Angelo Bisignano

DECEMBER 2025

1. Introduction

1.1 The Paradigm Shift in Wellness and the Challenge of Personalisation

The relentless evolution of technology and medical science has represented a historic milestone, significantly raising global average life expectancy. However, contemporary awareness has evolved beyond the mere numerical measurement of years, shifting the focus to enhancing their quality. The concept of **"healthy longevity"** places emphasis on targeted nutrition and a balanced lifestyle, recognising these elements as the fundamental pillars for maintaining optimal well-being over time.

This awareness, when considered in conjunction with the substantial and frequently unorganised volume of information available online, has resulted in a distinct necessity: the development of digital tools that can assist users in a **proactive and customised** approach to managing their well-being [1]. Despite the saturation of the health app market, the majority of these applications are confined to basic data tracking functions, such as monitoring food intake or sleep duration. The critical challenge, which remains unresolved, lies in converting this raw data into **actionable, sophisticated, and truly personalised advice**, a gap this thesis seeks to bridge.

1.2 Project Objective: The utilisation of a Rule Engine in the context of Lifestyle Consulting

The primary objective of this project was to address this lacuna by designing and implementing an **intelligent system** capable of providing high-level, multi-faceted wellness recommendations. The initial phase was dedicated to the essential task of **knowledge codification**: translating a vast array of nutritional and behavioural guidelines into a manageable, dynamic, and readily updatable format.

This requirement resulted in the implementation of a **Rule-Based Engine** developed in Python. The fundamental innovation resides in the system's **decoupled architecture**. In contrast to rigid, hard-coded approaches, the decision-making logic of the system is kept external: the rules are modelled using a high-level **Python GUI**, which allows experts to translate their instructions into a standardised, persistent logical structure in **JSON files**. This separation of the algorithmic core from the knowledge base ensures intrinsic modularity and **transparency**, allowing recommendations to be rapidly updated without requiring any changes to the engine's core code.

1.3 System Evolution and Integration

The system has rapidly evolved its capabilities, moving beyond mere nutritional analysis to become a fully-fledged **lifestyle advisor**. The successful incorporation of the **Sleep, Activity and Mood Recommendations modules** was a pivotal step, augmenting the system's analytical capabilities to encompass intricate patterns, correlating diet, circadian rhythm, and sleep efficiency [2]. This extension validated the flexibility of the core Rules Engine design.

The ultimate effectiveness of this recommendation engine is intrinsically linked to its ability to integrate into the user's digital ecosystem. The project advanced significantly with the pivotal transition to an **accessible backend service**. Following an initial phase in which input data was managed via files extracted from third-party systems (e.g., FatSecret), the entire architecture was reengineered for robust, real-time interaction. Currently, data is pulled directly from **Firebase**, which serves as the **central authoritative repository** for user logs. Communication with the final **frontend application**—developed in parallel by another graduate student—is ensured through a set of dedicated **RESTful APIs**, positioning our engine as a central computational service that provides timely and consistent data flow to the mobile environment.

1.4 Bridging the gap with explainable knowledge-driven approaches

Contemporary digital health applications increasingly rely on opaque, data-driven models that prioritize predictive performance but often lack interpretability and clinical traceability [3]. This thesis positions itself at the intersection between explainable AI and practical digital health by proposing a knowledge-driven framework that encodes expert rules in a maintainable and auditable format. Unlike black-box recommenders, the presented solution prioritizes clinical safety, transparency and maintainability, thereby enabling rapid updates to clinical guidelines and immediate traceability of every generated recommendation [4]. This characteristic is particularly relevant in medical and lifestyle domains where accountability and user trust are essential.

2. Analysis and Design of the Recommendation System

2.1 State of the Art of Nutritional and Lifestyle Recommender

The current State of the Art is dominated by **Nutritional Recommendation Systems (NRS)**. An NRS is formally defined as a software tool designed to guide users in selecting food items, recipes, or dietary plans that align not only with their stated **preferences** but also with their specific **nutritional constraints** and **health goals**. Functioning at the intersection of Information and Communication Technology (ICT), Artificial Intelligence (AI) and clinical science, these systems aim to bridge the gap between simple data logging and true personalised intervention. The ongoing evolution of NRS is driven by the need to address the inherent complexity of dietary planning, where personal preference must yield to non-negotiable health requirements (e.g. allergies and chronic diseases).

2.1.1 Classification of NRS Methodologies and Challenges

The development landscape of NRS can be broken down according to the primary methodology used to generate advice. The choice of methodology critically impacts the system's **accuracy, scalability, and transparency** [5]. See Table 1.1.

Knowledge-Based System

These systems leverage an explicit and formalised **knowledge base** (rules, ontologies, established nutritional guidelines) to generate recommendations. They do not rely primarily on the collective behavior of other users or historical rating data, making them the preferred choice for clinical accuracy.

There are two subtypes:

- **Rule-Based Systems**: They use **IF-THEN** rules defined by experts (e.g., "IF sodium intake exceeds X for 5 consecutive days, THEN recommend reducing salt"). The user always knows *why* a suggestion was made, which is crucial for building trust in health applications.
    - **In-Depth:** RBS are highly deterministic and **auditable**. This makes them exceptionally strong for enforcing **hard constraints** (e.g., allergies, severe

chronic diseases). The user always knows *why* a suggestion was made, which guarantees high **explainability** and builds trust in health applications.
    - **Limitations:** They suffer from **low adaptivity** as they require manual updates whenever clinical guidelines change, and they typically do not excel at discovering new or novel food items for the user.
    -

- **Ontology-Based Systems**: These utilise a formal conceptual model (an ontology) to represent the complex relationships between foods, nutrients, health conditions, and goals. This structure enables powerful **semantic reasoning**, allowing the system to deduce non-obvious relationships in the data.
    - **In-Depth:** Ontologies enable powerful **semantic reasoning**. They can infer non-obvious relationships (e.g., linking a high glycemic index food to diabetes risk) and are structurally designed for complex safety checks.
    - **Limitations:** They require extensive **ontology engineering** (a labor-intensive process) making knowledge scalability potentially low-to-medium.

Collaborative Filtering (CF)

They generate recommendations based on similarities between user preferences (user-based) or similarities between items (item-based). While effective, they are less common for pure health recommendations because the advice is driven by *taste satisfaction* rather than clinical necessity. Furthermore, CF systems struggle with the Cold Start Problem (difficulty recommending to new users or new items).

- **In-Depth:** CF systems are highly adaptive and scale very well with data volume, making them performant in terms of prediction.
- **Limitations:** They exhibit **low explainability** due to reliance on opaque latent factors. More critically in the health domain, CF may recommend **unsafe items** if similar users enjoyed them, thus offering low clinical safety. They also require large datasets of users/ratings, leading to the notorious **Cold Start Problem** for new users.

Content-Based System

They recommend items (recipes or foods) that are structurally similar to those the user has consumed previously or that match their profile (age, BMI, goals). They compare the characteristics of a recipe (ingredients, calories, macronutrients) with the user's desired nutritional profile. For example, if the user often eats protein-rich foods, they will be recommended other protein-rich recipes.

- **In-Depth:** Explainability is **medium** as it relates to feature-level explainability (e.g., "We recommend this because it has similar protein content to your favorites"). CBS adapts well to specific user profiles and is highly scalable with content automation.
- **Limitations:** The system's effectiveness largely depends on the **correctness of the feature profile** defined for the food items.

Hybrid System

The dominant trend in the current State of the Art is the adoption of Hybrid Systems. These approaches combine two or more methodologies (often Knowledge-Based for constraints and ML/CF for preference matching) to mitigate the limitations of a single method. This strategic combination ensures both the clinical safety (guaranteed by rules) and the user acceptance (guaranteed by preference learning), achieving a superior balance between efficacy and user experience.

- **In-Depth:** This approach offers **high clinical safety** because the Rules-Based layer is used as **safety guardrails**. The ML component is delegated to tasks related to preference learning (e.g., taste prediction), ensuring high adaptivity. The combination achieves a superior balance between efficacy and user experience.
- **Architecture:** Hybrid systems often separate tasks, with the **KBS enforcing constraints** and the **ML layer prioritizing options** that respect those constraints. The explainability is generally medium-to-high, depending on the transparency of the ML component used. This complexity is scalable because tasks can be delegated to the ML layer, while the safety rules remain manageable.

This thesis explicitly falls within the Knowledge-Based Systems category, adopting a rule-based architecture.

Approach	Explainability	Adaptivity	Scalability (knowledge)	Data requirement s	Clinical safety
Rule-Based Systems	High — explicit rules and provenance	Low — manual update required	Medium — manageable but grows linearly with rules	Low — can operate with limited historical data	High — deterministic, auditable decisions
Ontology-Base d Systems	High — semantic reasoning	Medium — can infer via ontology	Low–Medium — needs ontology engineering	Low–Mediu m	High — formal semantics support safety checks
Content-Based	Medium — feature-level explainability	Medium — adapts to content profile	High — scalable with automation	Medium	Medium — depends on feature correctness
Collaborative Filtering (CF)	Low — latent factors opaque	High — learns from user behavior	High — scales well with data	High — needs many users/ratings	Low–Medium — may recommend unsafe items

Hybrid (Rule + ML)	Medium–High — rules + ML explainability layers	High — ML layer increases adaptivity	High — can delegate tasks to ML	High	High — rules act as safety guardrails

Table 1.1

2.1.2 Customization and context dimension

The effectiveness and modernity of a Nutritional Recommendation System (NRS) is measured not only by the algorithmic methodology employed, but also by the depth and breadth of its personalisation capabilities. Recent research in recommender systems shows that true personalization emerges from the intersection of **data-driven learning**, **contextual awareness**, and **domain-specific constraints** [6].

From General Recommender Systems to Health-Oriented Adaptations

Traditional recommender systems were initially designed for domains such as **e-commerce**, **music**, or **movies** (e.g., Amazon, Spotify, Netflix). In these contexts, the primary goal is **preference optimization** (predicting what a user will like based on behavioral history or the preferences of similar users). Success metrics typically include **accuracy**, **precision**, and **user satisfaction**.

When these paradigms are adapted to **health and well-being**, several paradigm shifts occur:

- **Safety and ethics override preference.** In wellness domains, recommendations must ensure clinical safety (e.g., avoiding allergens or excess sodium), even if they conflict with user taste or popularity trends.
- **Explainability becomes essential.** In contrast to opaque collaborative filtering used in entertainment systems, health-oriented systems must provide traceable rationales, ensuring user trust and clinical auditability [7].

- **Data sparsity and multimodality.** Health data integrate multiple heterogeneous sources (sleep, nutrition, physical activity, biomarkers), often collected intermittently. This requires hybrid reasoning rather than pure statistical learning [8].
- **Outcome-oriented personalization.** Unlike entertainment RS, which optimize for engagement, health systems aim to improve measurable outcomes such as body composition, glycemic control, or sleep regularity.

This transition marks the evolution from **preference-driven personalization** to **goal- and safety-driven personalization**, redefining how customization and context are interpreted in wellness systems.

Personalized Nutrition

Advanced systems transcend traditional calorie-counting approaches. **Personalized nutrition** now entails a multi-dimensional understanding of user physiology and lifestyle, achieved through:

- **Macronutrient and Micronutrient Balance:** Beyond calorie tracking, recommendations focus on nutrient quality and proportionality relative to physiological goals (e.g., muscle hypertrophy, endurance optimization) [9].
- **Dynamic Adaptation:** Temporal models adjust recommendations in real time based on behavior, such as training sessions, sleep quality, or recovery metrics.
- **Feedback Loops:** Continuous learning enables systems to refine suggestions using user input or biometric data, forming adaptive feedback mechanisms.

Rule-based personalization leverages **explicit logical conditions** (IF-THEN structures), ensuring predictable and auditable decisions aligned with medical guidelines.

Holistic integration (Lifestyle)

Well-being is a multi-dimensional concept; diet cannot be treated in isolation. Contemporary systems evolve toward **Lifestyle Recommendation Systems (LRS)** that integrate complementary behavioral dimensions:

- **Sleep and Recovery:** Systems incorporating sleep data can adjust caloric and macronutrient targets based on circadian regularity and recovery status.

- **Physical Activity:** Integration with wearables (e.g., Google Fit, Apple Health, or Health Connect) enables dynamic energy recalibration.
- **Stress and Mood:** Experimental models incorporate emotional or hormonal indicators to detect stress-induced eating patterns, aiming to balance physiological and psychological wellness [10].

This holistic paradigm establishes a bidirectional relationship between nutrition and lifestyle, making recommendations more adaptive, sustainable, and personalized.

Contextualization

For recommendations to be effective, they must be **contextually actionable** in the user's daily environment. Contextualisation operates on several interconnected levels:

- **Behavioral Patterns:** Systems analyze temporal trends (e.g., habitual snacking, recurring late dinners) to address long-term habits rather than isolated behaviors.
- **Temporal Context:** Time-aware algorithms adjust meal suggestions based on circadian rhythms or work schedules.
- **Preference-Aware Constraints:** Even within rule-based systems, personalization considers taste adherence and feasibility, offering nutritionally equivalent but culturally acceptable alternatives.

Contextualization bridges **data interpretation** and **behavioral translation**, ensuring that clinically sound advice remains practical and engaging in real life.

Summary: From Personalized to Precision Wellness

In summary, the evolution of recommender systems within health and wellness domains represents a progression from:

1. **Preference-based filtering** → toward goal-oriented reasoning
2. **Static personalization** → toward contextual and adaptive modeling
3. **Single-domain focus (food)** → toward multi-domain lifestyle orchestration

The present thesis adopts this modern perspective, implementing a knowledge-based, rule-driven architecture designed for high explainability, clinical safety, and contextual adaptability.

This structure enables the integration of user data (nutrition, activity, sleep), laying the foundation for a Precision Wellness Recommendation Engine.

2.1.3 Structure and Deployment

The state of the art is strongly oriented towards integration and mobile access:

- **Microservices/API Architecture:** Most NRSs are implemented as backend services with dedicated APIs to enable access from mobile applications, wearables, or healthcare platforms.
- **Platforms (Mobile First):** Mobile applications are the most common platform for user interaction (approximately 28% of nutritional recommendation systems are mobile, according to some systematic reviews) [11].
- **Usability and Explainability:** There is a growing focus on User Experience (UX) and Explainability (XAI), which is essential in healthcare. Rule-based systems excel in this regard, as the recommendation is intrinsically linked to the rule that generated it.

2.2 System requirements

The design phase of a recommender system is crucial for ensuring that the final product aligns with the project goals and external integration needs.

In this section, the requirements for the nutritional and lifestyle recommendation engine are formally divided into functional requirements (what the system must do) and non-functional requirements (how the system must work).

2.2.1 Functional Requirements

Functional requirements define the system's specific actions and elaboration capacities. Since the project consists of three main modules (a rules engine, a knowledge manager and an API service), the functional requirements reflect this modularity.

RF-01: User Data Acquisition from Firebase

The system must establish a secure communication channel for accessing user data. This requirement is divided into two distinct security levels:

- Client-to-API security: Access to API endpoints (e.g. /nutrition and /sleep) must always be protected. The client (i.e. the mobile application) must send a valid Firebase ID token. The API must then validate this token using the firebase_admin.auth module to authenticate and authorise the request.
- API-to-database security: After client authentication, the backend must use Firebase's internal service credentials to establish a secure, persistent connection to the database, retrieving nutrition and sleep data for the requested user.

RF-02: Time Log Analysis

The engine must be able to process raw data received from Firebase, focusing specifically on analysing long-term patterns. This is achieved by aggregating and normalising information over a defined time window (typically the previous seven days' logs) to analyse weekly patterns.

This goes beyond simple summation and includes calculating aggregate metrics such as daily averages, consumption frequency counts (e.g. 'fried foods consumed three times in a week') and consistency assessment (e.g. sleep regularity). This pre-processing phase is essential in order to provide the context variables necessary for executing complex rules.

RF-03: Rule-Based Engine Execution

The Rule-Based Engine must systematically scan the analysed log (RF-02) using the externally defined knowledge base (RF-04). Its main function is to identify all deviations (violations) from the codified nutritional and lifestyle guidelines. The engine must handle both simple rule logic, such as direct comparisons with thresholds, and complex rule logic, which may require dynamic calculations or correlations between multiple parameters. Efficient execution is required to contribute to compliance with the non-functional requirement of low latency (RNF-01).

RF-04: User Management of the Knowledge Base

A dedicated Python GUI interface is required to enable non-technical users (e.g. expert nutritionists) to interact with the system. The aim is to enable users to enter, modify, categorise and save logical rules in a standardised format (JSON). This is a crucial requirement for maintainability (RNF-04), ensuring that clinical guidelines can be updated quickly without altering the Python source code of the analysis engine.

RF-05: Recommendation Generation and Prioritization

Based on the violations detected (RF-03) by the rule engine, the system must generate a set of clear textual recommendations regarding nutrition and sleep. The generated message must be personalized by inserting calculated dynamic values (e.g. placeholder) to make the recommendation specific. Ideally, the system should also assign a priority to each recommendation before sending it to the front end, even implicitly through order or categorisation, to optimise user action.

RF-06: Exposure of the API Service for the Frontend

The backend, which is implemented in FastAPI, must expose one or more RESTful API endpoints that can accept requests from the mobile application. After authentication (RF-01) is complete, the API must call the entire analysis process (RF-02 to RF-05) and return the complete package of recommendations in JSON format to the client. This ensures that the analysis engine is treated as a scalable, real-time service.

1.2.2 Non-Functional Requirements

Non-functional requirements are essential for the success of the system, particularly with regard to performance and architectural integration.

RNF-01: Performance (Service Latency)

Due to the integration with the mobile application and the requirement to deliver a responsive user experience, the entire recommendation processing procedure (from retrieving data from Firebase to the API response) must be carried out with minimal latency.

This performance is guaranteed in part by the use of FastAPI, which is renowned for its speed of execution, and by the computational efficiency of the Python Rules Engine.

RNF-02: Data Security and Protection

As a system that handles sensitive health data, access to data on Firebase and interaction with APIs (RF-06) must be strictly controlled.

- Channel security: All API communications must be via HTTPS (a secure protocol).
- Client authentication: Interacting with the APIs (RF-06) requires providing a valid Firebase ID token with each request. The backend must validate this token for authentication purposes.
- Database Authorisation: Access to user-specific Firebase data is regulated by internal service credentials, ensuring the API can only access necessary data and respecting the principle of least privilege.

RNF-03: Engine Reliability and Robustness

The system must be able to operate reliably, even when faced with imperfect input data or connectivity issues.

- Data error tolerance: The Rule-Based Engine must tolerate non-ideal inputs (e.g. missing fields, anomalous values and incomplete sleep logs) without crashing. Pre-processing functions must include default values (e.g. get('value', 0)) to prevent runtime errors.
- Centralized Service Error Handling: To explicitly meet the reliability requirement, a custom exception system has been implemented at the API level. In the event of a connection interruption or failure (or internal logical errors) to Firebase, the API does not simply crash with a 500 Internal Server Error, but rather captures the Python exception and translates it into a standardised format.
- Shared Error Protocol: The error response is encapsulated in a JSON object that includes a unique error code (e.g. 'ERR_DB_001' for a database connection error). Implementing a global exception handler ensures these codes and informational messages are returned to the front end. Standardising the error protocol in this way is essential to enable the mobile application to immediately identify the cause of the problem (e.g. lack of data versus database unavailable) and communicate it appropriately to the user.

RNF-04: Maintainability and Upgradeability

The design must facilitate the system's ongoing evolution and maintenance by developers and administrators:

- Logic/data separation: The clear distinction between the Python execution logic and the JSON knowledge base (RF-04) is fundamental to this requirement.
- Modular Extensibility: The architecture should allow new rule categories (e.g. Physical Activity) to be added simply by creating a new JSON file and a new Python pre-processing module. This should not require any changes to the main API other than the addition of a new endpoint. This ensures a high degree of loose coupling between components.

RNF-05: Knowledge Base Usability

The effectiveness of the rule-based system should not be compromised by the complexity of managing rules:

- Intuitive interface: The Python GUI should offer a straightforward interface for entering and editing rules. It should use high-level terminology (such as aggregate metric names) that is understandable to domain experts (nutritionists), not just programmers.
- Error prevention: The interface must integrate immediate syntactic validation mechanisms (e.g. JSONDecodeError on save) to prevent malformed rules being inserted that could compromise the entire system in production.

2.3 System architecture

The nutritional and lifestyle recommendation system was built using a decoupled microservices architecture — a modern approach chosen to ensure adherence to the non-functional requirements of scalability, maintainability (RNF-04) and integration with the external mobile application. Designed according to the asynchronous client-server model, the architecture positions the recommendation engine as an exclusively API-accessible computational backend service.

2.3.1 General Architectural Diagram

The logical architecture is developed across three interconnected levels that clearly define the data flow and separation of responsibilities (See Image 2.3):

Presentation level

This level consists of the mobile application, which was developed externally and acts as a client, and the GUIs that were developed in the first part of the project:

- Mobile Application (end user): Acts as the main client, recording nutrition, sleep, mood and activity data on Firebase and displaying the final recommendations. While this component is outside the scope of this thesis, it is the ultimate goal of the API service.
- Python Graphical User Interfaces (Internal Development Tools): These GUIs were developed with Tkinter for the initial prototyping and management phases and represent the presentation layer for the administrator/developer.
    - Rule Management GUI: Used by experts for entering, modifying and maintaining the JSON Knowledge Base.
    - Analysis and Debugging GUI: Used for local testing, parsing test files (FatSecret) and visualising internal results and activated rules.

Service level

This is the computational core of the project and is implemented as a Python service application:

- RESTful API Layer (FastAPI): The service exposes endpoints via a high-performance API framework.
    - Orchestration and security: It manages authentication through Firebase token validation (RF-01) and orchestrates data flow.
    - Endpoint modularity: The main endpoints (/nutrition, /sleep, /activity and /mood) are decoupled and call their respective analysis modules.
- Rules Engine (Python): This computational module is written entirely in Python and implemented in separate modules (recommendation_food.py, recommendation_sleep.py, reccomendation_sport.py and reccomandation_mood.py).

The API calls it to perform the process in three sequential steps (as detailed in Chapter 3.3.1).

- Data Acquisition: Retrieves the weekly log from Firebase using internal service credentials.
- Analysis: Performs pre-processing on the data and compares it with dynamically loaded rules.
- Generation: Returns a structured, customised list of recommendations in JSON format.

Data level

This layer manages all forms of information persistence:

- User data (Firebase Firestore): The Firebase Firestore database stores dynamic user data such as meals, sleep, weight, mood, and activity. This choice ensures high availability and ease of synchronisation with the mobile environment, supporting requirement RNF-01.
- Knowledge Base (JSON files): The operating logic (if-then rules) is stored in JSON files rather than in the Python code. This implementation choice decouples business logic from execution logic, facilitating the maintainability and updating of the knowledge base via the dedicated GUI.

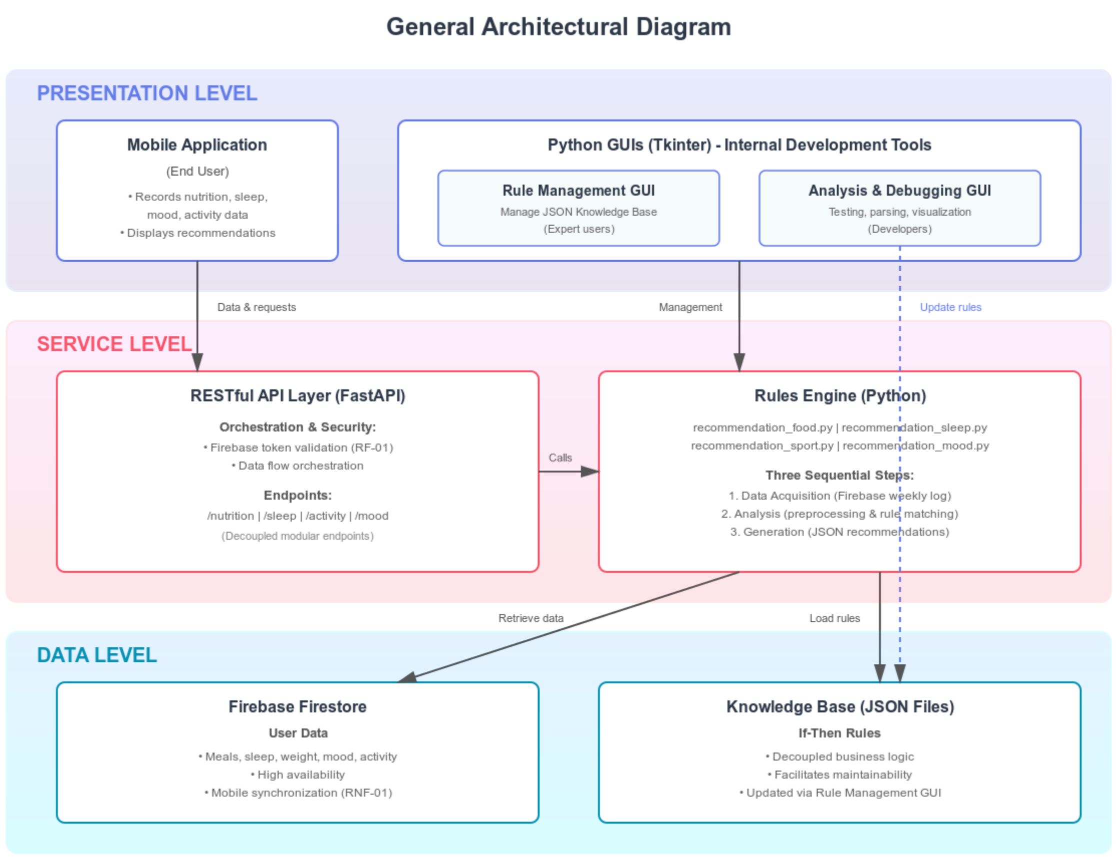
General Architectural Diagram
PRESENTATION LEVEL
Mobile Application
(End User)
• Records nutrition, sleep, mood, activity data
• Displays recommendations
Python GUIs (Tkinter) - Internal Development Tools
Rule Management GUI
Manage JSON Knowledge Base
(Expert users)
Analysis & Debugging GUI
Testing, parsing, visualization
(Developers)
Data & requests
Management
Update rules
SERVICE LEVEL
RESTful API Layer (FastAPI)
Orchestration & Security:
• Firebase token validation (RF-01)
• Data flow orchestration
Endpoints:
/nutrition | /sleep | /activity | /mood
(Decoupled modular endpoints)
Calls
Rules Engine (Python)
recommendation_food.py | recommendation_sleep.py
recommendation_sport.py | recommendation_mood.py
Three Sequential Steps:
1. Data Acquisition (Firebase weekly log)
2. Analysis (preprocessing & rule matching)
3. Generation (JSON recommendations)
Retrieve data
Load rules
DATA LEVEL
Firebase Firestore
User Data
• Meals, sleep, weight, mood, activity
• High availability
• Mobile synchronization (RNF-01)
Knowledge Base (JSON Files)
If-Then Rules
• Decoupled business logic
• Facilitates maintainability
• Updated via Rule Management GUI

Image 2.3

2.3.2 Choice of Technologies

The selection of technologies for the NRS was strategic, driven by the need to meet the functional requirements (RF) for high-level analysis while strictly adhering to the non-functional requirements (RNF) for performance, security, and maintainability. The core principle was to favor **open-source solutions** that support an agile, service-oriented architecture.

Development Language: Python

Python was chosen as the core language for the entire backend and analysis logic. Its widespread adoption in data science and scientific computing meant that robust libraries for data manipulation and mathematical analysis were available, which are essential for the pre-processing phase (RF-02).

- Rule Engine Suitability: Python's dynamic nature and clear syntax make it particularly well-suited to implementing the Rule-Based Engine (RF-03). Specifically, it facilitates the safe evaluation of conditional strings (the condition field in the JSON rules), which is critical for executing complex rule logic with high flexibility and maintainability (RNF-04).

API Framework: FastAPI

FastAPI was used to construct the API layer, defining the interface between the Python *backend* and the *frontend* mobile application. This choice was deliberate over alternatives like Flask or Django due to specific performance and integration needs:

- Performance: FastAPI provides exceptional speed, leveraging Python's asynchronous capabilities. This was non-negotiable for meeting the strict latency requirement (RNF-01), ensuring a near-instantaneous response after querying Firebase.
- Automatic Validation: The framework leverages Python and the **Pydantic** library for the automatic validation of input and output data. This functionality guarantees a **rigid data contract** with the *frontend* partner, drastically reducing the risk of *runtime* errors and ensuring data integrity upon request (RF-06).
- Automatic Documentation: FastAPI automatically generates **interactive API documentation (Swagger/OpenAPI)**. This feature was crucial for facilitating rapid integration, allowing the external thesis partner to immediately understand the structure, endpoints, and required JSON schema without manual documentation updates.

Database and Authentication: Firebase

Google Firebase was integrated as the primary data platform, leveraging its dual capabilities for data storage and security management:

- Data Acquisition: **Firebase Firestore** provides a flexible NoSQL structure ideal for storing the frequently updated, yet non-relational, *food* and *sleep logs*. The FirestoreManager class handles the efficient connection and querying required for the analysis time window (RF-02).

- Security: Firebase is central to the security architecture. It manages **user authentication**, allowing the API service to validate the client's **ID Token** using the firebase_admin.auth module. Furthermore, internal **service account keys** are used for secure, server-side data retrieval (RF-01), creating a protected communication tunnel between the API and the data itself.

Knowledge Storage: JSON and Modular Structure

The design of the Knowledge Storage was predicated on its ability to be easily managed and updated.

- Decoupling: The operating rules are encoded entirely in **JSON format**. This decouples the **business logic** (the JSON file) from the **programmatic logic** (the Python engine). The RulesEngine reads and interprets these JSON files dynamically at *runtime*, allowing rules to be updated without restarting or recompiling the core API service.
- JSON is a **human-readable** format, which is a prerequisite for its modification via the high-level GUI.

Management User Interface (GUI): Tkinter

To support the prototyping phase and meet the usability requirements (RNF-05), two dedicated GUIs were developed using **Tkinter**, Python's standard GUI library.

- Analysis GUI: This GUI was vital for **Validation and Debugging** (RF-07). It integrates the matplotlib library to provide graphical summaries (e.g., macronutrient distribution) alongside the textual recommendations. This visual debugging tool was crucial for verifying the logical correctness of the rules before the final API integration.
- Rule Management GUI: This tool ensures **Knowledge Base Usability** (RNF-05). It utilizes Tkinter components to allow direct editing and saving of JSON rule files, integrated with a mandatory **syntax validation check** to prevent the deployment of corrupted rules.

3. Nutritional Recommendation Form (Core of the Project)

3.1 Data Input: The Food Log

The quality and reliability of the recommendations generated by the system are contingent upon the consistency and completeness of the input data provided by the user. This chapter delineates the evolution of the Data Pipeline, a process that has transformed the system from a static file-based analyser to a dynamic service integrated with a cloud database.

3.1.1 Data Extraction and Initial Integration with FatSecret

In the project's preliminary phase, the priority was the validation of the Rules Engine through a set of standardised, precise and detailed data .For the purpose of this study, it was decided that the most suitable method of data collection would be through the utilisation of food logs that had been exported from the FatSecret platform.

This decision was executed through the establishment of an ad-hoc parsing module, which was integrated within the *parse_fatsecret_report* function. The following aspects of the module fall under its remit:

1. **Reading and Interpretation**: The .csv files exported from FatSecret were then opened.
2. **Structural Recognition**: Use regular expressions to recognize and separate entries by date, meal, and food, since FatSecret's CSV format is standardized.
3. **Normalization**: Extract nutritional values (calories, macronutrients, and micronutrients) and normalize them into a consistent data structure ready for analysis.

The preliminary integration with FatSecret, while effective for prototyping and functional validation of the rules engine, exhibited two primary limitations: it was a manual process (requiring file uploads via the GUI) and did not permit integration with a real-time frontend application.

3.1.2 Evolution: Data Acquisition from Firebase

The transition to a service-oriented architecture and integration with the mobile application necessitated the adoption of a cloud database for real-time data management. The decision was taken to designate Firebase as the repository for this information, thus establishing it as the definitive source of truth.

This evolution entailed the reengineering of the acquisition module, with file parsing being replaced by a direct database query mechanism.

A. Data Centralization and Authentication

It has been established that all user data, including that pertaining to nutrition and sleep, is recorded by the front-end application and stored directly in Firebase. The delegation of access management to the FirestoreManager module is pivotal in ensuring the authentication process is conducted via Firebase service credentials.

B. Data Retrieval and Time Window

The analysis modules have been updated to now invoke specific methods that retrieve data from Firebase. The retrieval process is parameterised to operate within a specific time period (seven days by default). This approach ensures that the analysis focuses on weekly patterns, in line with the system's objectives.

C. Architectural Advantages

The adoption of Firebase resolved the issue of manual labour (RF-01) and satisfied the non-functional requirements of scalability and security (RNF-02), thereby enabling the API service (FastAPI) to establish a connection asynchronously, retrieve data, and initiate analysis with minimal latency (RNF-01).

This transition from static files to a cloud database signifies a pivotal step in the project, thereby transforming a prototype into a contemporary, integrable backend service.

3.2 The Knowledge Model: Structure and Taxonomy of Rules

The Recommendation Engine is classified as a **Knowledge-Based System**. The effectiveness of this approach is contingent not on statistical learning, but rather on the validity and accuracy of the rule base that encapsulates nutritional knowledge and lifestyle guidelines.

The modelling of knowledge was achieved through the implementation of a hierarchical dictionary of logical rules, which was maintained externally to the execution code in order to ensure maintainability and transparency.

3.2.1 Rule Types: Defining Simple and Complex Logic

In order to address the different complexities of nutritional and sleep guidelines, the knowledge base has been split into two distinct types of rules, as specified by the "*type*" field within the JSON structure.

A. "Simple" rules

The objective of a simple rule is to evaluate a specific metric in relation to a constant threshold value. These measures are considered optimal for addressing fundamental nutritional standards and safety recommendations.

Logic: The condition of the rule is typically based on a standard comparison operator (<, >, =, <=, >=). Example Usage: It is imperative to ascertain whether the total daily calorie intake or the number of nights with less than 6 hours of sleep exceeds the recommended maximum.

B. "Complex" rules

The system's most sophisticated diagnostic core is constituted by complex rules that permit dynamic calculations and combined analyses of multiple metrics or variables dependent on the user's context.

Logic: The condition is a complex Boolean expression that can combine several aggregate metrics (e.g. saturated fat plus fried foods) and can include dynamic variables based on the user's weight (e.g. protein requirement in grams per kilogram of body weight).

The following example illustrates the application of the rule. This rule functions to ascertain whether the aggregate sum of specific saturated fats and fried foods exceeds a designated weekly threshold. The calculation involves the summation of specific saturated fats (weekly_counts.get('specific_saturated_fats', 0)) and fried foods (weekly_counts.get('fried_foods', 0)). Secondly, This type of rule is indicative of the complexity of a truly holistic nutritional assessment.

3.2.2 JSON Schema and Logical Category Mapping

To ensure that the Knowledge Base was maintainable and that recommendations could be grouped consistently for the user, a rigorous JSON Schema and a taxonomy of logical categories were defined.

Each rule, regardless of its type (simple or complex), adheres to the following atomic structure:

- name: Unique and descriptive identifier.
- group: Logical category to group the tips (e.g. "Macronutrients", "Hydration", "Sleep Hygiene").
- enabled: Flag for quick activation/deactivation of the rule.
- type: Indicates simple or complex.
    - if simple
        - metric: The variable in the context to evaluate
        - operator: The logical operator to apply (<, >)
        - threshold: The value to be considered
    - if complex
        - condition: The logical Python expression that the engine will evaluate, pre (e.g. 2000 <= avg_calories > 2500).
- message: The text of the tip, with dynamic placeholders (e.g. You have consumed {total_sat_fried_items} times).
- message_params: (Optional) Contains Python expressions to calculate values to inject into the message placeholders at run time (e.g. total_sat_fried_items).

The group field is essential for mapping the Knowledge Model. It facilitates the organisation of rules into coherent sets, enabling both the maintenance of these rules via the GUI and the orderly presentation of results to the end user.

This knowledge architecture ensures the extensibility of the system (by means of the addition of new JSON files or rules) and its transparency, as each recommendation can be uniquely traced back to the specific rule violated.

3.3 The Weekly Execution and Analysis Algorithm (Rules Engine)

The intellectual core of the project is constituted by the Rules Engine, a system developed in Python that models the decision-making process of a nutritionist or lifestyle expert. In contradistinction to machine learning-based systems, which necessitate voluminous labelled datasets and proffer minimal explainability, the Rule-Based Engine ensures transparency and clinical accuracy, as it is predicated on recognised nutritional guidelines and health parameters.

3.3.1 Algorithm Phases: Prep-Processing, Run-time and Message Generation

The analysis and recommendation algorithm operates in a cycle of three main phases, which are executed every time the API service receives a request:

Phase 1: Prep-Processing e Data Aggregation

The present phase constitutes a preparatory step for the raw data received from Firebase (or, in the testing phase, from the FatSecret parser) so that it is ready for logical evaluation:

1. **Retrieval and Time Window**: The data is retrieved from the database (or parsed from a CSV file) and filtered to include only the desired analysis time frame (by default, 7 days).
2. **Aggregate Metrics Calculation**: The objective of this process is to calculate the essential metrics that serve as variables for the rule conditions.
    - Nutrition: The programme performs calculations to determine daily macronutrient averages (avg_macronutrients), weekly serving counts for specific food groups (weekly_counts), and the average total calories.
    - Sleep: Calculation of the average sleep hours, average bedtime and wakeup hour, days with late bedtime, days with early wakeup, days with insufficient sleep and days with poor quality sleep.
3. **User Context Definition**: User-specific variables, such as weight in kilograms (user_weight), are loaded, and are essential for complex rules that require parametric calculations (e.g. protein requirement calculated as user_weight multiplied by X).

Phase 2: Rule Engine Runtime

This is the execution phase where the Knowledge Base logic is applied to the aggregated data:

1. **Context Injection**: The Rules Engine creates an execution context that contains all aggregate metrics and user variables.
2. **Iterative Evaluation**: The engine iterates through the entire list of enabled JSON rules (“enabled”: true). For each rule, the expression contained in the “condition” field is subject to dynamic evaluation within the execution context.
3. **Evaluation Mechanism**: The evaluation process is facilitated by a Python string execution mechanism (e.g. the eval() function), which allows for the flexibility required by complex rules (e.g. the condition "weekly_counts.get('specific_saturated_fat', 0) + weekly_counts.get('fried_food', 0)).
4. **Results Collection**: In the event of a condition being evaluated as True, the rule is considered to have been violated and the rule object is queued for message generation.

Phase 3: Message Generation and Finalization

The final step is to transform the violated rule into a custom recommendation and structure the output for the API.

1. **Dynamic Parameter Calculation**: For rules containing the message_params field, the engine performs the calculations specified in this dictionary.
2. **Placeholder Population**: The engine uses the values calculated and context metrics to replace placeholders ({parameter_name}) in the rule's message field. This ensures that the recommendation is numeric and contextualized (e.g., "You consumed food... 3 times this week").
3. **Output Structure**: The final list of all generated messages is structured in a JSON format, ready to be returned to the API layer, including the message and the group it belongs to.

3.3.2 Implementation and Conditional Logic in Python

The Python implementation is the key element that ensures the efficiency and power of the Rules Engine.

Dynamic Condition Execution

The capacity to implement intricate regulations without necessitating code recompilation is facilitated by the eval() function or analogous processes.

The benefit of this system is that it allows administrators to modify logical conditions via the GUI and load them immediately without intervention on the backend code (in accordance with the RNF-04 Maintainability requirement).

Execution Context: The script establishes a secure namespace, which is defined as a dictionary, and contains all the necessary variables (weekly_counts, avg_macronutrients, user_weight). The execution of code is dynamic within this specific scope, thereby mitigating potential security concerns associated with the use of the eval() function.

Modularity of the Analysis

The algorithm is structured into separate modules for nutrition (recommendation_food.py) and sleep (recommendation_sleep.py). This modularity is achieved by reusing the RulesEngine class and differentiating the inputs (the rules' JSON files and the specific aggregate metrics):

- recommendation_food.py: Focuses on processing the food log and using the weekly_counts and avg_macronutrients metrics.
- recommendation_sleep.py: Focuses on processing sleep data and using specific metrics such as regularity and duration.

The system's modular design facilitates future extensions, such as the incorporation of the Physical Activity Recommendation Module, which necessitates only new prep-processing and a revised set of JSON rules.

3.4 Rules Management GUI Development and Functionality

In order to guarantee that the Knowledge Model (Chapter 3.2) can be maintained and updated by non-technical personnel, such as nutritionists or future system administrators, a dedicated graphical user interface (GUI) was developed. This standalone application, implemented in the gui_rules.py file, serves as the single point of access for securely editing the JSON files containing the rules.

3.4.1 High Level Input Interface (Tkinter)

The rules management GUI was built using the Tkinter library, the standard Python toolkit. This choice prioritized the simplicity and independence of the administration tool:

- **Module Navigation**: The GUI enables the user to select the rule set to be edited (e.g. Nutrition, Sleep). This is of crucial importance since the rules are physically separated into different JSON files, thereby supporting the system's modularity.
- **Rule List**: The screen displays a clear list of rule names and their status, allowing for rapid identification and editing access.
- **Integrated Editor**: The interface provides a dedicated text editor that loads the raw JSON content of the selected rule. Although the input does not conform entirely to the "text box-like" paradigm, particularly in more complex fields such as condition (a string of Python code), the widget-based interface provides a controlled environment that mitigates the need for manual intervention in editing JSON files, thereby offering a heightened level of abstraction. The objective is to facilitate the incorporation of predefined logical structures, with the capacity to modify solely the critical parameters.
- **Basic functionality**: The tool incorporates essential features, including the capacity to expeditiously enable or disable rules (by modifying the "enabled" field to true or false) and to introduce new rules by duplicating existing ones.

3.4.2 Atomic Validation and Rescue Mechanisms

The maintainability and reliability requirements (RNF-03 and RNF-04) are satisfied by mechanisms that protect the Knowledge Base from destructive errors. Since a syntax error in a JSON file has the potential to result in the failure of the entire Rules Engine in production. Therefore, the GUI implements a precautionary save logic:

- **JSON Syntax Validation (Pre-Save Check)**: When the user presses the save button, the algorithm attempts to execute the json.loads(json_text) function on the text in the editor. In the event of failure (i.e. if the error code is 'json.JSONDecodeError'), a clear error message is displayed (messagebox.showerror in gui_rules.py) and the save to disk operation is aborted. This is an essential process as it prevents invalid data from being written, which would cause the Rules Engine to crash at runtime.

- **Atomic and Coherent Rescue**: Only after passing JSON validation is the file overwritten to disk (save function in gui_rules.py). This process is made as "atomic" as possible, where the data is written and the Rules Engine on the backend can reload the file with the new ruleset. Using the Tkinter toolkit, separate from the API environment, ensures that any GUI crashes or freezes have no impact on the stability of the active API service.

These syntax error prevention mechanisms make the GUI a fundamental component for the operational management of the Knowledge Model.

4. Extension of the Recommendation System (Lifestyle and Wellbeing Modules)

The initial nutritional recommendation system was strategically designed to be extensible, enabling the integration of **lifestyle analysis and advice**, with an initial focus on **sleep**. This extension maximizes the utilization of the **Rules Engine's modularity** (RNF-04), creating a second, parallel analysis pipeline to the nutritional module.

Integrating sleep is a critical clinical decision because inadequate rest is strongly linked to poorer food choices, metabolic dysregulation, and reduced motivational capacity, meaning that isolated nutritional recommendations are often ineffective.

4.1 Implementation of the Sleep Module

4.1.1 Sleep Data Integration

The successful integration of the Sleep module required extending the algorithm's **prep-processing** stage to define and manage a new distinct data pipeline,validating the flexibility of the core architecture.

A. Unified Data Source (Firebase)

Unlike the initial nutrition phase, which relied on external, static logs (e.g., FatSecret), sleep data was integrated directly into the project's real-time service architecture.

- **Database:** Sleep data (e.g., sleep hours) is recorded by the *frontend* application and saved to the **same Firebase Firestore instance** used for nutritional data. This maintains Firebase as the **single source of truth** for all user lifestyle data, simplifying the architecture and supporting the security model.
- **Acquisition Module:** The file reccomandation_sleep.py implements the core acquisition logic. The function analyze_sleep_from_firebase invokes the FirestoreManager object using internal service credentials (RF-01) to securely retrieve raw data. Access is consistently filtered by user_id and restricted to a rolling **seven-day time window** to effectively analyze weekly circadian patterns.

B. Specific Analysis Metrics

The algorithm executes a tailored **prep-processing** step on the raw sleep data to generate the high-level metrics required for rule evaluation in the execution context. This moves the system beyond simple duration logging to actionable insights:

- **Quantitative Duration:** Calculates the average number of hours of actual sleep per week.
- **Circadian Regularity/Consistency:** Analyzes the **variation** in bedtime and wake-up time, which is a key metric for evaluating the health of the user's **circadian rhythm**.
- **Deficiency Frequency:** Counts the number of days on which sleep was below a critical threshold (e.g., 7 hours).

These processed metrics are then robustly injected into the execution context of the **Rules Engine**, in a similar mechanism used for nutritional variables.

4.1.2 Logic of Additional Rules for Sleep

The cognitive extension of the system is materialized by the new JSON ruleset for sleep, which expands the **Knowledge Model**. This set of rules demonstrates the **applicability of the complex rule type** to non-nutritional data.

A. New Logical Categories

To maintain organization and transparency, two new primary categories (groups) were introduced to the taxonomy:

1. **Sleep Hygiene:** Rules focused on **behavioural and environmental factors** (e.g., timing of meals before sleep, consistency of schedule).
2. **Sleep Duration and Quality:** Rules based on **quantitative metrics** (total hours) and **consistency** over the seven-day period.

B. Examples of "Complex" Sleep Rules

Sleep recommendations often rely on analyzing consistency and frequency over time, necessitating the use of **complex rules** to combine multiple metrics. See Table 4.1.

Category	Example of Conditional Logic (Internal)	Typical Personalized Message Generated
Duration	IF the average sleep duration is < 7 hours **AND** this insufficient duration occurred **MORE THAN** 3 times in the last 7 days.	"Your average sleep duration is only {X} hours. Try to get at least 7-8 hours of rest per night."
Hygiene	IF the average difference between bedtimes is > 90 minutes (high inconsistency).	"Your sleep pattern is highly inconsistent, with an average difference of {Y} hours between nights. Try to maintain a fixed schedule to improve circadian health."

Table 4.1

The execution of the Sleep module is entirely **decoupled at the logic level** (using separate JSON files and analysis code reccomandation_sleep.py) but unified at the service level, with the API exposing a dedicated endpoint (/sleep in api_service.py) that specifically triggers this analysis and returns the structured recommendations.

4.2 Implementation of the Mood Module

The Mood module represents the expansion of the system into the domain of subjective psychological well-being. This allows for the analysis of emotional patterns in relation to objective behaviors (sleep and activity), leveraging the flexibility of the Rules Engine to analyze complex intensity data.

4.2.1 Rationale for Mood Tracking and Data Metrics

- **Rationale**: Tracking mood is essential for holistic counselling, as emotional states directly affect adherence to routines, sleep quality and motivation levels. Mood analysis enables targeted recommendations for psychological stability, a prerequisite for long-term behavioural change.

- **Data Metrics**: The data is recorded by the frontend application and saved on Firebase. The input consists of ten discrete values (on a scale of 1 to 5) that indicate the intensity of specific moods during the reference period (7 days).
- **Derived Metrics (Prep-Processing)**: The prep-processing phase calculates the following aggregate metrics, which are essential for evaluating rules and identifying patterns of stress or vigilance:
    - **Emotional Balance**: avg_positive_mood and avg_negative_mood, calculated as the normalized average of positive and negative metrics.
    - **Alertness and Attention**: avg_attentive (average attention intensity).
    - **Frequency of Critical States**: nervous_high_days, hostile_high_days, and their percentages (hostile_high_percentage, nervous_high_percentage). These metrics quantify the incidence of potentially dysfunctional emotional states during the week of analysis.

4.3 Implementation of the Physical Activity Module

The Physical Activity module completes the holistic approach by providing essential data for optimising consistency, frequency and recovery. All data is acquired in real time via Firebase from the front-end application.

4.3.1 Integration and Metrics from Activity Trackers

- **Data Source:** The raw data is taken from Firebase and includes basic quantitative metrics such as: steps (total steps), calories_burned (calories burned), biking_distance (distance cycled), and walking_distance (distance walked).
- **Derived Metrics (Prep-Processing):** The analysis engine aggregates this data on a weekly basis to calculate the metrics required for the rules:
    - **Total Volume**: Calculation of Total Caloric Expenditure (sum of calories burned), Total Steps, and Total Distance (sum of walking and biking distance).
    - **Consistency and Frequency**: Determination of the number of days of significant activity (e.g., days when steps exceed a threshold) and assessment of weekly consistency.

5. System Integration and API Services

The project's architectural journey involved two main phases: **initial validation** and **service deployment**. The first phase saw the implementation of the **Recommendation GUI** (used for internal testing), which was essential for debugging the core logic. The second phase involved transitioning the validated Rule-Based Engine from a standalone prototype to a robust, real-time backend service via the **RESTful API layer** (used for external deployment). This two-step process ensured that the core intelligence was proven correct before being exposed to the external mobile application.

5.1 The Recommendation GUI (Analysis and Output GUI)

Before developing and deploying the API backend, an initial Python GUI was implemented using the **Tkinter** library and **Matplotlib**. This application was vital, serving as the **internal validation and debugging environment** for the core logic.

- **Technical Validation:** It enabled to test the effectiveness of the Rules Engine locally using test logs (CSV files initially parsed from FatSecret), thereby completely bypassing the Firebase/API integration until the core logic was proven sound.
- **Debugging and Transparency:** It served as the primary **debugging tool**, capable of displaying both the final recommendation and the granular data that generated it, which was essential for verifying the correctness of the JSON rules..

5.1.1 Loading the Log and Viewing the Analysis Results

The GUI was designed to provide a comprehensive, all-in-one analysis environment capable of managing the entire algorithm execution pipeline within a local context

A. Loading the Log (Initial Testing Phase)

The interface includes an **'Upload File'** function that invokes the dedicated parsing module, allowing developers to select local CSV files and start the analysis immediately. Crucially, the interface requires the user to specify **context variables**, such as body weight, which are essential parameters for correctly executing the **complex rules** that calculate personalized protein requirements.

B. Data Visualization and Aggregations

Once the data has been uploaded and preprocessing is complete, the GUI visualizes the aggregated metrics before the rules are executed:

- **Numeric Summary:** A dedicated section displays key aggregated results of the analysis (e.g., Average Daily Calories, Average Grams of Protein).
- **Graphical Representation (Matplotlib):** The GUI uses the **Matplotlib** library to generate and display dynamic charts. Typical charts include **Weekly Macronutrient Balance** (pie/bar charts) and **Weekly Distribution** (line charts) showing the trend of calories or specific micronutrients day by day. This visual output was essential for **external validation** and for identifying anomalies in the input data before the Rules Engine was executed.

5.1.2 Presentation of Nutritional Advice

The final result of the Rules Engine execution is presented in a dedicated section, emphasizing structure and transparency.

- **Categorical Structure of Output:** Recommendations are logically organized using the **categories** (group) defined in the JSON files. For instance, sodium recommendations are categorized under **"Micronutrients,"** separated from protein balance recommendations **("Macronutrients and Balance")**. This structure prepares the data for an organized display in the application.
- **Engine Transparency and Debugging:** This GUI satisfies the **Explainability (XAI) requirement** through a built-in debugging mechanism not present in the final mobile application:

5.2 API Development for External Service

The final phase involved converting the **Rules Engine** into a network-accessible service. This was achieved by developing a **RESTful API** interface using the FastAPI microframework, which ensured high performance and automatic documentation.

These APIs act as the essential authenticated bridge between the front-end mobile application and the back-end analytics logic.

The decision to expose the analytic engine as a RESTful microservice was driven by three key considerations: interoperability, operational simplicity, and the ability to evolve independently.

REST over HTTPS is widely supported by mobile clients and serverless platforms, facilitating rapid integration with partner mobile applications and third-party services without the need for client-side libraries. Microservice boundaries enable the analytics engine to scale independently and be deployed using container orchestration tools or serverless functions to match production load profiles.

5.2.1 Designing API Endpoints

The design of the API endpoints adheres to the principle of **separation of responsibilities** (for the nutrition and sleep modules) and facilitates efficient, secure communication. See Table 5.2.

Endpoint	**Method**	**Description and Features**
/nutrition	**POST**	The primary endpoint for nutritional analysis. It requires the authenticated user ID and starts the entire pipeline: data fetch from Firebase, running reccomendation_food.py (Rules Engine), and returning recommendations.
/sleep	**POST**	Endpoint dedicated to analyzing sleep data. It invokes reccomandation_sleep.py and returns lifestyle recommendations.
/health	**GET**	*Health Check*. Used for deployment and monitoring, verifying service health and real-time connectivity to the Firebase database (RNF-03).

/mood	**POST**	Endpoint dedicated to analyzing mood data. It invokes reccomandation_mood.py and returns lifestyle recommendations
/activity	**POST**	Endpoint dedicated to analyzing activity data. It invokes reccomandation_sport.py and returns lifestyle recommendations
/categories	**GET**	Provides a list of available logical categories (group). This allows the *frontend* to understand the *backend*'s taxonomy for proper display and filtering.

Table 5.2

The choice of the **POST** method for the analysis endpoints is used to include user metadata and authentication tokens securely within the request body, even when data is retrieved from the database.

5.2.2 Communication and Data Exchange Protocols

Communication between the *frontend* (mobile application) and the *backend* API is standardized to ensure robustness and security.

- **Transport Protocol and Architecture:** The system uses the standard **HTTP/S protocol** with a **RESTful architecture**. The use of **FastAPI** ensures correct handling of HTTP status codes (e.g., 200 OK, 400 Bad Request), which is critical for the *frontend*'s error handling mechanisms.
- **Data Exchange Format: JSON: JSON** is the sole format used for data exchange. The **Input** (Request Body) includes a JSON object with the user ID and authentication token. The **Output** (Response Body) returns a JSON object containing the list of generated recommendations, where each item is a structured object including the message text and its associated category (group).
- **Authentication and Security (RNF-02):** Every API request requires authentication. The API verifies the client by validating the **Firebase ID Token** sent in the request header using the Firebase SDK. Furthermore, the implementation includes a dedicated

system of **customized exceptions** that standardizes error responses. In the event of a service failure (e.g., **Firebase** connection failure), the API returns a structured JSON error response, including a shared error code (e.g., ERR_DB_001), enabling the *frontend* to display specific and actionable feedback rather than a generic service crash.

5.2.3 Integration with the Other Thesis Student's Mobile Application

Integration with the mobile application (developed in parallel by another candidate) was the ultimate validation of the system's design. The decoupled **API approach** allowed the two development streams to work independently, converging efficiently through key integration points:

1. **Defined API Contract:** The **API documentation** served as the formal contract between the *frontend* and *backend*. Both parties agreed precisely on the endpoint names, the required JSON schema for requests and responses, and the complete list of structured error codes.
2. **Closed-Loop Data Flow:** The integration established a clear, closed-loop workflow: The mobile app writes data -> The mobile app authenticates and calls the API -> The API reads data from Firebase -> The Rules Engine executes -> The API returns the recommendations.

This integration demonstrated the viability of the microservices design, successfully positioning the **Rules Engine** as a scalable, secure, central service ready for use in a production environment.

6. Results, Evaluation and Discussion

This chapter presents the system's operational results, evaluates the effectiveness of the implemented solution and considers how well the functional (RF) and non-functional (RNF) requirements have been met.

6.1 System Demonstration and Operational Validation

The backend system that was implemented was tested in an end-to-end configuration to verify its ability to execute the full analytical pipeline, from retrieving data through the API to generating personalized recommendations.

This process confirmed the consistency and interoperability of all modules, ensuring that the engine fulfils all functional and integration requirements.

6.1.1 Use Case 1 – Nutritional Analysis and Complex Rules

This first scenario validates the execution of the core analytical logic (RF-03) and the dynamic generation of recommendations (RF-05).

Scenario:
A user weighing 70 kg records a weekly average protein intake of 65 g and consumes foods rich in saturated fats (e.g., fried foods) four times in seven days.

Process and Output:

- The system evaluates the protein-to-body-weight ratio, triggering a rule when intake falls below 1 g/kg.
- It simultaneously activates a complex rule combining two weekly counters (saturated fats and fried foods) to assess excessive frequency.

Generated Recommendations:

- Macronutrients and Balance:
  "Your average daily protein intake of 65 g is below the optimal target of 70–84 g. Increase your intake of lean sources."
- Foods to Limit:
  "You consumed foods high in saturated fat or fried foods four times this week. Limit to a maximum of twice to improve cardiovascular health."

Validation:
These results confirm that:

- The engine correctly evaluates both simple (threshold-based) and complex (aggregated) conditions.
- The recommendation text dynamically integrates user-specific variables (e.g., weight and weekly counts), achieving true personalization.

6.2 System Evaluation

The system was evaluated through quantitative and qualitative analyses to verify compliance with the non-functional requirements (RNF).

6.2.1 Validation of the Knowledge Base

The transparency and reliability of the Rule-Based Engine (RNF-03) were ensured through a two-step validation process supported by the dedicated GUIs.

1. **Syntactic Validation (RF-04):** The *Rule Management GUI* automatically checks JSON syntax before saving, preventing malformed rules (e.g., missing commas or brackets) from being stored. This guarantees that the engine always operates with a valid, readable Knowledge Base.
2. **Logical and Clinical Verification (RF-07):** Using the *Analysis GUI*, test logs were intentionally crafted to trigger specific rules. The threshold to verify the accuracy of the rules is specified in the rule output message. This immediate feedback confirmed that all *simple* and *complex* rules behaved consistently with their clinical intent and logical design.

Requirement	Description	Verification evidence
RF-01	Secure data acquisition from Firebase	Token validation tested; FirestoreManager logs and authenticated calls
RF-02	Weekly aggregation and time-window analysis	Preprocessing unit tests; example aggregated outputs in GUI
RF-03	Dynamic Rule Engine execution	Triggered rules in Use Case 1; debug output with violated condition
RF-04	Knowledge base management via GUI	GUI save validation and atomic update mechanism implemented
RF-05	Recommendation generation with dynamic placeholders	Message_params evaluated and populated in sample recommendations
RF-06	Exposed REST API endpoints (/nutrition, /sleep, /categories)	FastAPI auto-docs and integration with mobile client
RNF-01	Target latency < 3 s	Measured average ≈ 2 s for /nutrition endpoint
RNF-02	Data security & HTTPS	HTTPS enforced; Firebase authentication validated

RNF-03	Robustness and error handling	Centralized exception handler and structured error codes (ERR_...)
RNF-04	Maintainability (decoupling code and knowledge)	JSON rules loaded dynamically; modular rule files per domain
RNF-05	Knowledge-base usability	GUI syntax checks and rescue mechanisms implemented

Table 6.2.2

6.3 Discussion and Critical Analysis

The implemented solution successfully achieved its design goals, proving that a **Rule-Based and modular architecture** can deliver personalized and explainable lifestyle recommendations with low computational cost.

6.3.1 Strengths and Achievements

- **Explainability (XAI):** The rule-based approach inherently guarantees transparency: each recommendation can be traced back to the exact logical condition that produced it. This satisfies both clinical accountability and user trust requirements.
- **Maintainability and Modularity:** The separation between the Python engine and the JSON Knowledge Base, managed through the GUI, enables rapid updates without code changes. The addition of the Sleep module validated the system's extensibility and confirmed the robustness of its modular design (RNF-04).
- **Effective Hybridization:** While primarily rule-based, the system already incorporates a hybrid logic by combining nutritional and sleep data, forming the foundation for a comprehensive *Lifestyle Recommendation System* rather than a simple Nutritional Recommender.

6.4 Summary

In conclusion, the project delivers a **fully functional, secure, and transparent backend service** for personalized wellness recommendations.

It demonstrates that **Knowledge-Based Systems**, when properly modularized and coupled with cloud integration, remain a powerful paradigm for clinical-grade personalization, offering reliability, explainability, and low latency that purely data-driven systems often lack.

The work establishes a robust foundation for future hybrid systems, where **expert-defined rules and adaptive learning** can coexist to achieve both **trustworthiness and continuous personalization**.

7. Conclusions and Future Developments

7.1 General Conclusion

This thesis presents the design and implementation of a Knowledge-Based engine for generating personalised nutritional and lifestyle recommendations. This represents an evolution from traditional data-tracking applications towards intelligent, explainable wellness systems.

The system developed in this study shows that knowledge-driven approaches combined with modular software architecture and cloud integration can effectively convert raw behavioural data into actionable, clinically meaningful insights.

By formalising expert knowledge into a transparent and maintainable rule structure, the project was able to achieve a high level of personalisation without sacrificing interpretability, which is a common limitation of purely data-driven models.

From a methodological standpoint, the project's success lies in the **decoupling of logic and knowledge**:

- The **Python-based Rules Engine** ensures efficient execution and computational transparency.
- The **JSON Knowledge Base**, editable through an intuitive GUI, allows domain experts to modify rules dynamically, bridging the gap between technical implementation and clinical expertise.
- The **FastAPI backend** and **Firebase integration** enable real-time interaction with external applications, transforming the prototype into a production-ready, service-oriented architecture.

The system's **modular extension to lifestyle data**, particularly through the integration of sleep analysis, demonstrates the scalability and generality of the proposed framework. This confirms that the same logic can be applied to additional domains, positioning the system as a foundation for broader lifestyle intelligence applications.

7.2 Limitations

Despite its achievements, several limitations remain, offering opportunities for further refinement:

- **Manual Knowledge Expansion**: The rule base currently relies on expert-defined logic, which can be labor-intensive to scale. Automating knowledge extraction or supporting semi-automatic rule suggestions could enhance long-term sustainability.
- **Absence of Adaptive Feedback Mechanisms**: The system currently operates in a *prescriptive* mode, it issues recommendations but does not adapt based on user compliance or outcomes. Future iterations could implement learning mechanisms that adjust rule weights or thresholds according to behavioral trends.

7.3 Future Developments

Building upon the current architecture, several promising directions can be pursued:

1. **Hybridization with Machine Learning Models:** Introducing a **hybrid AI layer** on top of the Rule Engine (for example, a recommendation ranking model or a reinforcement learning agent) could allow the system to adapt dynamically while keeping the rule-based layer as a clinical safeguard.
2. **User Feedback Loop and Behavioral Adaptation:** Incorporating a mechanism to collect feedback from users on the usefulness and adoption of recommendations could enable an adaptive cycle of personalization, improving engagement and accuracy over time.
3. **Ontology-Based Knowledge Representation:** Migrating the rule base to an ontology-driven model would enhance interoperability, semantic reasoning, and automated rule generation, facilitating scalability to multiple health domains.
4. **Deployment and Evaluation in Real User Studies:** The next phase should involve deploying the system to real users within a controlled study, assessing not only technical performance but also **behavioral impact**, user adherence, and satisfaction.

7.4 Final Remarks

In conclusion, this thesis demonstrates that **Rule-Based Systems remain a cornerstone of explainable AI in the wellness domain**, particularly when safety, interpretability, and personalization are non-negotiable requirements.

The proposed architecture bridges the gap between **expert-driven reasoning** and **cloud-scale implementation**, offering a solid foundation for next-generation wellness platforms that combine scientific rigor with real-world usability.

Ultimately, the project contributes to a paradigm shift, from static tracking applications toward **proactive, personalized, and interpretable digital health companions**, capable of evolving alongside their users and learning from their daily habits.

8. Bibliography

[1] Carreiro S, Newcomb M, Leach R, Ostrowski S, Boudreaux ED, Amante D. Current reporting of usability and impact of mHealth interventions for substance use disorder: A systematic review. Drug Alcohol Depend. 2020 Oct 1;215:108201. doi: 10.1016/j.drugalcdep.2020.108201. Epub 2020 Aug 2. PMID: 32777691; PMCID: PMC7502517.

[2] Michie, Susan & van Stralen, Maartje & West, Robert. (2011). The Behaviour Change Wheel: a new method for characterising and designing behaviour change interventions. Implementation science : IS. 6. 42. 10.1186/1748-5908-6-42.

[3] Hulsen, T. (2023). Explainable Artificial Intelligence (XAI): Concepts and Challenges in Healthcare. AI, 4(3), 652-666. https://doi.org/10.3390/ai4030034

[4] Sittig DF, Singh H. A new sociotechnical model for studying health information technology in complex adaptive healthcare systems. Qual Saf Health Care. 2010 Oct;19 Suppl 3(Suppl 3):i68-74. doi: 10.1136/qshc.2010.042085. PMID: 20959322; PMCID: PMC3120130.

[5] Burke, R. (2002). Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction, 12(4), 331-370.Burke, Robin. (2002). Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-Adapted Interaction. 12. 10.1023/A:1021240730564.

[6] Ricci, Francesco & Rokach, Lior & Shapira, Bracha. (2010). Recommender Systems Handbook. 10.1007/978-0-387-85820-3_1.

[7] Tintarev, Nava & Masthoff, Judith. (2015). Explaining Recommendations: Design and Evaluation. 10.1007/978-1-4899-7637-6_10.

[8] Felfernig, Alexander & Burke, Robin. (2008). Constraint-based recommender systems: Technologies and research issues. ACM International Conference Proceeding Series. 3. 10.1145/1409540.1409544.

[9] Ordovas JM, Ferguson LR, Tai ES, Mathers JC. Personalised nutrition and health. BMJ. 2018 Jun 13;361:bmj.k2173. doi: 10.1136/bmj.k2173. PMID: 29898881; PMCID: PMC6081996.

[10] Tkalčič, Marko & Burnik, Urban & Odic, Ante & Kosir, Andrej & Tasic, Jurij. (2013). Emotion-Aware Recommender Systems – A Framework and a Case Study. 10.1007/978-3-642-37169-1_14.

[11] Abhari S, Safdari R, Azadbakht L, Lankarani KB, Niakan Kalhori SR, Honarvar B, Abhari K, Ayyoubzadeh SM, Karbasi Z, Zakerabasali S, Jalilpiran Y. A Systematic Review of Nutrition Recommendation Systems: With Focus on Technical Aspects. J Biomed Phys Eng. 2019 Dec 1;9(6):591-602. doi: 10.31661/jbpe.v0i0.1248. PMID: 32039089; PMCID: PMC6943843.

9. Appendix

9.1 nutrient_rules.json

```
[

  {

    "name": "Grassi Saturi e Alimenti Fritti",

    "group": {

      "it": "Alimenti da Limitare",

      "en": "Foods to Limit"

    },

    "enabled": true,

    "type": "complex",

    "condition": "(weekly_counts.get('grassi_saturi_specifici', 0) +
weekly_counts.get('alimenti_fritti', 0)) > 2",

    "message": {

      "it": "Hai consumato alimenti ricchi di grassi saturi o fritti
{total_sat_fried_items} volta/e questa settimana. L'assunzione dovrebbe
essere limitata a massimo 2 volte a settimana.",

      "en": "You have consumed foods high in saturated fats or fried foods
{total_sat_fried_items} time(s) this week. Intake should be limited to a
maximum of 2 times per week."

    },

    "message_params": {

      "total_sat_fried_items": "lambda weekly_counts:
weekly_counts.get('grassi_saturi_specifici', 0) +
weekly_counts.get('alimenti_fritti', 0)"

    },

    "reference": ""

  },

  {

    "name": "Apporto Proteico Basso",

    "group": {

      "it": "Macronutrienti e Bilancio",
      "en": "Macronutrients and Balance"

    },

    "enabled": true,

    "type": "complex",

    "condition": "avg_macronutrients.get('protein', 0) < user_weight * 0.8",

    "message": {

      "it": "Il tuo apporto proteico medio giornaliero è di
{avg_daily_protein}g, inferiore all'obiettivo di
{min_protein_target}-{max_protein_target}g (basato su un peso di
{user_weight}kg). Considera di aumentare le fonti proteiche, privilegiando
quelle vegetali (legumi, tofu, tempeh) e animali magre (pesce, pollame, uova,
latticini magri).",

      "en": "Your average daily protein intake is {avg_daily_protein}g, below
the target of {min_protein_target}-{max_protein_target}g (based on a weight
of {user_weight}kg). Consider increasing protein sources, prioritizing
plant-based (legumes, tofu, tempeh) and lean animal sources (fish, poultry,
eggs, low-fat dairy)."

    },

    "message_params": {

      "avg_daily_protein": "lambda avg_macronutrients:
avg_macronutrients.get('protein', 0)",

      "min_protein_target": "lambda user_weight: user_weight * 0.8",

      "max_protein_target": "lambda user_weight: user_weight * 1.2",

      "user_weight": "lambda user_weight: user_weight"

    },

    "reference": ""

  },

  {

    "name": "Apporto Proteico Alto",

    "group": {

      "it": "Macronutrienti e Bilancio",

      "en": "Macronutrients and Balance"

    },

    "enabled": true,

    "type": "complex",
    "condition": "avg_macronutrients.get('protein', 0) > user_weight * 1.2",

    "message": {

      "it": "Il tuo apporto proteico medio giornaliero è di
{avg_daily_protein}g, superiore all'obiettivo di
{min_protein_target}-{max_protein_target}g (basato su un peso di
{user_weight}kg). Valuta di bilanciare le fonti proteiche e di consultare un
professionista se hai dubbi.",

      "en": "Your average daily protein intake is {avg_daily_protein}g, above
the target of {min_protein_target}-{max_protein_target}g (based on a weight
of {user_weight}kg). Consider balancing protein sources and consulting a
professional if you have concerns."

    },

    "message_params": {

      "avg_daily_protein": "lambda avg_macronutrients:
avg_macronutrients.get('protein', 0)",

      "min_protein_target": "lambda user_weight: user_weight * 0.8",

      "max_protein_target": "lambda user_weight: user_weight * 1.2",

      "user_weight": "lambda user_weight: user_weight"

    },

    "reference": ""

  },

  {

    "name": "Percentuale Carboidrati Bassa",

    "group": {

      "it": "Macronutrienti e Bilancio",

      "en": "Macronutrients and Balance"

    },

    "enabled": true,

    "type": "complex",

    "condition": "(avg_macronutrients.get('carbs', 0) * 4 /
(avg_macronutrients.get('kcal') or 1)) * 100 < 45",

    "message": {

      "it": "L'apporto calorico dai carboidrati è del {perc_carbs}%. È
inferiore al range consigliato del 45-60%. Aumenta il consumo di carboidrati
complessi come cereali integrali, frutta e verdura.",
      "en": "The portion of calories from carbohydrates is {perc_carbs}%. It
is below the recommended range of 45-60%. Increase the intake of complex
carbohydrates such as whole grains, fruits, and vegetables."

    },

    "message_params": {

      "perc_carbs": "lambda avg_macronutrients:
(avg_macronutrients.get('carbs', 0) * 4 / (avg_macronutrients.get('kcal') or
1)) * 100"

    },

    "reference": ""

  },

  {

    "name": "Percentuale Carboidrati Alta",

    "group": {

      "it": "Macronutrienti e Bilancio",

      "en": "Macronutrients and Balance"

    },

    "enabled": true,

    "type": "complex",

    "condition": "(avg_macronutrients.get('carbs', 0) * 4 /
(avg_macronutrients.get('kcal') or 1)) * 100 > 60",

    "message": {

      "it": "L'apporto calorico dai carboidrati è del {perc_carbs}%. È
superiore al range consigliato del 45-60%. Cerca di bilanciare, privilegiando
le fonti integrali e limitando gli zuccheri semplici.",

      "en": "The portion of calories from carbohydrates is {perc_carbs}%. It
is above the recommended range of 45-60%. Try to balance, prioritizing whole
sources and limiting simple sugars."

    },

    "message_params": {

      "perc_carbs": "lambda avg_macronutrients:
(avg_macronutrients.get('carbs', 0) * 4 / (avg_macronutrients.get('kcal') or
1)) * 100"

    },

    "reference": ""
  },

  {

    "name": "Percentuale Grassi Bassa",

    "group": {

      "it": "Macronutrienti e Bilancio",

      "en": "Macronutrients and Balance"

    },

    "enabled": true,

    "type": "complex",

    "condition": "(avg_macronutrients.get('fat', 0) * 9 /
(avg_macronutrients.get('kcal') or 1)) * 100 < 20",

    "message": {

      "it": "L'apporto calorico dai grassi è del {perc_fat}%. È inferiore al
range consigliato del 20-35%. Assicurati di includere fonti di grassi sani
come olio EVO, frutta secca, avocado.",

      "en": "The portion of calories from fats is {perc_fat}%. It is below
the recommended range of 20-35%. Make sure to include healthy fats such as
EVO oil, nuts and seeds, and avocado."

    },

    "message_params": {

      "perc_fat": "lambda avg_macronutrients: (avg_macronutrients.get('fat',
0) * 9 / (avg_macronutrients.get('kcal') or 1)) * 100"

    },

    "reference": ""

  },

  {

    "name": "Percentuale Grassi Alta",

    "group": {

      "it": "Macronutrienti e Bilancio",

      "en": "Macronutrients and Balance"

    },

    "enabled": true,

    "type": "complex",
    "condition": "(avg_macronutrients.get('fat', 0) * 9 /
(avg_macronutrients.get('kcal') or 1)) * 100 > 35",

    "message": {

      "it": "L'apporto calorico dai grassi è del {perc_fat}%. È superiore al
range consigliato del 20-35%. Cerca di bilanciare l'assunzione di grassi,
privilegiando quelli insaturi.",

      "en": "The portion of calories from fats is {perc_fat}%. It is above
the recommended range of 20-35%. Try to balance the consumption of fats,
prioritizing unsaturated fats."

    },

    "message_params": {

      "perc_fat": "lambda avg_macronutrients: (avg_macronutrients.get('fat',
0) * 9 / (avg_macronutrients.get('kcal') or 1)) * 100"

    },

    "reference": ""

  },

  {

    "name": "Grassi Saturi Elevati",

    "group": {

      "it": "Macronutrienti e Bilancio",

      "en": "Macronutrients and Balance"

    },

    "enabled": true,

    "type": "complex",

    "condition": "(avg_macronutrients.get('sat_fat', 0) * 9 /
(avg_macronutrients.get('kcal') or 1)) * 100 > 10",

    "message": {

      "it": "L'apporto calorico dai grassi saturi è del {perc_sat_fat}%.
L'obiettivo è meno del 10%. Riduci il consumo di alimenti ricchi di grassi
saturi e aumenta i grassi insaturi (presenti in olio EVO, frutta secca,
avocado, pesce azzurro).",

      "en": "The portion of calories from saturated fats is {perc_sat_fat}%.
The goal is below 10%. Reduce the consumption of saturated fats and increase
unsaturated fats (present in EVO oil, nuts and seeds, and oily fish)."

    },

    "message_params": {
      "perc_sat_fat": "lambda avg_macronutrients:
(avg_macronutrients.get('sat_fat', 0) * 9 / (avg_macronutrients.get('kcal')
or 1)) * 100"

    },

    "reference": ""

  },

  {

    "name": "Grassi Insaturi Elevati",

    "group": {

      "it": "Macronutrienti e Bilancio",

      "en": "Macronutrients and Balance"

    },

    "enabled": true,

    "type": "complex",

    "condition": "(avg_macronutrients.get('fat', 0) * 9 /
(avg_macronutrients.get('kcal') or 1)) * 100 > 10",

    "message": {

      "it": "L'apporto calorico dai grassi insaturi è del {perc_fat}%.
L'obiettivo è meno del 10%.",

      "en": "The portion of calories from unsaturated fats is {perc_fat}%.
The goal is below 10%."

    },

    "message_params": {

      "perc_fat": "lambda avg_macronutrients: (avg_macronutrients.get('fat',
0) * 9 / (avg_macronutrients.get('kcal') or 1)) * 100"

    },

    "reference": ""

  }

]
```

9.2 general_rules.json

```
[

  {
    "name": "Consumo di Frutta e Verdura",

    "group": {

      "it": "Consumo di Frutta e Verdura",

      "en": "Fruit and Vegetable Intake"

    },

    "type": "simple",

    "enabled": true,

    "metric": "avg_fruit_veg_portions",

    "operator": "<",

    "threshold": 5,

    "message": {

      "it": "In media hai consumato solo {metric_value} porzioni di frutta e
verdura al giorno. L'obiettivo è almeno 5 porzioni. Aumenta la varietà e la
quantità per un maggior apporto di vitamine e fibre.",

      "en": "On average, you consumed only {metric_value} portions of fruits
and vegetables per day. The goal is at least 5 portions. Increase variety and
quantity for better vitamin and fiber intake."

    },

    "reference": ""

  },

  {

    "name": "Frequenza Legumi",

    "group": {

      "it": "Consumo di Legumi",

      "en": "Legumes Intake"

    },

    "type": "simple",

    "enabled": true,

    "metric": "legumi",

    "operator": "<",

    "threshold": 2,

      "message": {
```

```json
      "it": "Hai consumato legumi solo {metric_value} volta/e questa settimana. L'obiettivo è 2-3 volte a settimana per un buon apporto di proteine vegetali e fibre.",
      "en": "You have consumed legumes only {metric_value} time(s) this week. The goal is 2-3 times a week for a good intake of plant-based proteins and fiber."
    }
  },
  {
    "name": "Frequenza Pesce",
    "group": {
      "it": "Consumo di Pesce",
      "en": "Fish Intake"
    },
    "type": "simple",
    "enabled": true,
    "metric": "pesce",
    "operator": "<",
    "threshold": 2,
    "message": {
      "it": "Hai mangiato pesce solo {metric_value} volta/e questa settimana. Si consiglia almeno 2 volte a settimana, includendo pesce azzurro ricco di omega-3.",
      "en": "You have eaten fish only {metric_value} time(s) this week. It is recommended at least 2 times a week, including oily fish rich in omega-3."
    },
    "reference": ""
  },
  {
    "name": "Frequenza Carne Rossa Elevata",
    "group": {
      "it": "Consumo di Carne Rossa",
      "en": "Red Meat Intake"
    },

    "type": "simple",

    "enabled": true,

    "metric": "carne_rossa",

    "operator": ">",

    "threshold": 2,

    "message": {

      "it": "Hai consumato carne rossa {metric_value} volta/e questa settimana. È consigliabile non superare 1-2 volte a settimana per ridurre i rischi per la salute.",

      "en": "You have consumed red meat {metric_value} time(s) this week. It is recommended not to exceed 1-2 times a week to reduce the risk of health."

    },

    "reference": ""

  },

  {

    "name": "Rapporto Carne Bianca vs Rossa",

    "group": {

      "it": "Rapporto Carne Bianca vs Rossa",

      "en": "White vs Red Meat Ratio"

    },

    "type": "complex",

    "enabled": true,

    "condition": "weekly_counts.get('carne_bianca', 0) < weekly_counts.get('carne_rossa', 0) and (weekly_counts.get('carne_rossa', 0) + weekly_counts.get('carne_bianca', 0)) > 0",

    "message": {

      "it": "Stai mangiando più carne rossa ({carne_rossa}) che carne bianca ({carne_bianca}). Cerca di favorire il consumo di carne bianca come pollo e tacchino.",

      "en": "You are eating more red meat ({carne_rossa}) than white meat ({carne_bianca}). Try to favor the consumption of white meat such as chicken and turkey."

    },
    "reference": ""

  },

  {

    "name": "Consumo Uova Basso",

    "group": {

      "it": "Consumo di Uova",

      "en": "Egg Intake"

    },

    "type": "simple",

    "enabled": true,

    "metric": "uova",

    "operator": "<",

    "threshold": 2,

    "message": {

      "it": "Hai consumato uova solo {metric_value} volta/e. L'obiettivo è
2-4 volte a settimana per un buon apporto di proteine complete.",

      "en": "You have consumed eggs only {metric_value} time(s). The goal is
2-4 times a week for a good intake of complete proteins."

    },

    "reference": ""

  },

  {

    "name": "Consumo Uova Alto",

    "group": {

      "it": "Consumo di Uova",

      "en": "Egg Intake"

    },

    "type": "simple",

    "enabled": true,

    "metric": "uova",

    "operator": ">",
    "threshold": 4,

    "message": {

      "it": "Hai consumato uova {metric_value} volta/e. Si consiglia di non
superare 4 volte a settimana per mantenere un equilibrio alimentare.",

      "en": "You have consumed eggs {metric_value} time(s). It is recommended
not to exceed 4 times a week to maintain a balanced diet."

    },

    "reference": ""

  },

  {

    "name": "Consumo Latticini Insufficiente",

    "group": {

      "it": "Consumo di Latticini",

      "en": "Dairy Intake"

    },

    "type": "simple",

    "enabled": true,

    "metric": "avg_latticini_per_day",

    "operator": "<",

    "threshold": 1,

    "message": {

      "it": "In media hai consumato solo {metric_value} porzione/i di
latticini al giorno. L'obiettivo è 1-2 porzioni al giorno per l'apporto di
calcio.",

      "en": "On average, you consumed only {metric_value} portion(s) of dairy
products per day. The goal is 1-2 portions per day for good calcium intake."

    },

    "reference": ""

  },

  {

    "name": "Olio Extravergine d'Oliva Quotidiano",

    "group": {
      "it": "Olio Extravergine di Oliva",
      "en": "Extra Virgin Olive Oil"
    },
    "type": "simple",
    "enabled": true,
    "metric": "olio_evo",
    "operator": "<",
    "threshold_dynamic": "num_days",
    "message": {
      "it": "Hai consumato olio EVO solo {metric_value} volta/e su {num_days}
giorni. Assicurati di utilizzarlo quotidianamente come principale fonte di
grassi, preferibilmente a crudo.",
      "en": "You have used EVO oil only {metric_value} time(s) over
{num_days} days. Make sure to use it daily as the main source of fats,
preferably raw."
    },
    "reference": ""
  },
  {
    "name": "Frutta Secca e Semi",
    "group": {
      "it": "Frutta Secca e Semi",
      "en": "Nuts and Seeds"
    },
    "type": "simple",
    "enabled": true,
    "metric": "frutta_secca",
    "operator": "<",
    "threshold_dynamic": "num_days * 0.5",
    "message": {
      "it": "Hai consumato frutta secca e semi {metric_value} volta/e. Cerca
di includerli almeno a giorni alterni per i grassi sani e le fibre.",
      "en": "You have consumed nuts and seeds {metric_value} time(s). Try to
include them at least every other day for healthy fats and fiber."

    },

    "reference": ""

  },

  {

    "name": "Cereali Mancanti",

    "group": {

      "it": "Consumo di Cereali",

      "en": "Cereal Intake"

    },

    "type": "complex",

    "enabled": true,

    "condition": "(weekly_counts.get('num_days', 0) -
weekly_counts.get('days_with_cereals_present', 0)) > 2",

    "message": {

      "it": "I cereali sono mancati in {missing_cereal_days} giorni questa
settimana. I cereali dovrebbero essere presenti quotidianamente nei pasti
principali.",

      "en": "Cereals are missing in {missing_cereal_days} days this week.
Cereals should be present daily in the main dishes."

    },

    "reference": ""

  },

  {

    "name": "Cereali Raffinati Prevalenti",

    "group": {

      "it": "Consumo di Cereali",

      "en": "Cereal Intake"

    },

    "type": "complex",

    "enabled": true,
    "condition": "weekly_counts.get('days_with_refined_cereals_prevalent', 0)
> (weekly_counts.get('num_days', 0) / 2)",

    "message": {

      "it": "Hai consumato cereali raffinati prevalentemente in
{days_with_refined_cereals_prevalent} giorni su {num_days}. Cerca di
preferire i cereali integrali per un maggiore apporto di fibre.",

      "en": "You have consumed refined cereals prevalently in
{days_with_refined_cereals_prevalent} days over {num_days}. Try to prefer
whole grains for a better intake of fiber."

    },

    "reference": ""

  },

  {

    "name": "Consumo di Bevande Zuccherate",

    "group": {

      "it": "Alimenti da Limitare",

      "en": "Foods to Limit"

    },

    "enabled": true,

    "metric": "bevande_zuccherate",

    "source": "weekly_counts",

    "operator": ">",

    "threshold": 1,

    "message":{

      "it": "Hai consumato bevande zuccherate {metric_value} volta/e questa
settimana. È consigliato limitare il consumo a massimo {threshold} volta a
settimana.",

      "en": "You have consumed sugary drinks {metric_value} time(s) this
week. It is recommended to limit consumption to a maximum of {threshold}
times a week."

    },

    "reference": ""

  },

  {
    "name": "Consumo di Salumi",

    "group": {

      "it": "Alimenti da Limitare",

      "en": "Foods to Limit"

    },

    "enabled": true,

    "metric": "salumi",

    "source": "weekly_counts",

    "operator": ">",

    "threshold": 2,

    "message": {

      "it": "Hai consumato salumi {metric_value} volta/e questa settimana. È
consigliato limitare il consumo al massimo {threshold} volte a settimana.",

      "en": "You have consumed cold cuts {metric_value} time(s) this week. It
is recommended to limit consumption to a maximum of {threshold} times a
week."

    },

    "reference": ""

  },

  {

    "name": "Consumo di Dolci e Snack Dolci",

    "group": {

      "it": "Alimenti da Limitare",

      "en": "Foods to Limit"

    },

    "enabled": true,

    "type": "complex",

    "condition": "lambda weekly_counts: (weekly_counts.get('dolci', 0) +
weekly_counts.get('snack_dolci', 0)) > 3",

    "message": {

      "it": "Hai consumato dolci e snack dolci {total_sweet_snacks} volta/e
questa settimana. Si consiglia di moderare il consumo se superiore a
{threshold} volte a settimana.",
      "en": "You have consumed sweets and sweet snacks {total_sweet_snacks}
time(s) this week. It is recommended to moderate consumption if more than
{threshold} times a week."

    },

    "message_params": {

      "total_sweet_snacks": "lambda weekly_counts: weekly_counts.get('dolci',
0) + weekly_counts.get('snack_dolci', 0)",

      "threshold": 3

    },

    "reference": ""

  },

  {

    "name": "Alimenti Ultra-Processati",

    "group": {

      "it": "Alimenti da Limitare",

      "en": "Foods to Limit"

    },

    "enabled": true,

    "metric": "ultra_processati",

    "source": "weekly_counts",

    "operator": ">",

    "threshold_expression": "num_days * 0.5",

    "message": {

      "it": "Hai consumato alimenti ultra-processati {metric_value} volta/e
questa settimana. È fortemente consigliato ridurre drasticamente il loro
consumo in quanto spesso ricchi di zuccheri, grassi e sale.",

      "en": "You have consumed ultra-processed foods {metric_value} time(s)
this week. It is strongly recommended to drastically reduce their consumption
as they are often high in sugars, fats, and salt."

    },

    "message_params": {

      "threshold_calculated": "lambda num_days: num_days * 0.5"

    },
    "reference": ""

  },

  {

    "name": "Consumo di Alcool",

    "group": {

      "it": "Alimenti da Limitare",

      "en": "Foods to Limit"

    },

    "enabled": true,

    "metric": "alcool",

    "source": "weekly_counts",

    "operator": ">",

    "threshold": 0,

    "message": {

      "it": "Hai consumato alcool {metric_value} volta/e questa settimana.
Per una salute ottimale, si raccomanda di limitare al massimo il consumo di
bevande alcoliche, idealmente evitando l'assunzione regolare.",

      "en": "You have consumed alcohol {metric_value} time(s) this week. For
a healthy diet, it is recommended to limit the consumption of alcoholic
beverages, ideally avoiding regular consumption."

    },

    "reference": ""

  },

  {

    "name": "Idratazione Insufficiente",

    "group": {

      "it": "Idratazione",

      "en": "Hydration"

    },

    "enabled": true,

    "metric": "bicchieri_acqua",

    "source": "weekly_counts",
    "operator": "<",

    "threshold_expression": "num_days * 8",

    "message": {

      "it": "Hai bevuto {metric_value} bicchieri d'acqua questa settimana.
L'obiettivo è almeno {target_glasses} bicchieri ({glasses_per_day} al
giorno). Aumenta l'idratazione per migliorare il benessere generale.",

      "en": "You have drunk {metric_value} cups of water this week. The goal
is at least {target_glasses} cups ({glasses_per_day} per day). Increase the
hydration for a better overall health."

    },

    "message_params": {

      "target_glasses": "lambda num_days: num_days * 8",

      "glasses_per_day": 8

    },

    "reference": ""

  }

]
```

9.3 sleep_rules.json

```
[

  {

    "name": "Ore di sonno insufficienti",

    "group": {

      "it": "Stile di vita",

      "en": "Lifestyle"

    },

    "type": "simple",

    "enabled": true,

    "metric": "avg_sleep_hours",

    "operator": "<",

    "threshold": 7,

    "message": {
      "it": "In media dormi solo {metric_value} ore per notte. L'obiettivo è 7-9 ore per migliorare il recupero fisico e mentale.",

      "en": "On average, you sleep only {metric_value} hours per night. The goal is 7-9 hours to improve physical and mental recovery."

    },

    "reference": ""

  },

  {

    "name": "Andare a letto tardi",

    "group": {

      "it": "Stile di vita",

      "en": "Lifestyle"

    },

    "enabled": true,

    "type": "complex",

    "condition": "(sum(1 for d in sleep_data.values() if d.get('bedtime_hour', 22) >= 23.0 or d.get('bedtime_hour', 22) < 5.0) / len(sleep_data)) > 0.3",

    "message": {

      "it": "Sei andato a letto tardi in {late_days} su {total_days} giorni ({late_percentage}%). Cerca di coricarti prima delle 23:00 almeno il 70% delle notti per migliorare la qualità del sonno e il recupero.",

      "en": "You went to bed late on {late_days} out of {total_days} days ({late_percentage}%). Try to go to bed before 11:00 PM at least 70% of the nights to improve sleep quality and recovery."

    },

    "message_params": {

      "late_days": "lambda sleep_data: sum(1 for d in sleep_data.values() if d.get('bedtime_hour', 22) >= 23.0 or d.get('bedtime_hour', 22) < 5.0)",

      "total_days": "lambda sleep_data: len(sleep_data)",

      "late_percentage": "lambda sleep_data: round((sum(1 for d in sleep_data.values() if d.get('bedtime_hour', 22) >= 23.0 or d.get('bedtime_hour', 22) < 5.0) / len(sleep_data)) * 100, 1)"

    },

    "reference": ""
  },

  {

    "name": "Svegliarsi tardi",

    "group": {

      "it": "Stile di vita",

      "en": "Lifestyle"

    },

    "type": "simple",

    "enabled": true,

    "metric": "avg_wakeup_hour",

    "operator": ">",

    "threshold": 8,

    "message": {

      "it": "In media ti svegli alle {metric_value}:00. Cerca di svegliarti
prima delle 8:00 per allinearti meglio con i ritmi circadiani.",

      "en": "On average, you wake up at {metric_value}:00. Try to wake up
before 8:00 AM to better align with your circadian rhythms."

    },

    "reference": ""

  },

  {

    "name": "Caffeina serale",

    "group": {

      "it": "Stile di vita",

      "en": "Lifestyle"

    },

    "type": "simple",

    "enabled": true,

    "metric": "caffeine_after_hour",

    "operator": ">",

    "threshold": 16,
    "message": {
      "it": "Hai consumato caffeina alle {caffeine_after_hour}:00. Evita caffeina nelle 6-10 ore prima di coricarti per non disturbare il sonno.",
      "en": "You consumed caffeine at {caffeine_after_hour}:00. Avoid caffeine 6-10 hours before bedtime to avoid disturbing sleep."
    },
    "reference": ""
  },
  {
    "name": "Alcol serale",
    "group": {
      "it": "Stile di vita",
      "en": "Lifestyle"
    },
    "type": "simple",
    "enabled": true,
    "metric": "alcohol_evening",
    "operator": ">",
    "threshold": 0,
    "message": {
      "it": "Hai consumato alcol in serata. Ridurne l'assunzione può migliorare la produzione di melatonina e la qualità del sonno.",
      "en": "You consumed alcohol in the evening. Reducing its intake can improve melatonin production and sleep quality."
    },
    "reference": ""
  }
]
```

9.4 activity_rules.json

```
[
    {
```

```
    "name": "Steps giornalieri bassi",
    "group": {
      "it": "Stile di vita",
      "en": "Lifestyle"
    },
    "type": "simple",
    "enabled": true,
    "metric": "avg_steps",
    "operator": "<",
    "threshold": 5000,
    "message": {
        "it": "La tua media giornaliera di passi è inferiore a 5.000. L'OMS raccomanda almeno 10.000 passi al giorno. Prova ad aumentare gradualmente l'attività fisica quotidiana.",
        "en": "Your daily average steps are less than 5,000. The WHO recommends at least 10,000 steps per day. Try to gradually increase your daily physical activity."
    },
    "reference": ""
  },
  {
    "name": "Steps giornalieri medi",
    "group": {
      "it": "Stile di vita",
      "en": "Lifestyle"
    },
    "type": "complex",
    "enabled": true,
    "condition": "5000 <= avg_steps < 10000",
    "message": {
        "it": "La tua media giornaliera di passi è tra 5.000 e 10.000. Buon inizio! Cerca di raggiungere l'obiettivo di 10.000 passi al giorno per benefici ottimali sulla salute.",
        "en": "Your daily average steps are between 5,000 and 10,000. Good
start! Try to reach the goal of 10,000 steps per day for optimal health
benefits."

    },

    "reference": ""

  },

  {

    "name": "Giorni sedentari",

    "group": {

      "it": "Stile di vita",

      "en": "Lifestyle"

    },

    "type": "complex",

    "enabled": true,

    "condition": "lambda activity_data: sum(1 for day in
activity_data.values() if day.get('steps', 0) < 1000) >= 3",

    "message": {

        "it": "Hai avuto almeno 3 giorni con meno di 1.000 passi. Questo
indica un comportamento sedentario. Anche una breve camminata può fare la
differenza per la tua salute.",

        "en": "You had at least 3 days with less than 1,000 steps. This
indicates sedentary behavior. Even a short walk can make a difference to your
health."

      },

    "reference": ""

  },

  {

    "name": "Calorie bruciate basse",

    "group": {

      "it": "Stile di vita",

      "en": "Lifestyle"

    },

    "type": "simple",
    "enabled": true,

    "metric": "avg_burned_calories",

    "operator": "<",

    "threshold": 1500,

    "message": {

        "it": "Il tuo consumo calorico medio giornaliero è inferiore a 1.500
kcal, indicando un livello di attività molto basso. Considera di aumentare
l'attività fisica.",

        "en": "Your average daily calorie burn is less than 1,500 kcal,
indicating a very low activity level. Consider increasing physical activity."

    },

    "reference": ""

  },

  {

    "name": "Calorie bruciate medie",

    "group": {

      "it": "Stile di vita",

      "en": "Lifestyle"

    },

    "type": "complex",

    "enabled": true,

    "condition": "1500 <= avg_burned_calories < 2500",

    "message": {

        "it": "Il tuo consumo calorico è moderato (1.500-2.500 kcal/giorno).
Per aumentare il dispendio energetico, prova ad aggiungere attività più
intense.",

        "en": "Your calorie burn is moderate (1,500-2,500 kcal/day). To
increase energy expenditure, try adding more intense activities."

      },

    "reference": ""

  },

  {

    "name": "Biking activity",
    "group": {
      "it": "Stile di vita",
      "en": "Lifestyle"
    },
    "type": "simple",
    "enabled": true,
    "metric": "avg_biking_distance",
    "operator": ">",
    "threshold": 0,
    "message": {
        "it": "Ottimo! Stai includendo il ciclismo nella tua routine. Il
ciclismo è un'eccellente attività cardiovascolare a basso impatto sulle
articolazioni.",
        "en": "Great! You're including cycling in your routine. Cycling is an
excellent cardiovascular activity with low impact on joints."
      },
    "reference": ""
  },
  {
    "name": "Biking distance media",
    "group": {
      "it": "Stile di vita",
      "en": "Lifestyle"
    },
    "type": "complex",
    "enabled": true,
    "condition": "0 < avg_biking_distance < 3000",
    "message": {
        "it": "Le tue sessioni di ciclismo sono brevi (< 3 km in media al
giorno). Prova ad aumentare gradualmente la distanza per migliorare la
resistenza cardiovascolare.",
        "en": "Your cycling sessions are short (< 3 km average per day). Try
to gradually increase the distance to improve cardiovascular endurance."
```

```
      },
    "reference": ""
  },
  {
    "name": "Biking distance media",
    "group": {
      "it": "Stile di vita",
      "en": "Lifestyle"
    },
    "type": "complex",
    "enabled": true,
    "condition": "3000 <= avg_biking_distance < 5000",
    "message": {
        "it": "Buona distanza media di ciclismo (3-5 km al giorno). Mantieni questa routine o aumenta gradualmente per ulteriori benefici cardiovascolari.",
        "en": "Good average cycling distance (3-5 km per day). Maintain this routine or gradually increase for further cardiovascular benefits."
      },
    "reference": ""
  },
  {
    "name": "Biking distance ottima",
    "group": {
      "it": "Stile di vita",
      "en": "Lifestyle"
    },
    "type": "simple",
    "enabled": true,
    "metric": "avg_biking_distance",
    "operator": ">=",
    "threshold": 5000,

    "message": {

        "it": "Eccellente! Le tue sessioni di ciclismo superano i 5 km in
media al giorno. Continua così per mantenere un'ottima forma
cardiovascolare.",

        "en": "Excellent! Your cycling sessions exceed 5 km average per day.
Keep it up to maintain excellent cardiovascular fitness."

      },

    "reference": ""

  },

  {

    "name": "Walking activity",

    "group": {

      "it": "Stile di vita",

      "en": "Lifestyle"

    },

    "type": "simple",

    "enabled": true,

    "metric": "avg_walking_distance",

    "operator": ">",

    "threshold": 0,

    "message": {

        "it": "Bene! Stai includendo sessioni di camminata strutturate. La
camminata è un'attività accessibile e benefica per la salute
cardiovascolare.",

        "en": "Good! You're including structured walking sessions. Walking is
an accessible and beneficial activity for cardiovascular health."

      },

    "reference": ""

  },

  {

    "name": "Walking distance bassa",

    "group": {
      "it": "Stile di vita",
      "en": "Lifestyle"
    },
    "type": "complex",
    "enabled": true,
    "condition": "0 < avg_walking_distance < 2000",
    "message": {
        "it": "Le tue sessioni di camminata sono brevi (< 2 km in media).
Prova ad aumentare gradualmente la durata delle tue passeggiate a 30-40
minuti (circa 3-4 km).",
        "en": "Your walking sessions are short (< 2 km average). Try to
gradually increase the duration of your walks to 30-40 minutes (about 3-4
km)."
      },
    "reference": ""
  },
  {
    "name": "Walking distance media",
    "group": {
      "it": "Stile di vita",
      "en": "Lifestyle"
    },
    "type": "complex",
    "enabled": true,
    "condition": "2000 <= avg_walking_distance < 4000",
    "message": {
        "it": "Buona distanza media di camminata (2-4 km al giorno). Questa è
una buona base per la salute cardiovascolare. Continua così!",
        "en": "Good average walking distance (2-4 km per day). This is a good
foundation for cardiovascular health. Keep it up!"
      },
    "reference": ""
  },
  {
    "name": "Walking distance ottima",
    "group": {
      "it": "Stile di vita",
      "en": "Lifestyle"
    },
    "type": "simple",
    "enabled": true,
    "metric": "avg_walking_distance",
    "operator": ">=",
    "threshold": 4000,
    "message": {
        "it": "Eccellente! Cammini oltre 4 km in media al giorno. Questo
contribuisce significativamente alla tua salute cardiovascolare e al
benessere generale.",
        "en": "Excellent! You walk over 4 km on average per day. This
contributes significantly to your cardiovascular health and overall
well-being."
      },
    "reference": ""
  },
  {
    "name": "Varietà di attività",
    "group": {
      "it": "Stile di vita",
      "en": "Lifestyle"
    },
    "type": "complex",
    "enabled": true,
    "condition": "avg_biking_distance > 0 and avg_walking_distance > 0",
    "message": {
        "it": "Ottimo! Vari la tua routine con diverse attività (ciclismo e
camminata). La varietà aiuta a prevenire la monotonia e coinvolge diversi
gruppi muscolari.",

        "en": "Excellent! You vary your routine with different activities
(cycling and walking). Variety helps prevent monotony and engages different
muscle groups."

      },

    "reference": ""

  },

  {

    "name": "Nessuna attività strutturata",

    "group": {

      "it": "Stile di vita",

      "en": "Lifestyle"

    },

    "type": "complex",

    "enabled": true,

    "condition": "avg_biking_distance == 0 and avg_walking_distance == 0",

    "message": {

        "it": "Non sono state rilevate sessioni di esercizio strutturato
(camminata o ciclismo). Considera di aggiungere almeno 150 minuti di attività
moderata alla settimana.",

        "en": "No structured exercise sessions (walking or cycling) were
detected. Consider adding at least 150 minutes of moderate activity per
week."

      },

    "reference": ""

  }

]
```

9.5 mood_rules.json

```
[

  {

    "name": "Basso Umore Positivo Medio",
    "group": {
      "it": "Umore",
      "en": "Mood"
    },
    "type": "simple",
    "enabled": true,
    "metric": "avg_positive_mood",
    "operator": "<",
    "threshold": 2.5,
    "message": {
      "it": "Il tuo umore positivo medio è {metric_value}/4. Cerca attività
che aumentino le sensazioni di determinazione e ispirazione.",
      "en": "Your average positive mood score is {metric_value}/4. Seek
activities that boost feelings of determination and inspiration."
    },
    "reference": ""
  },
  {
    "name": "Alto Umore Negativo Medio",
     "group": {
      "it": "Umore",
      "en": "Mood"
    },
    "type": "simple",
    "enabled": true,
    "metric": "avg_negative_mood",
    "operator": ">",
    "threshold": 2.0,
    "message": {
      "it": "Il tuo umore negativo medio è {metric_value}/4. Un punteggio
costante sopra 2.0 per 'spaventato', 'nervoso', 'turbato', 'ostile' o
'vergognoso' suggerisce un alto livello di stress. Consulta un professionista
se persiste.",
      "en": "Your average negative mood score is {metric_value}/4. A
consistent score above 2.0 for 'scared', 'nervous', 'upset', 'hostile', or
'ashamed' suggests high stress. Consult a professional if this persists."
    },
    "reference": ""
  },
  {
    "name": "Tendenza all'Ostilità (Aggressività)",
    "group": {
      "it": "Umore",
      "en": "Mood"
    },
    "enabled": true,
    "type": "complex",
    "condition": "(sum(1 for d in mood_data.values() if d.get('ans_8', 1) >=
3.0) / len(mood_data)) > 0.3",
    "message": {
      "it": "Hai riportato un alto senso di 'ostilità' ({ans_8_high_days} su
{total_days} giorni, ovvero {ans_8_high_percentage}%). Questo può indicare
stress e fatica. Prova tecniche di rilassamento per 20 minuti al giorno.",
      "en": "You reported a high sense of 'hostility' ({ans_8_high_days} out
of {total_days} days, or {ans_8_high_percentage}%). This may indicate stress
and fatigue. Try relaxation techniques for 20 minutes daily."
    },
    "message_params": {
      "ans_8_high_days": "lambda mood_data: sum(1 for d in mood_data.values()
if d.get('ans_8', 1) >= 3.0)",
      "total_days": "lambda mood_data: len(mood_data)",
      "ans_8_high_percentage": "lambda mood_data: round((sum(1 for d in
mood_data.values() if d.get('ans_8', 1) >= 3.0) / len(mood_data)) * 100, 1)"
    },
    "reference": ""
  },
  {

    "name": "Bassa Concentrazione",

    "group": {

      "it": "Umore",

      "en": "Mood"

    },

    "type": "simple",

    "enabled": true,

    "metric": "avg_attento",

    "operator": "<",

    "threshold": 2.5,

    "message": {

      "it": "Il tuo punteggio medio per 'attento' (concentrato) è
{metric_value}/4. Una scarsa attenzione può essere collegata a un sonno di
bassa qualità o a stress cronico.",

      "en": "Your average score for 'attento' (attentive/focused) is
{metric_value}/4. Low attention may be linked to poor sleep quality or
chronic stress."

    },

    "reference": ""

  },

  {

    "name": "Alta Ansia (Nervosismo)",

    "group": {

      "it": "Umore",

      "en": "Mood"

    },

    "type": "complex",

    "enabled": true,

    "condition": "(sum(1 for d in mood_data.values() if d.get('ans_6', 1) >=
3.0) / len(mood_data)) > 0.5",

    "message": {
      "it": "Hai riportato alti livelli di 'nervosismo' in oltre la metà dei
giorni ({ans_6_high_percentage}%). Il nervosismo cronico può avere un forte
impatto sulla salute fisica.",

      "en": "You reported high levels of 'nervousness' in over half of the
days ({ans_6_high_percentage}%). Chronic nervousness can strongly impact
physical health."

    },

    "message_params": {

      "ans_6_high_days": "lambda mood_data: sum(1 for d in mood_data.values()
if d.get('ans_6', 1) >= 3.0)",

      "total_days": "lambda mood_data: len(mood_data)",

      "ans_6_high_percentage": "lambda mood_data: round((sum(1 for d in
mood_data.values() if d.get('ans_6', 1) >= 3.0) / len(mood_data)) * 100, 1)"

    },

    "reference": ""

  }

]
```