Deliverable D 5.2

ICT Solutions

Authors: Nikos Paraskevopoulos (ICCS)

Maria Nefeli Kousta (ICCS)

Ioannis Katsipis (ICCS)

Elias Doukas (ICCS)

Dimitris Kalogeras (ICCS)

https://www.dietwise.eu

This project is funded by the European Union's *Horizon 2020 program* under grant agreement No. 101181692

This work is dedicated to the memory of Justina Baršytė, author of the DietWise project idea, whose vision and commitment were invaluable to this project.

Project information

Program:	Horizon Europe
Topic:	HORIZON-CL6-2024-FARM2FORK-01-5
Type of action:	HORIZON-RIA HORIZON Research and Innovation Actions
Grant Agreement #:	101181692
Project title:	Systemic Solutions to Enhance Healthy and Sustainable Food Provision and Cooking at Home
Project Name:	DietWise
Project Start Date:	2024-11-01
Project End Date:	2027-10-31

Document information

Document name:	ICT Solutions
Related Work Package:	WP 5
Related Task:	Task 5.2
Related Deliverables:	D 5.2
Author(s):	Nikos Paraskevopoulos, Maria Nefeli Kousta, Ioannis Katsipis, Elias Doukas, Dimitris Kalogeras
Reviewer(s):	Siegfried Dewitte (KUL), Zivile Kaminskiene (AdC)
Submission date:	2026-04-30
Dissemination level:	Public

Document history

Version	Date	Changes	Responsible partner
v0.1	2026-03-12	TOC	ICCS
V0.2	2026-04-06	Draft ready for internal review	ICCS
V0.3	2026-04-27	Review – Comments and Suggestion	KUL, AdC
V1.0	2026-04-30	Final Version ready for submission	ICCS

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Research Executive Agency (REA). Neither the European Union nor the granting authority can be held responsible for them.

Abbreviations

Abbreviation	Full Form
AI	Artificial Intelligence
LLM	Large Language Model
RCA	Responsible Cooking Alliance
RW	Recipe Watch
ICT	Information and Communication Technology
FBDGs	Food-based dietary guidelines
GBD	Global Burden of Disease
WP	Work Package
RAG	Retrieval-Augmented Generation
PHB	Visiomenes Sveikatos Biuras
VIGL	Vlaams Institut voor Gezondheitdspromotie
IHU	Diethnes Panepistimio Ellados
SAFE	Safe Food Advocacy
PROL	Astiki Mi Kerdoskopiki Etaireia Proliptikis Perivallontikis kai Ergasiakis Iatrikis
VU	Vilniaus Universitetas
GBD	Global Burden of Disease
IAM	Identity and Access Management

Executive Summary

This deliverable presents the developed ICT solutions, shaped through extensive cooperation with stakeholders across all pilot countries and the nutritional experts of the consortium. It outlines the system design, including architecture, technology stack, security specifications, and the domain model shared across the backend and front-ends of both ICT solutions. It provides a detailed analysis of the LLM suggestions pipeline and justifies the design choices made. The document offers an explicit description of the DietWise Renderer and Backend Application, describes both ICT solutions in detail, demonstrates their use scenarios, presents their deployment, and discusses their impact and future development.

Significant progress has been achieved in developing the ICT solutions in a scalable manner. Development was structured across two pillars: a) designing and developing a functional and meaningful AI-powered mechanism for nutrition-related ingredient substitutions in recipes, and b) designing and developing a robust system architecture capable of hosting this mechanism and any future iterations. The architecture was designed around the core principles of scalability, maintainability, and security. The suggestions pipeline follows a staged approach to mimic expert logic and make full use of the capabilities of the available LLMs. In cooperation with the consortium's nutritional experts, a dataset was created to serve as a taxonomy for ingredient-based substitutions. User safety was a key design parameter: no user-specific data is shared with the LLM, as personalization is handled outside LLM inference based on GBD risk factors.

Future efforts will focus on multilinguality, extending the solutions to all three pilot countries: Belgium, Greece, and Lithuania. Further refinement is planned based on stakeholder feedback collected through pilot activities under WP7 (T7.1–3). With a solid foundation in place, the project is well-positioned to incorporate feedback and usage data from the ICT tools to iteratively improve system responses under WP7 (T7.4). While further adjustments are anticipated, the current ICT solutions already establish a comprehensive and robust baseline that addresses identified stakeholder needs and provides a strong foundation for continued development.

Table of Contents

List of Figures

List of Tables

1. INTRODUCTION

This deliverable (D5.2) reports the consolidated outcomes of T5.2 related to the development of the DietWise ICT solutions. It documents the design choices made and implemented - both for the AI-powered reasoning pipeline that generates ingredient alternatives, and for the underlying infrastructure and technical implementation of the two DietWise user-facing applications: MyRecipeWatch (RW), targeting citizens, and the Responsible Cooking Alliance (RCA) tool, targeting food influencers. D5.2 directly contributes to project objective O3 (Digital social innovations and AI-based apps to empower citizens) and supports KPIs 16–21 through the release of the DietWise ICT solutions. It also contributes to project objectives O5 (Engagement through co-creation and citizen science to boost innovations) and O6 (Pilot testing the effectiveness of apps and behavioral interventions) by providing the tools needed to engage citizens and support pilot activities, thus supporting KPIs 21–24.

Commented [ne1]: rename: MyRecipeWatch

This deliverable represents the technical realization of the DietWise ICT solutions and establishes the implementation basis for subsequent pilot testing under WP7. It provides a detailed account of the system architecture, implementation approach, and deployment of the beta versions of both tools.

1.1. PURPOSE AND SCOPE

The purpose of this deliverable is to document the technical implementation and foundational architecture of the DietWise ICT solutions developed under WP5, ensuring alignment with project objectives and stakeholder requirements.

The scope encompasses both ICT solutions - MyRecipeWatch (RW) and the Responsible Cooking Alliance (RCA) tool - and covers their full technical realization. The deliverable describes the system architecture and the rationale guiding its design, including how recipe-level recommendations are operationalized through ingredient-level substitutions and how the AI-driven suggestion pipeline is structured.

In addition, it outlines key considerations related to user safety, including personal data protection and responsible AI usage. A comprehensive overview of the DietWise technical framework is provided, covering shared backend infrastructure, identity management and authentication mechanisms, database separation principles, AI execution components, and recipe extraction and rendering processes.

In line with the structured development phases outlined in the Grant Agreement, this deliverable covers:
(i) the formulation of detailed specifications in collaboration with T4.1;
(ii) the establishment of a robust system architecture;
(iii) the design and implementation of intuitive user interfaces (UI) and user experiences (UX); and
(iv) the deployment of beta versions of both ICT solutions, conducted in collaboration with T4.2.

This deliverable serves both as a technical specification and as a launch document for the beta versions of the ICT solutions, in accordance with the Grant Agreement. It establishes a solid foundation for iterative refinement leading up to M34, when pilot testing activities are expected to conclude.

1.2. INTERCONNECTION WITH OTHER WORK PACKAGES AND DELIVERABLES

This D5.2 is closely linked to WP4 "Development of the DietWise Framework" and Deliverable D4.2 "Co-development of ICT solutions", which served as the integration point where upstream research and evidence were translated into ICT design and implementation decisions. Building on these inputs, D5.2 documents the realization of the resulting ICT solutions.

Commented [ŽK2]: I suggest adding WP and deliverable names OR explain what kind of research.

The findings of WP3 (which focuses on the development and evaluation of behavioural interventions aimed at fostering user adoption and sustained engagement) reported in D3.1 inform both the feature roadmap and the presentation strategies embedded within the ICT solutions - for example, how recommendations for improving recipes are framed, how transparency regarding suggestions is communicated, and how social norms are operationalized to encourage behavioural change. D5.2 supports this integration by documenting the technical context within which these interventions can be implemented and subsequently validated during pilot testing.

Within WP5, D5.2 is primarily associated with Task T5.2, which delivers both ICT solutions. Furthermore, D5.2 provides detailed ICT design and architecture documentation supporting WP7 which aims to pilot test the effectiveness of different tools and applications in combination with the most efficient behavioral interventions and factors such as culture, religion, seasonality, regionality. As such, it plays a key enabling role in ensuring that the ICT solutions are not only aligned with stakeholder needs but also technically mature and ready for pilot deployment.

Commented [ŽK3]: recommendations for improving recipes

Just thinking that for a new reader it might be difficult to follow the thoughts.

Commented [ŽK4]: Whose transparency?

Commented [ŽK5]: You already wrote it in section 1.1.

Commented [ŽK6]: which aims...

ar add name

1.3. METHODOLOGY AND STRUCTURE OF THE DELIVERABLE

This deliverable is structured to present the implemented ICT design and its technical realization in a clear and systematic manner. The methodological approach is reflected in the organization of the document as follows:

- Section 2 describes the overall system design, including architecture, technology stack, security provisions, and the shared domain model across backend and frontend components.
- Section 3 details the AI-powered pipeline for generating ingredient substitutions at the recipe level, including step-by-step logic, prompt design, and underlying rationale.
- Section 4 provides a comprehensive description of the DietWise Renderer, including its purpose and technical specifications.
- Section 5 outlines the backend application, focusing on its architecture and implementation details.
- Section 6 presents the MyRecipeWatch application, including a demonstration scenario.
- Section 7 presents the RCA tool, including a demonstration scenario.
- Section 8 describes the deployment of both ICT solutions.
- Section 9 discusses expected impact and future development directions.

Commented [ŽK7]: Abbreviation is already introduced

This structure ensures a coherent progression from foundational system components to application-level implementation and forward-looking considerations, resulting in a comprehensive and technically robust deliverable.

2. SYSTEM DESIGN

2.1. SYSTEM ARCHITECTURE

In this section, we present the overall architecture of the DietWise technical solution. For the diagrams we utilize the System context[1] and Container[2] level diagrams, as specified by the C4 Model for visualising software architecture[3].

[1] https://c4model.com/diagrams/system-context
[2] https://c4model.com/diagrams/container
[3] https://c4model.com/

Overarching goals

The user-facing aspect of the DietWise ICT solutions consists of two applications:

1. RCA - Responsible Cooking Alliance tool, a browser extension/plugin
2. RW - MyRecipeWatch, a mobile application

The main functionality of both applications is to assess an online recipe and offer healthier, more sustainable alternatives for its ingredients. RW needs to keep statistics for the suggestions it provides, how many users accepted, and how many rejected each suggestion. It displays these statistics and takes them into account when ordering suggestions, so that users feel a sense of co-creation - their choices have the opportunity to affect others within the community.

Commented [ŽK8]: previously it was called application, consistent terminology should be used.

Commented [ni9R8]: We have been referring to it as "tool", added here for consistency

Commented [ŽK10]: You can use the abbreviation which already was introduced

The two applications target distinct user groups and use cases. RW is designed for the general public, whereas RCA is tailored to recipe creators (influencers).

A key requirement for RCA is the ability to operate on content that has not yet been published. For this reason, it is implemented as a browser plugin. Browser plugins can access any page the user is viewing, enabling RCA to analyze draft or password-protected content on an influencer's website, as well as locally stored or unpublished materials. RCA ultimately targets the influencer's entire audience, therefore its recommendations are optimized for broad applicability and do not incorporate individual user preferences. In contrast, RW is user-centric: it provides recommendations tailored to the individual user and therefore relies on personalized data to generate relevant suggestions.At the moment, RCA does not offer functionality specifically tailored to influencers and could, in principle, be made available to the general public. However, it will be kept privileged for influencers only, with the goal of incorporating influencer-specific features in the context of the Responsible Cooking Alliance initiative.

Architectural principles

1. Modularity and Componentization: The building blocks are loosely coupled, highly cohesive to increase maintainability, allow independent deployment, and facilitate upgrades without impacting the whole system.
2. API-First Design: Modelling the system began with its external API. The API, by which we mean both the data model and the service interfaces, is a first-class citizen in the software modules of the backend (see Maven modules `dietwise-model`, `dietwise-service-interfaces` under the `dietwise-architecture` container in the code).
3. Scalability and Performance: Design for flexibility and future growth, ensuring the system can handle increased loads without downtime or major re-engineering. The backend services are stateless precisely for this reason.
4. Best practices for security and incorporating it into the development lifecycle (e.g., by including and running the OWASP Maven plugin).
5. Standardization & Interoperability: DietWise uses standard protocols for its communications. In our case it is only JSON over HTTPs (often called REST). It uses the industry-proven OAuth2 protocol for authenticating users across devices.
6. Code design struggles to adhere to the SOLID principles.
7. Testability is a design goal, try to cover the important paths with tests
8. Asynchronicity baked-in utilizing the latest tools (reactive from the API to the DB access), thus ensuring high performance.

Application design

The design of the DietWise backend follows the *Ports and Adapters* architectural pattern (also known as *Hexagonal Architecture*), a widely adopted and well-established architectural pattern for building modular, maintainable, and testable backend systems. This architectural style promotes a clear separation between the core business logic and external systems, ensuring that the application remains modular, testable, and adaptable to change.

Commented [ŽK11]: , a widely adopted and well-established architectural pattern for building modular, maintainable, and testable backend systems.

At its core, this architectural pattern isolates the domain logic from (a) infrastructure concerns such as user interfaces, databases, and third-party services consumed by the application and (b) from the way the application is accessed from the outside world.

This document provides more information about the implementation specifics of the design in the section that describes the implementation of the backend.

Core principles of Ports and Adapters

The Ports and Adapters architecture is based on the following principles:

(a) Separation of Concerns

The system is divided into distinct layers:

- Core Domain (Business Logic): Contains the application's essential rules and use cases. This layer is independent of external technologies.
- Ports (Interfaces): Define how the core interacts with the outside world. These are abstract contracts that describe required or provided functionality.
- Adapters (Implementations) Provide concrete implementations of the ports for specific technologies (e.g., REST APIs, databases, external services).

This separation ensures that changes in external systems do not impact the core business logic.

(b) Dependency Inversion

Dependencies always point inward toward the core domain. The core defines interfaces (ports); external components implement those interfaces (adapters). This means that (a) the domain does not depend on frameworks or infrastructure, and (b) infrastructure depends on the domain. As a result, the system is resilient to changes in external technologies.

(c) Technology Agnosticism

Ideally, the core application logic in Ports and Adapters is completely independent of UI frameworks (mobile app, browser extension), backend frameworks, identity providers (e.g., Keycloak), databases, and external APIs. This allows the application to evolve or replace technologies without affecting business logic.

However, while the core domain is designed to remain largely independent of external frameworks, a limited number of carefully selected technologies are used within the core to improve developer productivity and code clarity.

Specifically:

- Contexts and Dependency Injection[4] (CDI) is used for dependency injection, limited to constructor-based patterns. This ensures that components remain decoupled from the runtime container and can be instantiated in isolation for testing or executing in alternative environments.
- Mutiny[5] is used to model asynchronous and reactive workflows. As reactive execution is a fundamental characteristic of the system, this dependency reflects an architectural choice rather than an infrastructure concern.
- Jakarta Bean Validation[6] is used within the domain model to declaratively express validation constraints, improving readability and ensuring consistency of business rules.

These technologies were selected because they are lightweight, widely adopted, and based on stable specifications or well-supported libraries. Their usage does not compromise the overall architectural goal of isolating business logic from infrastructure concerns. Rather, it represents a pragmatic balance between purity and maintainability.

Benefits for DietWise

The adoption of the Ports and Adapters architecture provides several key advantages.

In terms of maintainability, changes in one part of the system (e.g., switching the authentication provider or database) do not propagate to the core logic. The core logic remains readable and focused on the implementation of business flows, not littered with specifics pertaining to external systems or infrastructure.

In terms of testability, core business logic can be tested in isolation (in line with the "Testability is a design goal" principle), without needing full system integration. External dependencies can be mocked via ports, i.e. mock implementations of the port interfaces.

[4] https://jakarta.ee/specifications/cdi/
[5] https://smallrye.io/smallrye-mutiny/
[6] https://beanvalidation.org/

System context diagram

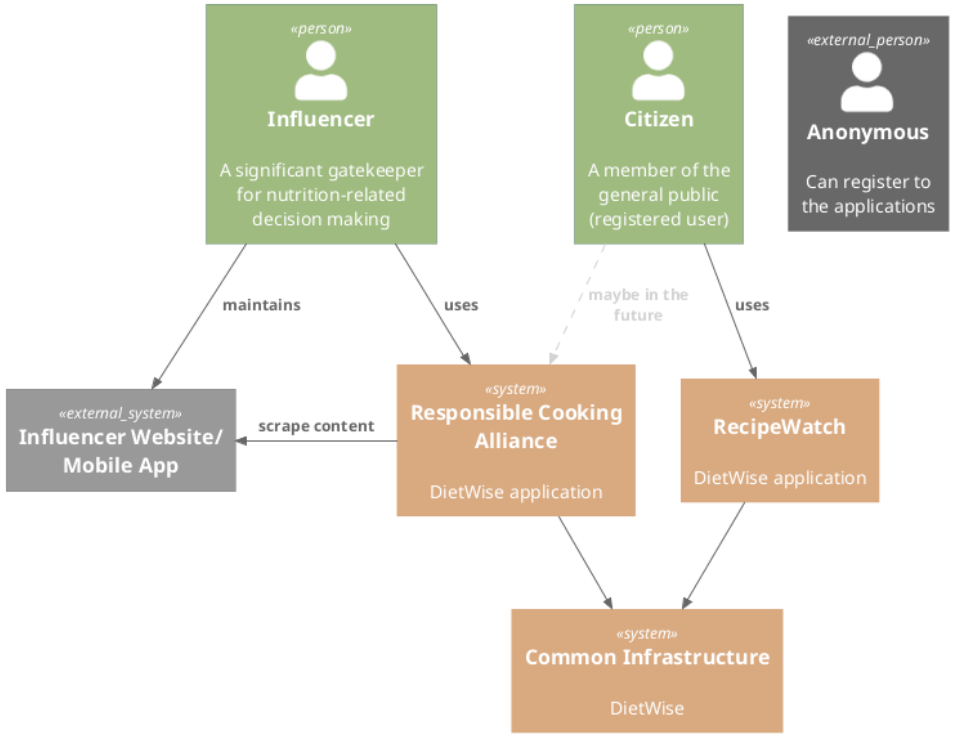


Figure 1 System context diagram

We clearly identify the two user roles plus the capabilities of the anonymous user:

- Influencers are owners of one or more external content platforms (e.g., websites or blogs) and are granted access to RCA.
- Citizens use RW to get personalized advice.
- Anonymous users can freely sign up with MyRecipeWatch.

An important decision to document here: DietWise does not store recipes long term. There may only be some short-term caching of external content. By not storing external content for the long term, we avoid any copyright issues. By always fetching content from the source, we guarantee that the suggestions refer to the latest content, incorporating all current changes. On the other hand, short-term caching (minutes or hours) may be used selectively to improve system performance, reduce latency, and help limit unnecessary repeated requests to external websites, thereby avoiding excessive traffic, particularly for high-demand sources.

Commented [ŽK12]: or?

Commented [ŽK13]: Suggestion to slightly update the framing:

‹ … › is used selectively to improve system performance and reduce latency. It also helps limit unnecessary repeated requests to external websites, thereby avoiding excessive traffic, particularly for high-demand sources.

Authentication

The application leverages the OAuth 2.0 authorization framework in combination with Keycloak[7] as the central Identity and Access Management (IAM) solution. This approach provides a secure, scalable, and standards-compliant authentication mechanism across both delivery channels, the mobile application (RW) and the browser extension (RCA).

Commented [SD14]: Add to abbreviations list?

By delegating authentication responsibilities to a dedicated identity provider, the system ensures a clear separation of concerns between business logic and security-critical processes.

The following sections summarize the reasons for adopting this technology.

(a) Adoption of Industry Standards

OAuth 2.0 is a widely adopted, battle-tested, industry-standard authorization framework used by major platforms and services worldwide. It has proven its reliability through extensive real-world usage. It is compatible with a wide ecosystem of libraries and tools. These characteristics make it very easy to integrate with third-party systems, if required in the future. Overall, adherence to standards reduces long-term technical risk and avoids vendor lock-in.

(b) Elimination of Sensitive Credential Handling

A key architectural decision is that the application does not directly handle user credentials (usernames and passwords). Instead, it lets the IDM handle all credentials while the application only receives and handles secure tokens.

This approach reduces the risk of credential leakage, minimizes the application's security attack surface, and simplifies compliance with security best practices and regulations.

[7] https://www.keycloak.org/

Container diagram

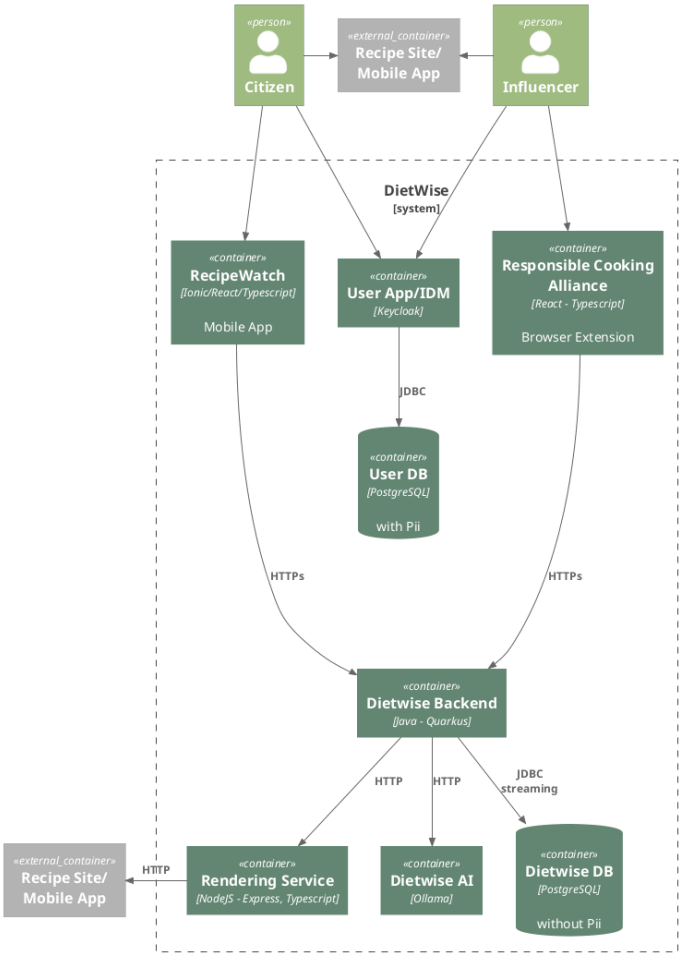


Figure 2 System container diagram

A container in C4 is an independently deployable program. The following is a description of each container.

User App/Identity Manager (IDM)

Keycloak was chosen to handle user authentication, registration, and management. It is a well-established, open-source IAM solution that offers many advantages. It has built-in support for the chosen authentication protocol, OAuth2 and OpenID Connect providing a multitude of robust authentication and authorization capabilities. As a very popular open-source solution with extensive enterprise adoption, it benefits from an active community that provides extensive documentation, examples, and support. In addition, its ongoing development and regular releases demonstrate sustained commitment to maintenance, feature enhancements, and security updates.

Commented [ŽK15]: <…>, Keycloak benefits from an active community that provides extensive documentation, examples, and support. In addition, its ongoing development and regular releases demonstrate sustained commitment to maintenance, feature enhancements, and security updates.

A secondary advantage is that Keycloak is built on the same platform as the DietWise backend, i.e., Quarkus. The ability to deploy Keycloak instances on premises fits the chosen deployment model.

Keycloak's maturity ensures stability, long-term maintainability, and access to a rich feature set without requiring custom development. The fact that the user database, the only DB storing personally identifiable information, Pii, is separate from the main DB further limits the possibility of leaking sensitive data.

Avoidance of Custom Authentication Implementation
Implementing authentication mechanisms from scratch is complex and error prone. By using Keycloak, the system avoids the need to develop and maintain user registration flows, login/logout mechanisms, credentials storage, account recovery (e.g., password reset), user management (activating/deactivating accounts, assigning roles), multi-factor authentication (if ever needed in the future), integration with external identity providers (e.g., Facebook, if ever needed in the future). This significantly reduces development effort while improving overall security posture.

MyRecipeWatch and Responsible Cooking Alliance applications

Commented [ŽK16]: terminology

Commented [ni17R16]: Both applications and tools apply.

Both these applications comprise the DietWise front-end. They are user interfaces deployed in their respective app stores. There are separate chapters for each, so no more details will be included here.

DietWise backend

This is the central application that executes the business logic of DietWise (most importantly the recipe assessment flows). It is presented in its own chapter.

Rendering Service (DietWise renderer)

The Rendering Service or DietWise Renderer is a small HTTP service that fetches recipe web pages in a real but headless browser, captures the fully rendered HTML, and transforms that HTML into a cleaner representation suitable for downstream processing. It is designed as an internal backend component for DietWise, where the main backend can delegate webpage rendering and content extraction to a controlled server-side environment.

The Rendering Service was introduced because the mobile app (MyRecipeWatch) cannot retrieve the content of web pages displayed within its an in-app browser due to security restrictions of the mobile runtime environment. The browser plugin (RCA) does not have this restriction. The HTML simplification code is common and runs in the plugin, in the case of RCA.

The Rendering Service is fully documented in its own section.

DietWise AI

AI-related functionalities are executed on a dedicated server, separate from the main backend server of the system. The functionality is made available to the application through an Ollama server. The AI server utilizes two graphics cards for acellerating AI processing, one Nvidia A40 48GB VRAM and an older Nvidia Tesla T4 16 VRAM.

DietWise DB

PostgreSQL is used as the primary relational database of both the DietWise backend and Keycloak. It stores both application reference data, such as recommendations and suggestion rules, and user-specific operational data, such as personal information and usage statistics.

2.2. TECHNOLOGY STACK

The implementation uses well-known and industry-standard solutions. The chosen technologies are:

Technology	Version	Purpose
Java	25	Most of the server-side components
Quarkus[8]	3.32	The main server infrastructure
NodeJS	24	Runs the renderer, also for development of the front-ends
Express[9]	5.2	The server infrastructure for NodeJS (the renderer)
Ollama[10]	0.17	Server that runs and exposes AI services
PostgreSQL	18	The general relational storage
Keycloak[11]	26.5	The Identity Manager
Ionic[12]	8	Application infrastructure for cross-platform mobile applications
React[13]	19	For all the UIs (both Ionic and the browser extension)
TypeScript[14]	5 (*)	For all NodeJS/browser/Ionic applications See comment below
Vite[15]	8	Building the three JS/TS projects
Vitest[16]	4	Writing and running tests for the three JS/TS projects

Table 1 Technologies and versions used

We strive to use the latest long-term support version for all components, where applicable. We upgrade regularly to stay up-to-date and mitigate any vulnerabilities or bugs in the infrastructure.

TypeScript version: As this project was in the final stages of release, TypeScript 6 was released. Many dependencies, however, did not support it yet, so we chose to remain on the previous version 5.

2.3. SECURITY

Overall, the implementation has considered security by incorporating best practices, adding AI to the code review cycle, and using tools to detect potential risks. Last but not least, the dependencies of each project are frequently updated to the latest versions that solve security issues.

Commented [ŽK18]: ?

Commented [ni19R18]: Online Word hiccups, thanks!

Backend

Centralized authentication through OpenID Connect

The application uses Quarkus OIDC integration and relies on an external identity provider for user authentication. The backend does not implement password handling, password storage, or local credential validation itself.

The security benefits of this design include: authentication is delegated to a dedicated identity platform, the application never stores or handles user passwords locally, and token handling follows a standard OIDC/JWT integration model.

[8] https://quarkus.io/
[9] https://expressjs.com/
[10] https://ollama.com/
[11] https://www.keycloak.org/
[12] https://ionicframework.com/
[13] https://react.dev/
[14] https://www.typescriptlang.org/
[15] https://vite.dev/
[16] https://vitest.dev/

Application-specific authorization enforcement

After authentication, the backend replaces the default security context with an application-specific context that carries the DietWise user model and roles. This allows authorization to be enforced consistently in business services, close to where the logic is implemented not only at the HTTP boundary. It also allows for dedicated unit tests of the authorization logic.

Defensive HTTP response headers

The configuration adds security headers to reduce common browser-side risks:

- Referrer-Policy: no-referrer
- Permissions-Policy: disabling browser access to sensitive features such as camera, microphone, and geolocation
- X-Content-Type-Options: nosniff
- Strict-Transport-Security: in production with one-year duration and subdomain coverage

For the browser extension callback page, additional protections are applied:

- Cache-Control: no-store, max-age=0
- Pragma: no-cache
- X-Frame-Options: DENY
- A restrictive Content-Security-Policy that denies all content by default and blocks framing and form submission

Together, these headers reduce exposure to content-type confusion, clickjacking, caching of sensitive callback data, and browser feature misuse.

Restricted Cross-Origin Resource Sharing configuration

Cross-origin access is enabled only for explicitly configured origins. The default configuration limits access to approved browser-extension origins and a specific localhost origin, while development adds only a small set of known local and project environments.

SSRF-oriented validation for externally supplied recipe URLs

The URL of the recipe page to assess is a serious security risk. If not handled and sanitized carefully, it can expose the backend to server-side request forgery (SSRF), where an attacker tricks the application into fetching internal or non-public resources instead of a legitimate recipe page. This can lead to unauthorized access to all kinds of internal services. It can also allow abuse of the rendering pipeline through malformed, unexpected, or credential-bearing URLs. Careful validation of scheme, host, and address type is implemented to ensure that external fetching is limited to legitimate public HTTP(S) targets only.

The backend-side validation, however, is only the first barrier. The renderer was hardened too, to reduce SSRF and browser-abuse risk, even as an internal service. Strict input validation was applied for the requested URL, allowing only http/https full URLs for live rendering plus the test URLs that match the exact pattern `^[0-9]{3}\.html$`. Requests to localhost, private IP ranges, loopback, link-local, and hostnames that resolve to internal addresses are rejected; The same validation is applied not only to the initial page URL, but also to outbound browser requests for scripts, images, iframes, and other subresources. Non-GET top-level navigations triggered by page scripts or auto-submitting forms are blocked. Downloads, notifications, geolocation, microphone/camera access, screen capture, and clipboard APIs of the headless browser are explicitly disabled. Each render runs in a fresh browser context, with no persistent session state carried across jobs. Cookies, local/session storage, IndexedDB, cache storage, and service workers are prevented from persisting. Outbound browser requests are limited to a reduced header set, with cookies and referrer

Commented [SD20]: In general, I am not sure when an acronym makes it to the list or not. It is never used beyond this paragraph, perhaps no need to acronymize then either? Not sure. Not to oimportant either.

Commented [ni21R20]: Yes, this is so specific and focused to this paragraph that I agree it is redundant to mention in the abbreviations list.

information stripped. In combination, these changes constrain the renderer to its intended role: fetch and render external pages without being usable as a general-purpose internal browser or write-capable network client.

Input validation, SQL injection, controlled error handling

Bean validation is applied to request DTOs where required; for example, statistics requests require a non-null suggestion identifier. The system sanitizes AI and renderer outputs before further use. Internal exceptions are logged server-side, while client responses remain intentionally limited to reduce accidental exposure of stack traces, internal class names, SQL details, or infrastructure information. The persistence layer uses Hibernate Reactive, entity lookups, and JPA criteria queries. There is no direct string-built SQL or ad hoc native-query construction in the application.

OWASP, SpotBugs and FindSecBugs

We have included the OWASP Maven plugin[17] to report any vulnerabilities in the dependencies of the project and SpotBugs[18] + FindSecBugs[19] to identify potential security-related problems in the code. Any issues reported from SpotBugs + FindSecBugs have been fixed, and false positives have been excluded in its configuration file (`config/spotbugs/exclude.xml`).

The OWASP report run on April 30th, 2026 has 2 findings:

	Dependency	Vulnerability IDs	Highest Severity
1	plexus-utils-3.6.1.jar	cpe:2.3:a:codehaus-plexus:plexus-utils:3.6.1:*:*:*:*:*:*:*[20] cpe:2.3:a:utils_project:utils:3.6.1:*:*:*:*:*:*:*	HIGH
2	quarkus-hibernate-validator-spi-3.34.6.jar	cpe:2.3:a:hibernate:hibernate-validator:3.34.6:*:*:*:*:*:*:* cpe:2.3:a:quarkus:quarkus:3.34.6:*:*:*:*:*:*:*	MEDIUM

Table 2 OWASP findings

Mitigations/comments:

1. This is a build time dependency. Building is done locally (i.e. develoment machine, no build server). No real threat.
2. This warning is about the SafeHtmlValidator. Not used in DietWise. False positive in our case.

Essentially, OWASP reports that our code has no dependency on libraries with known vulnerabilities.

MyRecipeWatch

The key security points for MyRecipeWatch were the authentication flow, token handling and local persistence, configuration for Android/iOS, transmission of authenticated requests, logging of potentially sensitive information, and exposure to client-side script injection such as XSS.

[17] https://dependency-check.github.io/DependencyCheck/dependency-check-maven/index.html
[18] https://spotbugs.github.io/
[19] https://find-sec-bugs.github.io/
[20] https://nvd.nist.gov/vuln/detail/CVE-2025-67030

Authentication flow

Authentication is based on OAuth/OpenID Connect, requests are made using bearer tokens. PKCE has been enabled for the OAuth authorization flow. This is an appropriate security model for a mobile application because it separates identity management from the application itself and allows tokens to be issued, refreshed, and validated through a dedicated identity provider.

Token handling and local persistence/Mobile platform configuration

MyRecipeWatch stores tokens and other application state on the device, in secure data storage rather than open web storage such as localStorage or sessionStorage. This is a positive design choice because it reduces casual exposure of sensitive state within browser-like storage surfaces. In addition, Android OS backups have been disabled. This prevents local application data from being copied through standard Android backup channels, which is a sensible precaution for an application that handles authenticated sessions and user-specific data.

Logging And Operational Data Exposure

Sensitive logging is now restricted to development builds. This is important because logs can become an unintended secondary storage channel for authentication event payloads, error objects containing authentication context, and user personalization or profile data. All these pieces of information are logged in development mode. Restricting such logs to development environments only reduces the risk of exposing sensitive information through device logs, support workflows, or external monitoring systems.

Client-Side Injection Exposure

The implementation does not contain XSS-prone rendering patterns in the main application flows. In particular, no obvious unsafe HTML injection points can be identified in the reviewed code paths for recipe data, personalization data, or authentication-related content. The review was carried out with the help of AI.

Responsible Cooking Alliance

The security measures taken are identical to those of MyRecipeWatch.

2.4. THE MODEL

A domain model was developed for the needs of DietWise that is shared across the backend and the two front-ends. It is inspired by the recipe model of schema.org, but simplified a lot to reflect the needs of this project. The central types are presented here in alphabetical order.

Domain types

Ingredient		
Field name	Type (Optional)	Comments
id	IngredientId	
nameInRecipe	String	
triggerIngredient	TriggerIngredient (O)	
roleOrTechnique	RoleOrTechnique (O)	

Table 3 Type Ingredient

Recipe		
Field name	Type (Optional)	Comments
name	String	
recipeYield	String (O)	Unused
recipeIngredients	List of Ingredient	

recipeInstructions	List of String	Each element is a simple String representation of an execution step. Used to extract the role of the ingredients in the recipe.
text	String (O)	The raw textual representation of the recipe in the source page.

Table 4 Type Recipe

Commented [ŽK22]: Is this sentence cut off?

Commented [ni23R22]: Online Word hiccups, thank you!

Rule extends RuleData		
An ingredient substitution rule; representation in code of the data that triggers a substitution		
Field name	Type (Optional)	Comments
id	RuleId	
recommendation	Recommendation	A GBD recommendation
roleOrTechnique	RoleOrTechnique	
cuisineContext	String	

Table 5 Type Rule – extends RuleData

ScoringData		
Data required to calculate the score/rating/ranking for a recipe and update it dynamically in the client		
Field name	Type (Optional)	Comments
totalNumberOfRecomendations	int	
recommendationWeights	Map<RecommendationComponentName, RecommendationWeight>	
recommendationsPerIngredient	Map<IngredientId, Set<RecommendationComponentName>>	

Table 6 Type ScoringData

SuggestionTemplate		
A configured suggestion for replacing a single ingredient, as identified by a Rule object, with a single alternative; also stores additional information about the substitution		
Field name	Type (Optional)	Comments
id	SuggestionTemplateId	
alternative	AlternativeIngredient	
restriction	String (O)	
equivalence	String (O)	
techniqueNotes	String (O)	

Table 7 Type SuggestionTemplate

Suggestion extends SuggestionTemplate		
A suggestion transmitted to the front-end to be displayed to the user		
Field name	Type (Optional)	Comments
target	AppliesTo	The target ingredient
ruleId	RuleId	Id of the rule that triggerred the suggestion
recommendation	Recommendation	A GBD recommendation
rationale	String (O)	An explanation about this substitution, optionally provided by the AI

alternativeComponentNames	Set<RecommendationComponentName>	
totalSuggestionStats	SuggestionStats	Statistics about this suggestion for all users
userSuggestionStats	SuggestionStats	Statistics about this suggestion for the current user
text	String (O)	A human friendly message describing the substitution

Table 8 Suggestion – extends SuggestionTemplate

Value types

In Domain-Driven Design (DDD), a Value Object is an immutable type that is defined by its attributes rather than by identity. In DietWise, we specialize this definition by characterizing "Value types" as simple data structures containing at most a few fields that are not central to the business logic. So, a Recipe is included in the Domain types even though it has no identity, because it contains more than a few fields, and additionally, it is central to the system's logic. We have chosen to use specialized value types for things that can be represented as a simple type, e.g., a String or int, to make the intent of their usage explicit in all contexts. For example, a mapping of user ids to emails could be represented in Java by a `Map<UUID, String>` type. In this case, it is not clear what does the UUID key and the String value represent without context. With explicit types, this becomes `Map<UserId, Email>` which is self-explanatory. The following table summarizes the simple value types:

Type name	Base simple type	Role
AlternativeIngredient	String	The name of an alternative ingredient for a substitution
IngredientId	UUID	Id of a known ingredient
Recommendation	String	A recommendation from the GBD
RecommendationComponent Name	String	The name of the component/ingredient that affects a GBD recommendation. E.g., for the recommendation "Decrease red meat", the name of the component that affects it is "red meat"
RecommendationWeight	enum	Whether an ingredient is beneficial in the context of DietWise (ENCOURAGED) or harmful (LIMITED)
RoleOrTechnique	String	The "Role or Technique" column from
RuleId	UUID	Id of a specified rule
SuggestionTemplateId	UUID	Id of a suggestion template
TriggerIngredient	String	Name of a trigger ingredient, from the specifications Excel

Table 9 Simple value types

SuggestionStats		
Statistics about a suggestion		
Field name	Type (Optional)	Comments
timesSuggested	int	

timesAccepted	int	
timesRejected	int	

Table 10 Value type SuggestionStats

3. THE LLM SUGGESTIONS PIPELINE

This section describes the end-to-end pipeline developed to generate ingredient substitution suggestions for a given recipe. Large Language Models (LLMs) exhibit inherent limitations, including the potential for inconsistent or inaccurate outputs. Although fine-tuning a specialized LLM agent could help mitigate these issues, such an approach typically requires large, carefully curated datasets -resources that were not available within the scope of this project. To address these constraints, we adopted a structured, pipeline-based approach. Rather than relying on end-to-end generation, we decomposed the task into a series of well-defined steps that mimic the substitution logic provided by domain experts. This design enables more reliable guidance for the model throughout the process. Furthermore, the pipeline was designed to explicitly control both the context provided to the model and the format of its outputs. Expert nutritional guidelines were directly embedded into the prompting strategy, ensuring that the generated recommendations remain aligned with domain-specific requirements. The resulting system combines deterministic algorithmic filtering with LLM-based components to produce evidence-based, contextually appropriate substitution suggestions grounded in GBD dietary risk factor data and an expert-defined substitutions dataset. The pipeline processes each recipe ingredient through seven sequential steps, alternating between LLM-driven tasks and deterministic operations. Figure 3 provides a high-level overview of the workflow.

The system takes a structured recipe as input and produces a ranked set of ingredient substitution suggestions, each grounded in an evidence-based rule, adapted to the ingredient's culinary function (role/technique), and prioritized according to the user's demographic risk profile. The pipeline decomposes into a per-ingredient reasoning loop - comprising five LLM invocations and deterministic database operations - followed by a cross-ingredient GBD prioritization step. The central design principle is a strict separation of concerns: the first two LLM invocations are closed-vocabulary classifiers that feed the deterministic retrieval step, and the rest invocations perform a selection thus introducing a bounded generative judgment, anchored to validated database entries. The database is designed to provide high-level epidemiological principles, enforcing evidential grounding on the LLM's pre-trained culinary and nutritional knowledge.

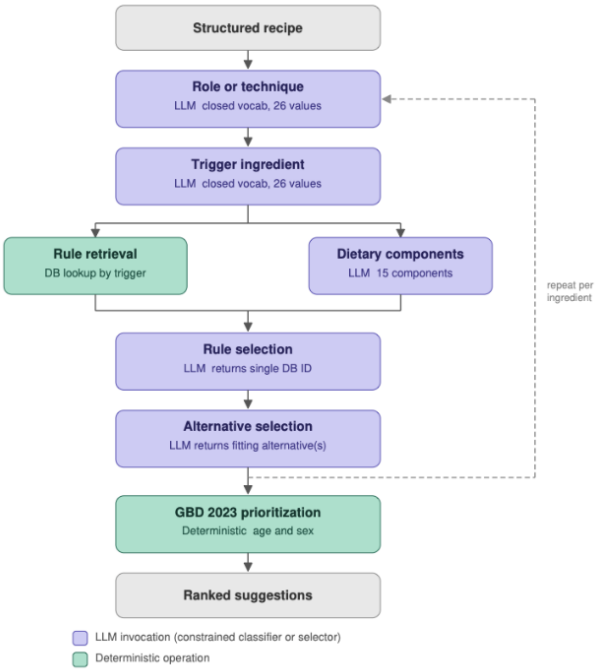


Figure 3 Overview of the per-ingredient reasoning pipeline. Purple nodes denote LLM invocations operating under closed-vocabulary constraints; teal nodes denote deterministic operations. The parallel branch executes rule retrieval and dietary component classification concurrently. The loop iterates over all ingredients before the GBD prioritization step.

3.1. INGREDIENT ROLE CLASSIFICATION

The same ingredient can serve fundamentally different purposes depending on how it is used. Olive oil used as a sauté fat is functionally different from olive oil used as a baking fat or as a finish oil. Classifying the role precisely ensures that the later steps of the pipeline have the necessary context. For each ingredient in the recipe, the LLM is called with:

- The full list of allowed role/technique values (controlled vocabulary),
- The ingredient name,
- The recipe preparation steps.

The model selects exactly one value from the controlled vocabulary. If no value clearly matches, it outputs unknown. This single-value constraint prevents ambiguity in the downstream trigger-matching step. In the system prompt, we have added examples as a few-shot learning technique to help the LLM better understand the task at hand and avoid cases of inconsistent output. The full system prompt (with examples) can be found in the Annex E LLM Prompts.

SYSTEM PROMPT

```
You are a classification model.

Task: determine the role or technique of an ingredient in a recipe.
```

Commented [SD24]: Looks funny to me but I take it that this is the way it is done (-:

Commented [ni25R24]: Yeap looks strange but it is common practice to "ground" the LLM's behavior.

```
Context: This classification feeds a lookup system. The RoleOrTechnique value will be used
to match this ingredient against a database of food alternatives. Choose the value that
most precisely describes how THIS ingredient functions in the recipe — not the dish
category, not surrounding ingredients. Precision matters: a wrong role will retrieve
irrelevant alternatives.

You are given:
- a list of allowed RoleOrTechnique values
- an ingredient
- the recipe instructions

You must choose the single best matching value from the list of allowed RoleOrTechnique
values.

Strict output rules:
- Output EXACTLY one value from the list of allowed RoleOrTechnique values.
- Output only the value.
- Do not output explanations.
- Do not output punctuation.
- Do not output quotes.
- Do not output multiple values.
- Do not invent new values.
- If no value clearly matches, output: unknown

Critical classification rule:
- Classify based on what THIS ingredient does in the recipe, not what is done around it.
- If an ingredient is cooked IN fat, it is not the fat itself.
- If an ingredient is added as a garnish at serving, it is a topping, not a protein or
sauce.
```

USER PROMPT TEMPLATE

```
Allowed RoleOrTechnique values:
{availableRolesAsMarkdownList}

ingredient: {ingredientNameInRecipe}

instructions:
{instructionsAsMarkdownList}

Select the RoleOrTechnique value.

Output only the value.
```

Table 11 Summary of LLM input parameters for role classification.

Parameter	Value / Description
{availableRolesAsMarkdownList}	A Markdown-formatted list of all permitted role/technique values drawn from the expert-defined substitution dataset.
{ingredientNameInRecipe}	The ingredient name as it appears in the recipe (e.g., "butter", "olive oil", "feta cheese").
{instructionsAsMarkdownList}	The preparation steps of the recipe are formatted as a Markdown list.

3.2. MATCH INGREDIENT TO TRIGGER

The substitution database is indexed by trigger ingredient labels (e.g., "White pasta", "General fat choice", "Minced meat"). This step bridges the ingredient as it appears in the recipe to the canonical label used in the database. It takes both the ingredient name and its role/technique classification as input, so that context-dependent ingredients (e.g., olive oil used as a finish oil vs. a sauté fat) can be mapped to the most appropriate trigger.

For each ingredient, the LLM is called with:

- The full list of allowed trigger ingredient labels,
- The ingredient name,
- The role/technique value assigned.

The model selects exactly one trigger label. If no label clearly matches the ingredient - for example, if the ingredient is a vegetable, herb, or spice not covered by the database - the model outputs "unknown" and the ingredient is excluded from further processing. This avoids forced matches that would retrieve irrelevant alternatives. In the system prompt, we have added examples as a few-shot learning technique to help the LLM better understand the task at hand and avoid cases of inconsistent output. The full system prompt (with examples) can be found in Annex E LLM Prompts.

SYSTEM PROMPT

```
You are a classification model.

Task: Classify the ingredient into ONE trigger ingredient value.

Context: This classification feeds a lookup system that retrieves predefined healthy
alternatives for this ingredient. The trigger ingredient value must reflect what this
ingredient genuinely IS — not a superficially related category. If the ingredient does not
closely match any value, output unknown. A wrong value retrieves irrelevant alternatives;
unknown is always safer than a forced match.

You will be given:
- the allowed trigger ingredient values
- an ingredient name
- the ingredient's role/technique in a recipe

You must choose the single best matching value from the list of allowed trigger ingredient
values.

Strict output rules:
- Output EXACTLY one value from the allowed list.
- Output only the value.
- Do not output explanations.
- Do not output punctuation or quotes.
- Do not output multiple values.
- Do not invent new values.
- If no value clearly matches, output: unknown.
- If the ingredient is salt, olive oil or water output: unknown.
```

USER PROMPT TEMPLATE

```
Allowed trigger ingredient values:
{availableTriggerIngredientsAsMarkdownList}

ingredient: {ingredientNameInRecipe}
```

```
roleOrTechnique: {ingredientRoleOrTechnique}

Select the trigger ingredient value.

Output only the value.
```

Table 12 Summary of LLM input parameters for trigger classification.

Parameter	Value / Description
{availableTriggerIngredientsAsMarkdownList}	A Markdown-formatted list of all trigger ingredient labels present in the substitution dataset.
{ingredientNameInRecipe}	Ingredient name as it appears in the recipe.
{ingredientRoleOrTechnique}	Role/technique value assigned to this ingredient in Step 1.

3.3. CONTEXT FILTERING

The substitution dataset contains expert-defined mappings between trigger ingredients and their healthier alternatives, along with the corresponding recommendation (grounded in GBD study), cuisine context, restrictions, equivalence and technique notes. Context filtering reduces this full dataset to only those entries relevant to the current ingredient, ensuring that the LLM operates on a targeted subset rather than the full database.

For each ingredient that received a valid (non-unknown) trigger label:

- The substitution database is queried for all rows where the Trigger Ingredient column matches the assigned label.
- The matching rows are assembled into a filtered context block.
- Ingredients mapped to unknown trigger are dropped at this stage; no rows are retrieved for them.

The filtered entries are passed to the LLM as structured context for selecting the best recommendation.

3.4. DETERMINE INGREDIENT COMPOSITION

In parallel with 3.3. CONTEXT FILTERING, a third LLM invocation matches the ingredient's dietary components against the 15 GBD 2023 risk factors (e.g., red meat, sodium, whole grains). The resulting component set is used in the later step to break ties between candidate rules.

For each ingredient, the LLM is called with:

- the ingredient name,
- the list of allowed components.

The model selects the appropriate values from the controlled vocabulary. Note that an ingredient may match more than one dietary component. In the system prompt, we have added examples as a few-shot learning technique to help the LLM better understand the task at hand and avoid cases of inconsistent output. The full system prompt (with examples) can be found in Annex E LLM Prompts.

SYSTEM

```
You are a constrained classifier.

Task: determine the composition of an ingredient that is part a recipe from a list of
components.

You are given:
- a list of allowed component names with an optional explanation in parentheses
- the name of the ingredient

You must choose all the components that apply to the ingredient.

Strict output rules:
- Output one component per line.
- Output only the component name, not the explanation.
- Do not output explanations.
- Do not output punctuation.
- Do not output quotes.
- Do not output multiple values in the same line.
- Do not invent new values.
```

USER PROMPT TEMPLATE

```
Components:
{availableRecommendationsAsMarkdownList}

Ingredient: {ingredientNameInRecipe}
```

Table 13 Summary of LLM input parameters for determining ingredient composition.

Parameter	Value / Description
{availableRecommendationsAsMarkdownList}	The list of GBD 2023 dietary components.
{ingredientNameInRecipe}	Ingredient name as it appears in the recipe.

3.5. DETERMINE BEST FITTING RECOMMENDATION

The fourth invocation performs a selection. Given the:

- pre-filtered rule set (ID, recommendation, trigger and role/technique), and
- ingredient's role/technique, trigger, and dietary components,

the model returns exactly one rule ID. Selection criteria are applied in priority order: role/technique match first, dietary component relevance second, with a defined first-entry fallback if neither criterion yields a clear winner. The model operates on a closed, pre-verified list and returns an identifier - an auditable, database-verifiable output. In the system prompt, we have added examples as a few-shot learning technique to help the LLM better understand the task at hand and avoid cases of inconsistent output. The full system prompt (with examples) can be found in Annex E LLM Prompts.

SYSTEM PROMPT

```
You are a selection model.

Task: identify the single best fitting rule from a list of filtered database entries for a
given ingredient.

Context: This selection feeds a lookup system. The chosen rule id will be used to retrieve
```

```
predefined healthy alternatives for the ingredient. The entries have already been pre-
filtered by trigger ingredient — your task is to rank them by fit and return the id of the
best match. Precision matters: a wrong rule retrieves irrelevant alternatives.

You are given:
- the ingredient name as it appears in the recipe
- the ingredient's role or technique in the recipe
- the ingredient's dietary components
- a list of filtered database entries, each with an id, a recommendation and a role or
technique

You must choose the single best matching entry and output its id.

Selection criteria — apply in this order:
1. Role or technique match: prefer the entry whose role most closely matches the
ingredient's roleOrTechnique.
2. Dietary component relevance: if still tied, prefer the entry whose recommendation is
most relevant to the ingredient's dietaryComponents.

Strict output rules:
- Output EXACTLY one id from the list of filtered database entries.
- Output only the id value.
- Do not output explanations.
- Do not output punctuation.
- Do not output quotes.
- Do not output multiple values.
- Do not invent new values.
- If no entry clearly matches on any criterion, output the id of the first entry in the
list.
```

USER PROMPT TEMPLATE

```
ingredient: {ingredientNameInRecipe}
roleOrTechnique: {ingredientRoleOrTechnique}
triggerIngredient: {triggerIngredient}
dietaryComponents: {dietaryComponents}
Filtered db entries: {FilteredRulesMarkdownList}

Select the id of the best fitting entry.
Output only the id.
```

Table 14 Summary of LLM input parameters for selecting the best fitting recommendation.

{ingredientNameInRecipe}	Ingredient name as it appears in the recipe.
{ingredientRoleOrTechnique}	Role/technique value assigned to this ingredient in Step 1.
{triggerIngredient}	Trigger value assigned to this ingredient in Step 2.
{dietaryComponents}	Matched dietary components assigned to this ingredient in Step 4.
{FilteredRulesMarkdownList}	Filtered set of rules from Stage 3.

3.6. SUGGEST ALTERNATIVE

Using the rule ID from the previous step, the database returns the corresponding candidate alternatives. Each carries a name, optional restriction annotations (e.g., 'Not suitable if gluten-free'), equivalence notes, and

technique notes. The fifth LLM invocation evaluates the candidate set of alternatives and returns the most suitable ones. Candidates whose restriction annotations conflict with the ingredient's role are omitted. This mechanism ensures that even when the LLM exercises contextual judgment, every suggestion remains anchored to a validated evidential source.

Again, in the system prompt, we have added examples as a few-shot learning technique to help the LLM better understand the task at hand and avoid cases of inconsistent output. The full system prompt (with examples) can be found in Annex E LLM Prompts.

SYSTEM PROMPT

```
 You are a culinary nutrition assistant.

Task: given an ingredient that needs to be substituted in a recipe, select the best
alternatives and return them.

Context: You are the final step of a healthy eating recommendation pipeline. A curated
expert database has already been consulted and a set of candidate alternatives has been
retrieved. Your job is to evaluate these candidates and return the most suitable ones.
You are given:
- the ingredient name as it appears in the recipe
- the ingredient's role or technique in the recipe
- a list of candidate alternatives, each with a name, an optional explanation, and
optional restrictions
- optional equivalence notes (quantity / ratio guidance)
- optional technique notes (cooking method adaptations)

You must return the suitable alternatives, according to the restrictions and the role or
technique of the ingredient to be replaced.

Output rules:
- Output ONLY the name of each suitable alternative.
- Return between 1 and 3 alternatives. Prefer fewer, higher-confidence results over many
uncertain ones.
- Omit candidates that have a restriction that makes them clearly unsuitable given the
role or technique.
- Do not output null values — use an empty string "" if a field has no content.
- One alternative name per line.

Classification rules:
- The alternative is ALWAYS a name from the candidate list. Never invent a value here.
```

USER PROMPT TEMPLATE

```
We need to substitute the ingredient {ingredientNameInRecipe}.
Its role in the recipe is {ingredientRoleOrTechnique}.
The allowed substitutes are:
{alternativesAsMarkdownList}
```

Table 15 Summary of LLM input parameters for suggesting alternatives.

Parameter	Value / Description
{ingredientNameInRecipe}	Ingredient name as it appears in the recipe.

{ingredientRoleOrTechnique}	Role/technique value assigned to this ingredient in Step 1.
{alternativesAsMarkdownList}	Filtered set of candidate alternatives.

3.7. RANK SUGGESTIONS

After the per-ingredient loop completes, a deterministic prioritization step ranks all generated suggestions by their GBD 2023 risk contribution for the user's specific age group and sex. This step operates independently of the LLM: personalization is a lookup against validated epidemiological data, not a function of model inference about the user. See 5.8 SUGGESTION PRIORITIZATION for the technical details.

Table 16 summarizes each step, its type (algorithmic or LLM action), and its inputs and outputs.

#	Step	Type	Input	Output
1	Role Classification	LLM	Ingredient + recipe instructions	role/technique label
2	Trigger Matching	LLM	Ingredient + role/technique	trigger ingredient label
3	Context Filtering	Algorithmic	Trigger label + expert substitution database	Filtered set of expert-defined recommendations
4	Determine Ingredient Composition	LLM	Ingredient	Components of ingredient
5	Determine best fitting recommendation	LLM	Role/technique + trigger + components + filtered set of expert-defined recommendations	Best fitting recommendation
6	Load expert-defined alternatives related to the identified recommendation	Algorithmic	Best fitting recommendation	Filtered alternatives
7	Guided Suggestions	LLM	Role/technique + trigger + components + filtered alternatives	Contextualised substitution suggestions
8	Rank Suggestions	Algorithmic	All ingredient substitutions produced for the recipe + GBD risk factor contributions by sex and age	Ranked ingredient substitutions

Table 16 Step-by-step breakdown of the hybrid LLM–algorithmic pipeline for generating contextualized, evidence-based ingredient substitution recommendations.

3.8. DESIGN DECISIONS AND RATIONALE

Several architectural choices in the pipeline warrant explicit justification. The placement of role/technique classification as the first stage reflects a judgment that culinary function is a stronger predictor of substitution relevance than ingredient identity alone: the same ingredient can require fundamentally different substitutions depending on how it is used, and retrieving rules against identity alone would produce candidates that are nutritionally motivated but culinarily misaligned. Resolving this dimension before any database interaction is therefore a prerequisite for correct retrieval, not an optional refinement.

Viewed across all stages, the constraint profile forms a deliberate gradient. At every stage, the scope of what the LLM may output is defined in advance so that generative freedom is permitted only within boundaries. Every user-facing suggestion must derive from the expert-defined database, ensuring no recommendation exists without an evidential anchor. This reflects the intended division of labor between the expert-defined dataset and the LLM: the LLM applies the expert-defined principles contextually, deriving a fitting suggestion from the step-by step high-level guidance.

The pipeline deploys two models at different tiers, matched to the cognitive demand of each stage. Model selection was informed by early-stage experimentation in which Llama-family models demonstrated stronger instruction-following compared to alternatives tested. Both models are open-weight and can be deployed on-premise. The lighter model, Llama 3.1 8B, is assigned to recipe parsing, rule pre-filtering, and rule selection. These tasks are structurally constrained as the output space is narrow and the decision is guided by exact-match or near-exact-match criteria. The heavier model, Llama 3:70B, handles role and technique classification, trigger ingredient identification, dietary component matching, and alternative suggestion. These stages require the model to reason about culinary function, nutritional semantics, and contextual fit simultaneously, and a classification error at any of them propagates through the remainder of the pipeline in ways that cannot be corrected by later stages. Assigning greater capacity precisely where the output is both semantically richer and failure-consequential reflects a principled allocation of inference cost, rather than a uniform choice of model across the board.

Finally, GBD-based prioritization is applied after the per-ingredient loop rather than within it. This separates correctness concerns from relevance concerns, ensures the LLM prompts carry no user-specific information, and allows the same pipeline outputs to be re-ranked for any demographic profile without re-invoking any LLM stage - a property that matters for both efficiency and auditability in health-adjacent deployment.

4. THE DIETWISE RENDERER

4.1. PURPOSE

DietWise Renderer is a stateless HTTP service that turns a recipe page URL into one or more machine-friendly outputs. Its primary responsibility is to render the target page and retrieve its content in a way that matches what the user sees, as accurately as possible. To do that, it uses a real but headless browser that executes any code that would produce dynamic content. It returns the final rendered HTML, optionally simplifying it into a compact representation for downstream LLM or parser use. It can return cleaned-up HTML or go a step further and convert the cleaned-up HTML into Markdown. In practice we found that for the needs of DietWise it always has to simplify all the way and produce Markdown. Finally, it extracts `Recipe` objects from JSON-LD scripts embedded in the page. It returns the JSON-LD recipes along with the cleaned text.

Running as a server-side component means that it does not get the opportunity to interact with the user. One consequence is that if the page requires a password, the renderer does not have access to it. This is a known limitation of the system.

The service is built as a Node.js + TypeScript application using Express for the HTTP layer and Playwright/Chromium for page rendering. The build system is Vite with Vitest for testing.

4.2. MAIN COMPONENTS

HTTP server

The application starts in `src/server.ts`. On startup it reads runtime configuration from environment variables, creates a `BrowserPool` with a fixed number of Chromium processes, creates a `Semaphore` to cap

concurrent render jobs, and finally it initializes the Express app and listens on the port defined by the environment variable `PORT` **(3000 by default).**

Express application

The app wiring in `src/app.ts` is intentionally small. It registers two HTTP endpoints:

HTTP method	URI	Function
GET	`/health`	service health checks
POST	`/render`	page rendering and optional transformation

Table 17 Endpoints of the DietWise renderer

Browser pool

The browser pool in `src/util/BrowserPool.ts` maintains a fixed number of headless Chromium instances. Each request borrows one browser and creates an isolated browser context for that specific render operation. This design separates the browser process count, controlled by `BROWSER_COUNT`, from the active page job count, controlled by `MAX_CONCURRENT_JOBS`. This separation allows the service to reuse browsers while still limiting total concurrent work.

Concurrency gate

The semaphore in `src/util/Semaphore.ts` prevents too many requests from executing at once. If the concurrency limit is reached, new requests will wait until an active job is complete.

Cleaner and extractor pipeline

After rendering, the /render handler can run two optional post-processing steps: JSON-LD extraction from `src/cleaner/extractJsonLdRecipes.ts` or HTML simplification from `src/cleaner/cleanHtmlForLLM.ts`. These steps are controlled by independent flags in the request, so consumers can request raw rendered HTML, simplified output, structured recipe data, or both.

4.3. REQUEST FLOW

A POST /render request follows this lifecycle:

1. The request enters the Express route in `src/routes/render.ts`.
2. The route acquires a semaphore slot before starting browser work.
3. The route validates the request body and extracts the render options.
4. If DW_RENDERER_TEST_DIR is set and the url looks like 123.html, the service loads a local fixture file instead of visiting the network.
5. Otherwise, it acquires a browser from the shared browser pool.
6. It creates a new Playwright browser context with a canonical desktop-like profile:
    a. fixed user agent
    b. en-US locale
    c. America/New_York timezone
    d. normalized viewport
7. It opens a new page, navigates to the target URL, and waits for `domcontentloaded`.
8. It checks the main document HTTP status. Responses >= 400 are treated as request failures.
9. It waits an additional 1 second to allow late DOM updates before capturing `page.content()`.
10. If enabled, it extracts JSON-LD recipe data from the HTML.
11. If enabled, it simplifies the HTML or converts it to minimal text.

12. It returns the final JSON response.
13. The browser context is always closed in a `finally` clause, and the semaphore slot is released.

Commented [ŽK26]: missing object?

Commented [ni27R26]: No this was intentional :)
I tried to make it more clear -> "finally clause". It is a programmatic construct that tries to guarantee that some code is run even in the event of error.

4.4. TIMEOUT AND FAILURE MODEL

Each request has two timeout layers: a Playwright navigation timeout taken from the request timeout field, defaulting to 15000 ms and a hard service-level timeout from application config, currently 20000 ms in `src/server.ts`.

If the hard timeout triggers, the service attempts to save a screenshot to `/tmp/dietwise-renderer/<timestamp>.png`, closes the browser context, and replaces the underlying browser instance in the pool. This is a defensive recovery mechanism in the case of stuck or broken browser state.

Errors are returned as JSON. The route maps HTTP fetch failures and common network-resolution/connection errors to HTTP error code 400 (client error) and other unexpected failures to 500 (server error).

4.5. RENDERING PROFILE

The service deliberately renders pages with a stable browser profile rather than inheriting environment defaults. The current profile has a fixed, Chrome-like user agent, non-mobile desktop behavior, `deviceScaleFactor` of 1, fixed timezone. The stable browser profile reduces output variance across hosts and requests.

Viewport is normalized through the function `normalizeViewport()` to a width clamped to 320..1440 pixels and a height clamped to 480..2000 pixels. The exact dimensions of the viewport are an optional request parameter; if not supplied, the default is 1440x900.

4.6. HTML SIMPLIFICATION PIPELINE

The simplification logic is designed for deterministic downstream consumption, not visual fidelity. To be more precise, what is needed is the final, fully rendered HTML content. Having obtained this, the cleaner performs the following operations:

1. Removes scripts, styles, iframes, forms, SVG/canvas, headers, footers, nav, asides, ads, and hidden UI.
2. Removes or unwraps non-whitelisted tags.
3. Strips nearly all attributes.
4. Keeps only safe `href` values on links.
5. Removes media by default.
6. Removes empty elements after cleanup.
7. Normalizes whitespace and block boundaries.
8. Optionally emits minimal plain text/Markdown instead of HTML.

The cleaner also runs consent-banner removal heuristics from `src/cleaner/removeConsentUI.ts`, targeting common CMP vendors and generic cookie/consent overlays.

For recipe-oriented simplification, the route uses the RECIPE_MINIMAL_TAGS whitelist and sets `keepTables: false`, `dropMedia: true` to keep the output minimal. The outcome has a compact structure centered on headings, paragraphs, lists, and links.

4.7. JSON-LD EXTRACTION

JSON-LD[21] (JavaScript Object Notation for Linked Data) is a lightweight, W3C-standardized format used to structure data on web pages, making it easily understandable by search engines and machines. It is recommended by Google for Developers[22]. It is not a visible element of the page, but it seems to be present in the code of most pages from dedicated recipe sites we tested.

The JSON-LD extractor parses `script[type^="application/ld+json"]` blocks and looks for Recipe schema objects. It supports top-level recipe objects, arrays with schema.org context, nested `@graph` structures, and nested recipe-like objects under fields such as `mainEntity`, `itemListElement`, and `about`. The structure of a recipe in JSON-LD is documented in schema.org[23].

The extractor normalizes results into a simpler internal Recipe shape with the fields `name`, `recipeYield`, `recipeIngredients`, `recipeInstructions`. It also tolerates inconsistent publisher data by falling back across common schema variants.

4.8. TEST FIXTURE MODE

For repeatable local testing, the route supports a fixture path mode controlled by the environment variable `DW_RENDERER_TEST_DIR`. When enabled, a request with e.g. `url: "001.html"` loads its input from `<DW_RENDERER_TEST_DIR>/001.html`, without contacting the real site. JSON-LD extraction and simplification still run as normal. This is useful for testing the cleaner, validating extraction behavior, and reproducing site-specific parsing issues without live network dependencies.

4.9. API DETAILS

Please see Annex C.

4.10. OPERATIONAL CONFIGURATION

Environment variables:

Name	Purpose	Default
`PORT`	HTTP listen port	3000
`BROWSER_COUNT`	number of Chromium processes in the pool	2
`MAX_CONCURRENT_JOBS`	maximum simultaneous render jobs	4
`DW_RENDERER_TEST_DIR`	optional path containing canned response files for local testing	

Table 18 Operational configuration of the DietWise renderer

Current hard timeout configuration is internal rather than environment-driven with `hardTimeoutMs = 20000`

4.11. COMPONENT DIAGRAM

This is a diagram of the important code components of the renderer and their interconnections.

[21] https://json-ld.org/
[22] https://developers.google.com/search/docs/appearance/structured-data/intro-structured-data
[23] https://schema.org/Recipe

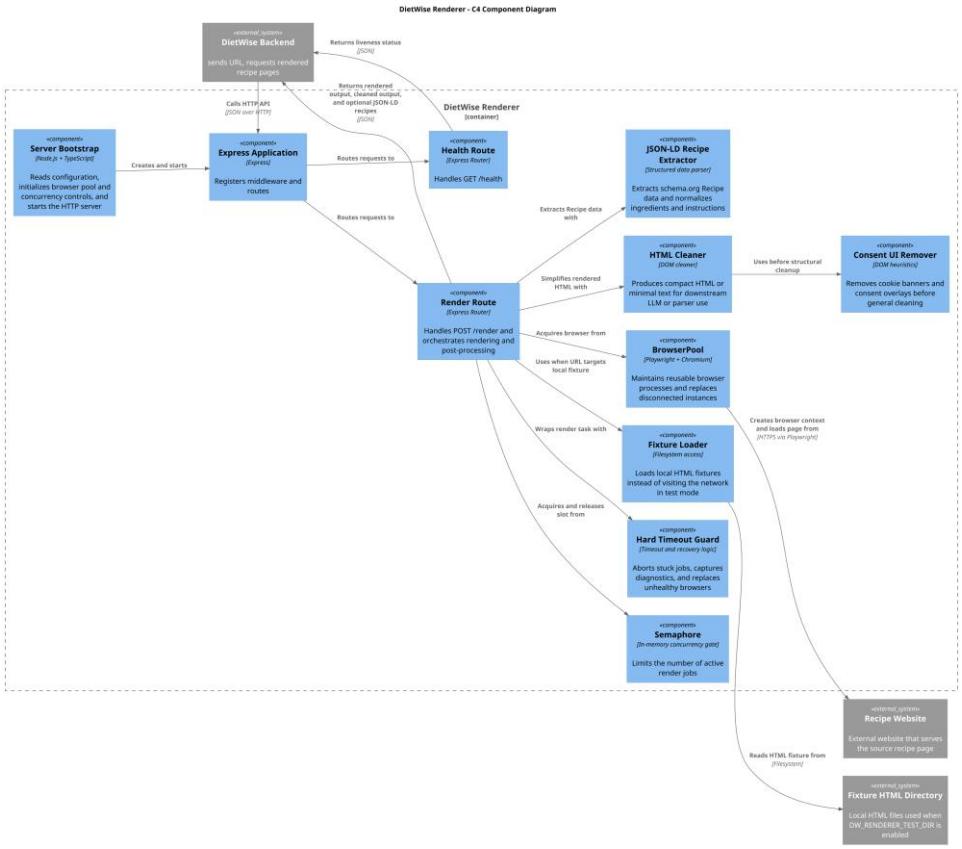


Figure 4 Software component diagram of the DietWise renderer

4.12. DESIGN CHARACTERISTICS

The rendering service has a few explicit design properties:

- It is not exposed to the outside world, so plain HTTP is good enough.
- Stateless request handling: no persistent application state beyond the browser pool.
- Isolated browser contexts per request: cookies, storage, and page state do not leak across renders.
- Deterministic output shaping: cleaner rules are structural and aggressive by design.
- Operational resilience: crashed browsers are replaced automatically, and stuck jobs trigger hard recovery.
- Testability: fixture mode allows offline reproduction of cleaning and extraction behavior.

5. THE BACKEND APPLICATION

5.1. ARCHITECTURE AND IMPLEMENTATION

The DietWise backend follows a modular, layered backend architecture implemented as a Maven multi-module monorepo. The design separates stable contracts from runtime-specific implementations: domain models and service interfaces are defined in shared architecture modules, while executable runtime concerns are implemented in dedicated container modules. This keeps the core application model relatively independent from infrastructure details such as Quarkus, database access technology, and deployment packaging, in accordance with the Ports and Adapters model discussed earlier.

At runtime, the application is implemented as a Quarkus-based service exposing versioned REST APIs. The overall style is service-oriented within a single deployable backend. The number of operations does not warrant further split into finer-grained microservices. Business capabilities such as recipe assessment, personal information management, suggestion generation, and user statistics are organized into distinct service areas with clear responsibilities. This supports maintainability and makes it easier to evolve individual functional areas without tightly coupling all parts of the codebase.

A key architectural principle is the separation of concerns across layers. An API layer handles HTTP and request/response concerns; the service layer orchestrates business workflows; and the data access layer encapsulates persistence logic. Shared domain types are reused across these layers through explicit interfaces and model contracts. This reduces leakage of persistence and framework-specific details into business logic and helps preserve a clean application structure.

Another important principle is contract-first modularization. Core domain models, API-facing data structures, and service interfaces are defined in reusable modules that act as architectural boundaries. Implementation modules depend on these contracts, not the other way around. This improves testability, supports clearer ownership of responsibilities, and enables the system to evolve while keeping its public and internal contracts stable.

The backend also adopts reactive programming as a design choice for I/O-heavy workloads. Reactive types are used across service and persistence boundaries, which is consistent with the application's need to coordinate multiple external interactions, including database access, authentication context handling, and AI-assisted processing. This style is particularly suited to workflows such as recipe extraction and assessment, where several asynchronous steps need to be chained and streamed to the UI progressively.

The persistence architecture reflects another clear principle: abstraction over implementation. DAO interfaces are defined separately from their Hibernate Reactive implementations, allowing database access logic to remain isolated and replaceable in principle. Database schema evolution is managed explicitly through Liquibase changelogs, which introduces traceability and repeatability in structural data changes.

The application also shows a pragmatic integration-oriented style. It is designed to combine conventional backend capabilities with AI-assisted processing, using specialized services for recipe extraction, ingredient-role identification, trigger matching, and scoring support. These AI-related concerns are encapsulated behind dedicated service abstractions rather than being spread across the codebase. This approach limits coupling to a specific AI provider and keeps the orchestration logic within the application layer.

From a design perspective, the system favors explicitness and operational clarity over hidden conventions. Configuration for infrastructure dependencies such as database access, OIDC authentication, and AI endpoints is externalized. Authentication is integrated into the request pipeline and adapted into an

application-specific security context, ensuring that downstream services work with application-level user models instead of raw framework principals.

Overall, the architectural style can be described as a modular monolithic backend with layered responsibilities, explicit contracts, reactive execution, and integration-focused orchestration. The guiding design principles are separation of concerns, clear module boundaries, stable shared contracts, infrastructure isolation, and pragmatic support for AI-enhanced processing within a maintainable enterprise backend structure.

5.2. BACKEND MODULE STRUCTURE

The top-level dietwise-parent Maven module contains four child modules:

The dietwise-architecture contains the shared domain model and service interfaces. It is also the place to include any documentation. The idea behind the architecture module is to define an abstract, high-level view of the application's functionalities and data. These can be used by many "clients." The implementation code benefits most from this representation, but it can prove useful to the client applications and end-to-end tests. Since the JSON representation of the model is ubiquitous, we have added a Maven module that defines the mapping of the model to JSON using Jackson annotations.

Commented [ŽK28]: I suggest simplifying:
The implementation code benefits most from this representation, but client applications and end-to-end tests can also benefit.

Commented [ni29R28]: Thanks, applied slightly modified.

The dietwise-common module contains common technical building blocks, including shared types, utilities, reactive persistence abstractions, and test support.

The dietwise-container houses the bulk of the DietWise-specific implementation code. It contains the backend implementation, including REST resources, business services, DAO implementations, the Quarkus application executable module, and database migrations.

Finally, the dietwise-docker module defines Docker-related artifacts used for local and peripheral environment setup, like the database and Keycloak.

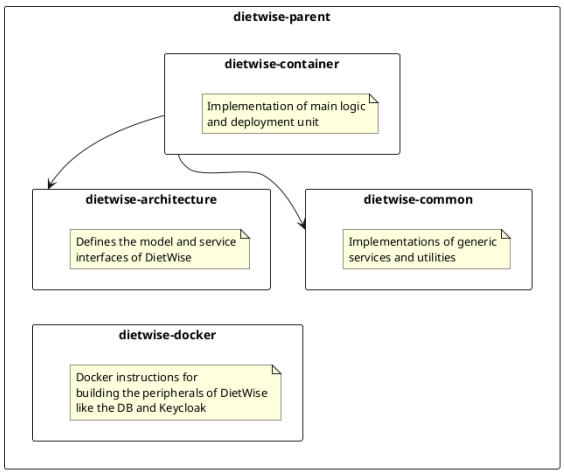


Figure 5 Backend module structure

5.3. STRUCTURE OF THE DIETWISE-CONTAINER MODULE

The dietwise-container module is the place where the bulk of the code that implements the DietWise functionality is located. It is also the place where the Ports and Adapters principles are applied. It consists of six child modules.

The dietwise module is a Quarkus application. This module assembles all the implementation modules required by the backend into a deployable unit. It depends directly on dietwise-services, dietwise-jaxrs, and dietwise-dao-hibernate-reactive.

The module dietwise-services implements the service interfaces defined in dietwise-architecture/dietwise-service-interfaces. These are the services exposed to the outside world, constituting the core of DietWise. More services implement parts of the functionality in a composable and componentized way. It will be described in more detail in later sections of this document.

The module dietwise-jaxrs implements the JSON over HTTP API of the application using JAX-RS.

Finally, dietwise-dao-hibernate-reactive implements the data access layer (DAO) of DietWise using Hibernate Reactive. The JPA entities that persist the application's domain are visible only in this module. No JPA-related code leaks out of this module; this decoupling allows easy mocking of the DB layer for testing and the freedom to replace the persistence logic with any other technology.

Commented [ŽK30]: Sounds a bit weird. Maybe "are restricted to this module".

Commented [ni31R30]: Hmmm neither is exactly what I meant. I think the current "are visible only in this module" is clearer, still not fully satisfied, but can't come up with something better.

Two more modules exist inside the dietwise-container: the dietwise-services-model contains data objects that do not need to be exposed in the domain of the application but still need to be shared between the implementation packages. Last, dietwise-dao defines the DAO interfaces used by the core logic to access the persistent store of the application. It is the *port* to the database, and dietwise-dao-hibernate-reactive is the *adapter*.

There are more implementations of the Ports and Adapters in dietwise-services. The client for the DietWise Renderer and the clients for the AI services. They are implemented with a clear separation between interface and implementation, but not in their own module.

We have to mention that most modules depend on the core or common models. This relation is not shown in the next diagram to keep it clean.

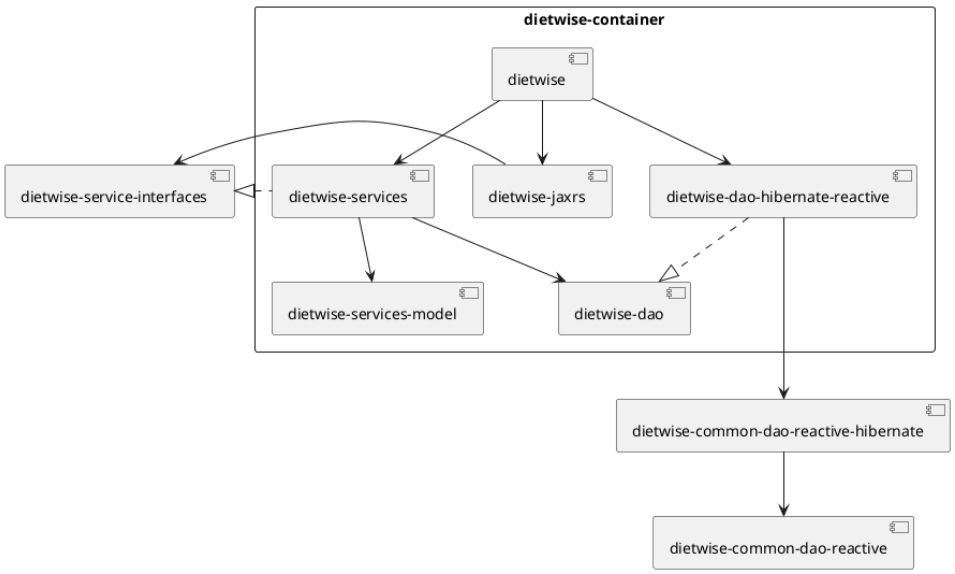


Figure 6 Structure of the dietwise-container backend module

5.4. API ARCHITECTURE AND VERSIONING

The DietWise backend exposes its functionality through a REST-based HTTP API implemented in Quarkus using JAX-RS. The API layer is responsible for receiving client requests, validating and deserializing payloads, resolving the authenticated user context, and delegating execution to the appropriate application services. This keeps HTTP concerns separate from business logic and allows the service layer to remain focused on orchestration and domain behavior. This separation also means that if the need arises, the domain services can easily be exposed in a different transport technology.

The API design follows a versioned structure. Versioning is reflected in the Java package organization and in the API layer itself, with the current implementation centered on version v1. This approach provides a stable contract for client applications while allowing future API evolution without forcing disruptive changes on existing consumers. It also creates a clear boundary between externally exposed contracts and internal implementation details.

The current API surface is organized around the main business capabilities of the backend. This means that the JAX-RS endpoints expose the services defined by the architecture `dietwise-service-interfaces` module.

Commented [ŽK32]: Something is missing here?

Commented [ni33R32]: Nope, "dietwise-service-interfaces" is the technical module name. Changed to code style font, perhaps it is more distinguishable.

A notable characteristic of the API architecture is that not all endpoints are simple request-response operations. Some flows (currently only the recipe assessment) are designed as progressive interactions that can emit multiple messages during processing. The backend's reactive architecture is a perfect fit for this. Progressive interactions support use cases where the client benefits from incremental delivery of processing results, such as extraction output, suggestion generation, and scoring of a recipe.

The API layer also incorporates security adaptation as part of request processing. Authentication is handled through OpenID Connect (OIDC) integration, and the framework-provided identity is transformed into an

application-specific user representation before business services are invoked. This ensures that downstream components work with a consistent domain-level security model rather than low-level token details. The `DietwiseAuthenticationFilter` is responsible for this transformation.

Overall, the API architecture is designed around four principles: clear separation between transport and business logic, stable versioned contracts, alignment with business capabilities, and compatibility with reactive execution patterns. This makes the API suitable both for current client applications and for controlled future evolution of the platform.

5.5. SERVICE LAYER DESIGN

The service layer is the central orchestration layer of the DietWise backend, where application behavior is assembled end-to-end, while persistence and transport remain encapsulated behind narrower interfaces. A key design principle is the separation between service contracts and service implementations.

The service design follows the application's reactive programming model. Service methods use reactive types to compose asynchronous operations such as database access, authentication-related processing, AI inference calls, and progressive result emission. This is especially relevant for longer-running flows such as recipe assessment, where the system can produce multiple intermediate or final messages rather than waiting for a single blocking result. The service layer is therefore not only the business orchestration layer, but also the main composition layer for reactive execution.

Another important aspect is the distinction between domain-oriented services and supporting technical or non-domain services. Domain-oriented services implement application capabilities visible to clients, such as handling personal information or updating statistics. Supporting services handle cross-cutting or infrastructural concerns, such as user resolution, time-related support, renderer interaction, or AI-facing helper functionality. This distinction keeps the core use-case services focused while still allowing the runtime implementation to integrate external dependencies cleanly.

Overall, the service layer design is centered on explicit orchestration, clear service boundaries, contract-driven development, and reactive workflow composition. This makes it the architectural core of the backend and the primary location where the platform's business capabilities are implemented in a structured and maintainable way.

5.6. AI INTEGRATION

The DietWise backend includes an AI integration layer that supports selected application workflows requiring interpretation of unstructured or semi-structured food-related content, both as input from the web and as output from the LLMs. This integration is implemented within the service layer. Its role is to assist business workflows such as recipe extraction, ingredient-role identification, trigger matching, and other AI-assisted assessment tasks.

From a technology perspective, the integration is built on quarkus-langchain4j, with requests targeting Ollama-hosted language models. This provides a structured way to connect the Quarkus application to LLM-based capabilities while keeping the integration consistent with the rest of the application's dependency injection, configuration, and runtime model. The backend therefore treats AI services as application-managed components rather than ad hoc external calls.

Each task executed by the AI has its own configuration. So, it is free to use different models of different sizes and with different parameters, like temperature, top-p etc. It is implemented by an interface conventionally ending in -AiService (e.g., IngredientRoleAiService) as per the quarkus-langchain4j specifications. This is convenient but has two drawbacks: it does not support the application's reactive model, and often the AI's

output needs further processing, normalization or cleanup. To keep the application logic clean, the AiService is wrapped with a facade component (e.g., SuggestionsAiFacade). A facade component may wrap many related AI services. It adapts their synchronous model to the application's reactive model and applies any post or pre-proccesing to the data.

Overall, the AI integration architecture is designed to be modular, application-controlled, and operationally configurable. By using quarkus-langchain4j with Ollama and encapsulating AI-assisted behavior behind dedicated service abstractions, the backend incorporates language-model capabilities in a way that is consistent with the broader architecture of the platform.

The following table provides an alphabetical index of the quarkus-langchain4j components and a summary of their role in the suggestion pipeline, referencing chapter 3 where applicable:

AI service class name	Role
AlternativeSuggestionAiService	Suggest an alternative ingredient, see 3.7
FindBestruleAiService	Determine the best rule to apply, see 3.5
IngredientMatchInRecommendationsAiService	Determine the GBD rules corresponding to an ingredient in the recipe, see 3.4
IngredientRoleAiService	Determine the role or technique of an ingredient in the recipe, see 3.1
RecipeExtractionAiService	Extract a recipe from the content of the page transformed to Markdown; part of the recipe extraction
RecipeFilterAiService	Assess Markdown fragments, keep only those that are relevant to the recipe; part of the recipe extraction
TriggerIngredientMatcherAiService	Match each ingredient in the recipe to a trigger ingredient known to the system, see 3.2

Table 19 AI service names and roles

5.7. RECIPE EXTRACTION

The DietWise backend supports recipe extraction from two different input forms: directly from page content provided as Markdown-like text, and from a remote URL pointing to a recipe page. These two entry paths serve different client needs, but they converge into a common processing objective: transforming unstructured or semi-structured web content into a normalized internal recipe representation that can be used for downstream suggestion and scoring workflows. Recipe extraction from Markdown serves the RCA, which has direct access to the page content, while extraction from URL utilizes the DietWise Renderer and serves the MyRecipeWatch.

When the client already has access to the page content, the backend can process Markdown/text input directly. In this mode, the extraction workflow first reduces the input to the most relevant content blocks, using an AI-assisted filtering step to discard page fragments that are unlikely to contain recipe information. The remaining text is then passed to the recipe extraction logic, which produces a structured recipe model containing fields such as recipe name, yield, ingredients, and instructions. This path is intended to work efficiently when the page has already been collected or preprocessed by the client.

When the client provides only a URL, the backend first invokes an external renderer service. This renderer retrieves the target page, renders it, and returns both textual output and any structured recipe information found in embedded metadata, particularly JSON-LD. The backend then follows a two-stage strategy. If valid JSON-LD recipe data is present, it uses that structured data directly, since it is generally the most reliable and

Commented [ŽK34]: In some places it starts with lowercase letter.

Commented [ni35R34]: I think changed all to capital now.

Commented [SD36]: Please change into MyRecipeWatch throughout!

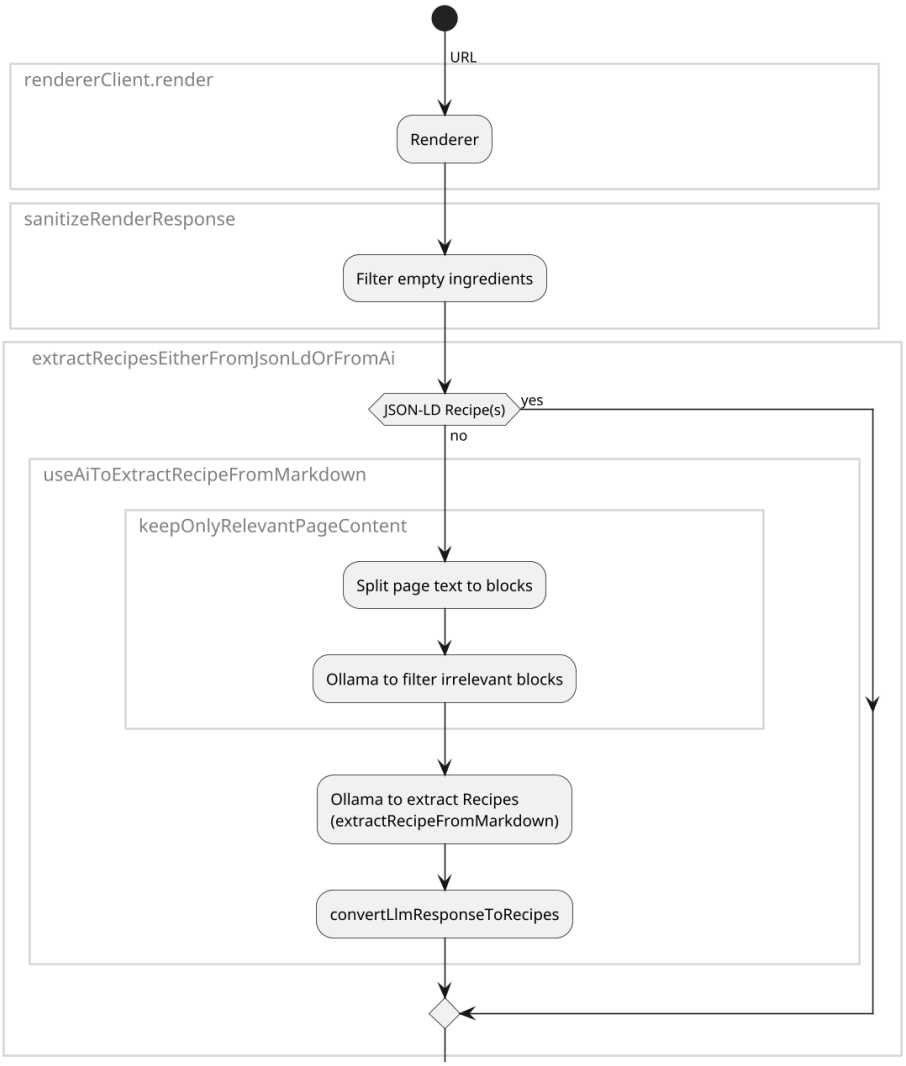
least ambiguous source. If structured data is absent or insufficient, the backend falls back to AI-assisted extraction from the rendered page text.

This dual strategy is an important design characteristic of the extraction process. The backend does not rely exclusively on AI when a structured machine-readable source is available but instead prioritizes native recipe metadata where possible and uses AI as a fallback for less structured cases. This improves robustness and reduces unnecessary dependence on heuristic extraction.

After extraction, the resulting recipe data is normalized into the internal model of the backend (see 2.4). Ingredients are converted into structured ingredient entries, and the extracted content is validated to ensure it is usable by later processing stages. Recipes without meaningful ingredients are rejected, since they are not suitable for suggestion generation or scoring. This validation step helps ensure that downstream workflows operate only on minimally complete recipe data.

The extraction workflow also preserves information about how the recipe was detected. The system distinguishes, for example, between recipes obtained from structured JSON-LD and recipes inferred from text using AI-assisted extraction. This metadata is useful for diagnostics, transparency, and client-side interpretation of the result.

A diagram of the recipe extraction is the following. The green step, "Assess Recipe", invokes the process described in 3 THE LLM SUGGESTIONS PIPELINE.


URL
rendererClient.render
Renderer
sanitizeRenderResponse
Filter empty ingredients
extractRecipesEitherFromJsonLdOrFromAi
JSON-LD Recipe(s)
yes
no
useAiToExtractRecipeFromMarkdown
keepOnlyRelevantPageContent
Split page text to blocks
Ollama to filter irrelevant blocks
Ollama to extract Recipes
(extractRecipeFromMarkdown)
convertLlmResponseToRecipes

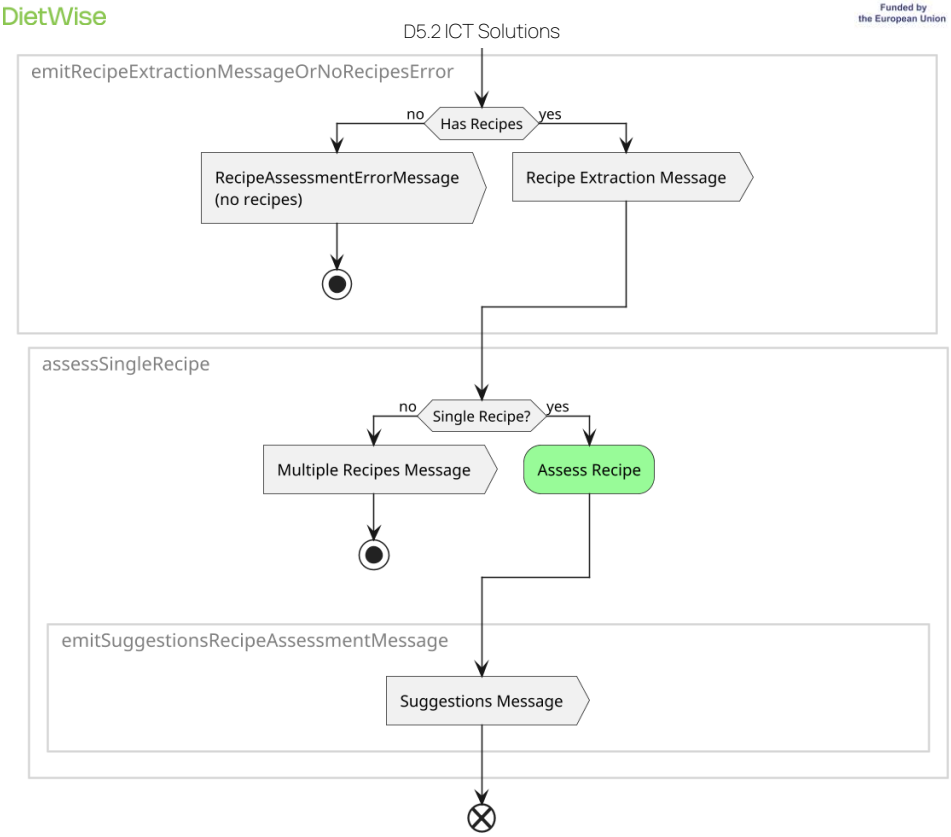


Figure 7 Recipe extraction flow diagram

5.8. SUGGESTION PRIORITIZATION

The suggestion-generation workflow is responsible for ordering substitution proposals that help users improve a recipe in line with the nutritional and recommendation logic of the DietWise platform. This process operates on the structured recipe produced by the extraction stage and the transformed individual recipe ingredients from the assessment stage.

The process combines reference data from the database with already extracted recipe data from the previous steps (see 3 THE LLM SUGGESTIONS PIPELINE). The goal of prioritization is to select and order suggestions in a way that is more relevant and useful to the user. The model of a suggestion can be found in chapter 2.4, under the Table 8 Suggestion – extends SuggestionTemplate. The work of the `SuggestionPrioritizer` is simple: load the GBD weights from the DB according to user personalization information and then sort the `Suggestion` objects according to the weight of their recommendation field.

5.9. RECIPE SCORING

The recipe's scoring/rating is performed on the client side, taking into account the substitutions the user has decided to accept. The process is described in detail in section 6.2 under "Recipe Scoring." The server is responsible only for assembling the data required for scoring and transmitting it to the client. The dedicated

`ScoringRecipeAssessmentMessage` contains the `ScoringData` structure described in section 6.2. An example can be seen in Annex C API usage details and examples, under "THE BACKEND APPLICATION".

5.10. REACTIVE PROGRAMMING UTILITIES

Reactive programming brings many benefits to the application. It enables DietWise to implement I/O-intensive and orchestration-heavy workflows efficiently and explicitly. By using Mutiny-based reactive types and aligning API, service, and persistence behavior around the same model, the backend achieves a coherent approach to asynchronous execution and progressive processing.

It also comes with some costs. One is complexity, which manifests in two ways. One is learning a new API, its intricacies, and making the necessary mental paradigm shift away from the synchronous code we are used to writing. This is something the developer gets used to in time. The other way complexity manifests is a bulky API. An example from the application is a part of the recipe assessment workflow: the system (1) uses the AI to extract a recipe from Markdown, (2) emits a message containing the extracted recipe, (3) assesses the recipe and emits suggestions and (4) calculates the score. The (simplified) Mutiny code to accomplish this looks like the following:

```
recipeExtractionService.useAiToExtractRecipeFromMarkdown()
        .flatMap(emitRecipeExtractionMessageOrNoRecipesError())
        .flatMap(emitRecipeExtractionMessageOrNoRecipesError())
        .flatMap(calculateScoreData());
```

The `flatMap` calls add noise to the code and make the developer lose focus. It is worse when a step in the process depends not only on the result of the previous step, but on results before that. A Mutiny chain of three fictional operations, op1, op2 and op2 each requiring the result of all the previous ones looks like:

```
op1().flatMap(result1 -> op2(result1).flatMap(result2 -> op3(result1, result2)))
```

This example is small but still shows the nesting which ends up being confusing and hard to read. A small helper was developed that uses overloaded static methods to get rid of the `flatMap` noise. The class is called `UniComprehensions` and the method(s) `forc`, inspired by Scala's "for comprehension" syntax. With it the chains above can be written as:

```
forc (
    recipeExtractionService.useAiToExtractRecipeFromMarkdown(),
    emitRecipeExtractionMessageOrNoRecipesError(),
    emitRecipeExtractionMessageOrNoRecipesError(),
    calculateScoreData()
);
// and
forc (
    op1(),
    result1 -> op2(result1),
    (result1, result2) -> op3(result1, result2)
);
// or, using method references:
forc (
    op1(),
    this::op2,
    this::op3
);
```

5.11. DB SCHEMA OVERVIEW

The database schema of DietWise consists of two main categories of data. The first is relatively stable reference data that supports the recommendation and substitution logic, such as recommendations, age groups, trigger ingredients, roles or techniques, alternative ingredients, rules, and suggestion templates. The second is user-specific operational data, including the internal user record, personal information, application usage statistics, and per-suggestion interaction statistics.

Structurally, the schema is centered around the rule-based suggestion model. A rule links a recommendation to a trigger ingredient and a cooking role or technique, while one or more suggestion templates define the concrete alternatives associated with that rule. Recommendation values are parameterized by age group and biological gender, allowing scoring and personalization logic to use structured nutritional guidance data. On the user side, the schema keeps the persisted user identity intentionally minimal and attaches optional profile and statistics records through foreign-key relationships.

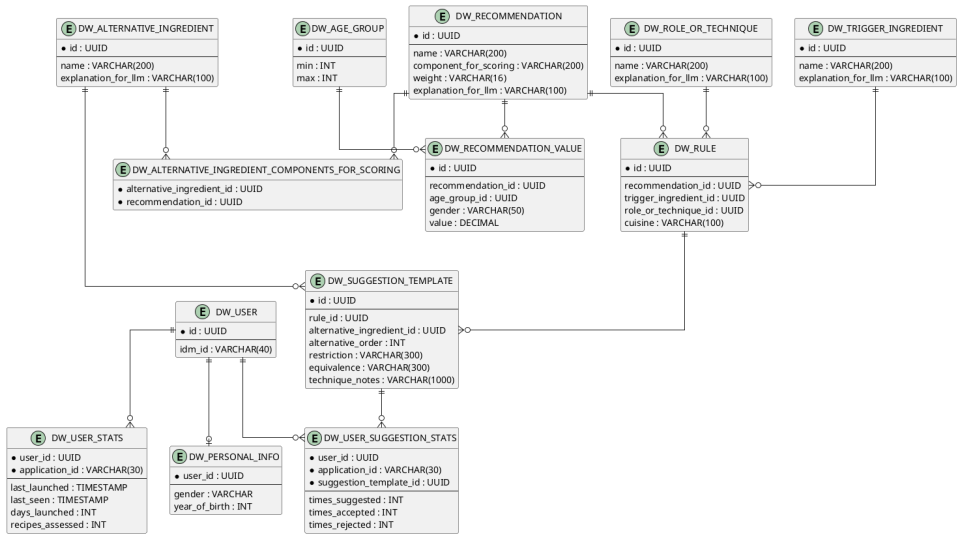


Figure 8 Database schema

5.12. C4 COMPONENT DIAGRAM

Here we present a C4 simplified component diagram of the backend. It centers on the `dietwise-services` module, i.e. the application-service layer. It includes all the components that are important for the business of case of recipe assessment, leaving out e.g. the `PersonalInfoService`.

The entry point is the API, as discussed earlier, represented by the `dietwise-jaxrs` module at the bottom left of the diagram.

The primary component is the `RecipeAssessmentService`, which is directly called by the API and serves as the central orchestrator of the DietWise workflows. It defines and drives the recipe extraction flow. The second component is the `RecipeSuggestionsService`, which is responsible for defining and driving the suggestions subflow.

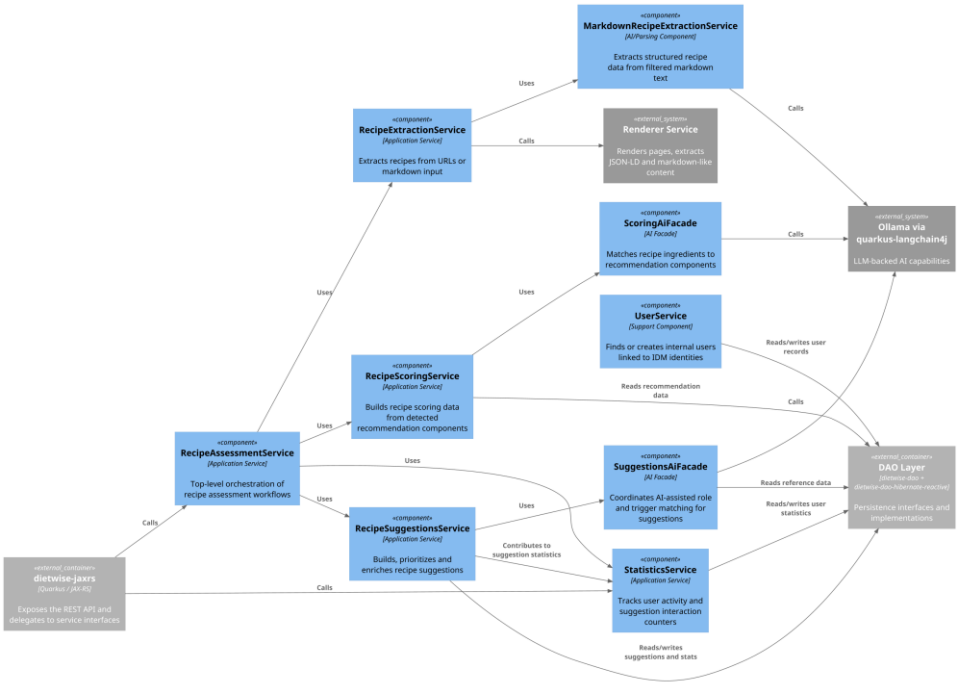


Figure 9 Backend C4 component diagram

5.13. CONFIGURATION AND RUNTIME DEPENDENCIES

The DietWise backend uses externalized configuration so that environment-specific settings are supplied at runtime rather than embedded in the application code. This allows the same application build to be used across different environments while keeping infrastructure parameters, credentials, and service endpoints under deployment control.

In practice, the backend relies on Quarkus configuration for its main runtime settings, including database connectivity, authentication integration, and connections to external supporting services. Environment-specific adjustments can be applied through standard configuration overrides, allowing local development and deployed environments to use different values without changes to the codebase.

A more detailed description of each configuration option is given in Annex D Backend configuration.

5.14. AUTHENTICATION AND AUTHORIZATION

The DietWise backend applies security primarily at the application boundary, where incoming requests are authenticated and associated with an application-level user context before business processing begins. Authentication is integrated through OIDC, allowing the backend to rely on an external identity provider rather than managing credentials directly within the application. This reduces responsibility in the backend for identity lifecycle management and aligns security with a centralized authentication model.

The User object isolates the application logic from the specifics of the underlying framework. This improves separation of concerns and helps keep business logic independent from low-level token handling details.

DietWise does not have complex authorization needs, so the implementation is simple. However, as good practice, the authorization code is separated from the main business logic. The interface `Authorization` and its implementation `AuthorizationImpl`, expose two methods to authorize actions, `requireLogin` and `requireApplicationId`. The authorization service implements any kind of authorization logic, simple or complex, so that the domain services that consume it remain simple, which means easier to understand and reason about. Testing becomes more straightforward too, because one can mock the `Authorization` instead of manufacturing the data and context required by the `AuthorizationImpl` for every business test case. Conversely, the `AuthorizationImpl` is easier to test because we only need to create the authorization context, not the business data.

The application maintains minimal, non-personally identifiable user data in its own database to link authenticated identities to application data. This is personal information (gender, year of birth) and usage statistics. This approach supports traceability and domain-level data ownership while avoiding duplication of full identity management responsibilities.

Sensitive runtime settings, such as database credentials and service connection parameters, are externalized through configuration rather than embedded in source code. This reduces the risk of accidental exposure through the codebase and supports environment-specific security controls at the deployment level.

A specific authentication support mechanism is provided for the Responsible Cooking Alliance (RCA) client, implemented as a browser add-on. Because browser extensions cannot use the same callback handling approach as a conventional web application, the backend serves a dedicated callback page, extension-callback.html. This page receives the authorization response from Keycloak, extracts the code and state parameters, forwards them to RCA through a browser message event, removes the sensitive query string from the visible URL, and closes itself. In this way, the backend supports the OAuth login flow required by the RCA extension without exposing credentials or tokens directly in extension code.

This callback page is intentionally hardened through restrictive HTTP and browser security headers. It is served with `no-cache` directives, a restrictive Content Security Policy, `no-referrer` behavior, and frame protection. These measures reduce the risk of callback data being retained, leaked, or embedded in an unsafe context.

At a broader level, the backend's security model is based on four principles: delegated authentication, controlled propagation of user identity, minimal persistence of identity-related data, and separation of security infrastructure from business logic. Together, these measures provide a sound foundation for securing access to the application while maintaining the implementation and keeping it consistent with the overall architecture.

5.15. TESTING STRATEGY & IMPLEMENTATION

The DietWise backend follows a modular testing approach aligned with its layered architecture. Tests are placed close to the modules they validate, which allows domain models, service logic, and persistence behavior to be verified independently. This supports faster feedback during development and helps keep defects isolated to the relevant architectural layer. Every component is designed to be testable with fast and focused unit tests. The test framework of choice is JUnit 6 (the latest).

This is not to say that integration tests are avoided. Specifically, DAO implementation tests use not only a real database through Testcontainers[24], but also apply the real Liquibase migrations to them. Launching

[24] https://testcontainers.com/

PostgreSQL as a test container and applying the migrations for DietWise's small schema is surprisingly fast. If the schema was bigger, we would have split the backend into microservices to reap the same benefits.

A couple of JUnit extensions are implemented in the `dietwise-common-testutils` module. The `HibernateReactiveExtension` helps set up a Hibernate Reactive test environment needed for testing the DAOs. Its counterpart, the `MockReactivePersistenceContextFactory`, is used for testing business logic where the database is a stub[25]. The latter records the operations performed on the stub reactive transaction, which effectively becomes a test spy.

The second extension, the `LiquibaseExtension`, bootstraps and executes Liquibase in the test environment.

A simple, concrete example is the `UserDaoImplTest` (only relevant parts shown):

```
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)
@Testcontainers                                                          ❶
public class UserDaoImplTest {
  @Container                                                             ❷
  private static final PostgreSQLContainer postgres =
    new PostgreSQLContainer(POSTGRES_IMAGE);

  @RegisterExtension
  @SuppressWarnings("unused")
  private static final LiquibaseExtension liquibaseExtension =           ❸
    new LiquibaseExtension(postgres::getJdbcUrl,
      postgres.getUsername(), postgres.getPassword());

  @RegisterExtension
  @SuppressWarnings("unused")
  private static final HibernateReactiveExtension hibernateReactiveExtension =   ❹
    new HibernateReactiveExtension(postgres::getJdbcUrl,
      postgres.getUsername(), postgres.getPassword());

  @Test
  @Order(1)
  void test(Mutiny.SessionFactory sessionFactory) {                      ❺
    // ordinary test code here, able to use the reactive SessionFactory
  }
}
```

❶ Activate the TestContainers

❷ Define a PostgreSQL test container, it launches the DB in Docker

[25] According to the definition of various Test Doubles by Fowler: https://martinfowler.com/bliki/TestDouble.html

❸ Our own `LiquibaseExtension` will apply the migrations to the DB test container. See how we obtain connection information from the injected test container (2).

❹ Our own `HibernateReactiveExtension` that activates Hibernate Reactive in the test methods

❺ Thanks to the `HibernateReactiveExtension`, the reactive session is injected to the test methods

5.16. BUILD AND RELEASE PROCESS

The DietWise backend is built as a Maven multi-module project, with the root project acting as the main build entry point for the full backend codebase. The primary build output is the Quarkus-based backend application. In accordance with Maven's philosophy, running `mvn clean package` in the project's root is enough to build the executable and run the tests, requiring only Java, Maven and Docker. More targeted developer workflows exist, as well as a Maven profile for building the *development* version of the application's peripherals. They are thoroughly documented in the project's README.md file.

Release preparation follows a conventional, versioned Maven process. The project version is updated explicitly, i.e., running a `mvn versions:set -DgenerateBackupPoms=false -DnewVersion=x.y.z` command; corresponding git commits and tags are created, and development then continues on the next snapshot version. The git tag is the Maven version prefixed with "v" as is customary in Git, so for Maven version 1.2.3 the tag is "v1.2.3".

Version numbers follow the semantic versioning specification[26]. The current version follows the Maven convention of using the -SNAPSHOT pre-release identifier.

The project naturally supports container-based packaging. The root Dockerfile executes the entire Maven build (no tests) and then creates a deployable application image. To keep the build fast, it caches the Maven dependencies first in a cache mount[27], then copies the sources and builds. As part of this process, the Maven module hierarchy and Docker packaging remain aligned through generated Dockerfile sections - see the `dietwise-architecture/src/scripts/update-dockerfile-pom-copy.sh` script. In conclusion, the Docker build allows one to obtain the application's executable image with Docker as the only dependency. This is utilized in the actual deployment of DietWise.

6. MYRECIPEWATCH APP

Commented [SD37]: One word?

MyRecipeWatch is a mobile application for assessing recipes found on third-party web pages and presenting healthier or more sustainable ingredient substitutions. It is an Ionic application using React and Typescript, with Capacitor packaging for native mobile deployment. It uses Jotai for state management and `react-i18next` for managing translations.

The application is built with Vite and Vitest for testing.

This section describes the implementation down to the component level. It focuses on runtime structure, state ownership, data flows, and the responsibilities of modules and UI components.

[26] https://semver.org/
[27] https://docs.docker.com/build/cache/optimize/

6.1. FUNCTIONAL DESCRIPTION

The core functionality of MyRecipeWatch is to display personalized alternatives of recipe ingredients to citizens that use the app to improve the healthiness and sustainability of the entire recipe. Users enter a URL address, and the application exchanges data with the backend to complete the assessment.

The mobile application cannot access browser content like a browser add-on (the RCA) can. This forced the introduction of the DietWise Renderer, the service described earlier that renders web page content. It also limits MyRecipeWatch's recipe extraction capabilities. It cannot extract a recipe from a page requiring a login, for example. This should not affect many real use cases, since most recipe content is publicly available.

To support the core functionality, MyRecipeWatch includes a personalization page where the user can enter their biological gender and year of birth. The system takes this optional information into account when making suggestions.

The DietWise services are provided to freely registered users, so MyRecipeWatch integrates with Keycloak to provide authentication. User registration is available from Keycloak's login screen as an alternative option. Once the user is authenticated, MyRecipeWatch stores a long-lived offline token, as per the OAuth2 specification. The token is stored in secure device storage using the `capacitor-secure-storage-plugin`.

Besides these, MyRecipeWatch has a welcome page and a preferences page that allows the user to select the UI language.

6.2. ARCHITECTURE & IMPLEMENTATION

Code design/directory layout

The project follows a feature-oriented frontend design with a thin application shell and per feature/functionality subdirectories under the directory `src/`.

Subdirectory	Purpose
auth	authentication integration and user identity exposure
config	runtime and build-time backend configuration
settings	persisted client settings (language selection)
personalization	user profile data capture and persistence
recipe	core recipe assessment workflow, state machine, and interaction UI – this is where most of the functionality is implemented
common	reusable low-level utilities
services	General purpose services, for not it houses infrastructure adapters used by authentication
components	application-shell UI shared across features

Table 20 MyRecipeWatch, code directory layout

Within each feature/functionality subdirectory, the code is broadly separated into the following files:

File	Purpose
model.ts	TypeScript interfaces and domain types
atoms.ts	Jotai state and derived state
api.ts	backend communication
*Page.tsx	implementation of the main page/route
components/	any fine-grained components used by the page

Table 21 MyRecipeWatch, files per feature/functionality directory

Internally, the application is organized around a bootstrap phase in `src/main.tsx`, a shell/router layer in `src/App.tsx`, and a master/global stylesheet in `src/App.css`. The bootstrap phase is intricate, involving the following steps to initialize the application:

1. Load persisted user settings from secure storage
2. Load application configuration from `config.json`, Vite env vars, or hardcoded fallbacks
3. Configure the authentication server host using the resolved app config
4. Initialize i18n using the persisted language
5. Initialize the auth service
6. Attempt to restore a stored token, refresh it, and load user info
7. Create a dedicated Jotai store
8. Seed that store with resolved config and settings
9. Finally, mount the React application inside a JotaiProvider and display the UI

The application components

Root component – App.tsx

`src/App.tsx` is the application shell. Its responsibilities are to initialize Ionic React, configure the routing system and host the side menu and main outlet.

Routes and menu

The shell exposes these route-level pages:

URI	Page component
/Home	HomePage
/Recipe	RecipePage
/Personalization	PersonalizationPage
/Settings	SettingsPage
/authcallback	AuthCallbackPage
/endsession	EndSessionPage
/	Redirects to /Home

Table 22 MyRecipeWatch routes

`src/components/Menu.tsx` implements the application menu, accessible from the top left "hamburger" that is visible on all pages. The menu shows the appropriate login/logout buttons, the user's email for authenticated users, and version information.

Configuration

The configuration subsystem determines backend endpoints without rebuilding the app for every environment. It contains only two pieces of information: the `authServerHost` (the base URL for the authentication server, i.e., Keycloak) and the `apiServerHost` (the base URL of the DietWise backend).

`src/config/loadAppConfig.ts` resolves configuration in this order: hardcoded local fallback defaults (used primarily for development), build-time environment variables, and the runtime `config.json` file served under the app base path.

`src/config/atoms.ts` stores the resolved config in `appConfigAtom` and exposes the derived atoms `authServerHostAtom` and `apiServerHostAtom`. These atoms are read by auth and API-facing feature logic.

Settings

The only user-editable application setting is the UI language. The user's selection is persisted exclusively on the device in secure storage, like the offline token.

Internationalization/UI translations

`src/i18n/index.ts` configures i18next with bundled translation resources, one JSON file per language under the same folder. The i18n layer is initialized during bootstrap so route pages and atoms can safely use translated messages once the UI mounts.

Authentication

The authentication is implemented as a service integration rather than a React-only hook-based subsystem.

The core service, `src/auth/authService.ts`, creates a singleton named `authService` based on `ionic-appauth`. It encapsulates OpenID Connect configuration, redirect URL construction for each environment (web for development and Capacitor for production), token refresh on app resume for native builds, in-memory tracking of token presence, and a helper to request a valid access token.

Platform-specific request transport, a request abstraction implemented in `src/services/RequestorImpl.ts` selects the appropriate requestor implementation for `ionic-appauth`. The `src/services/CapacitorRequestor.ts` adapts Capacitor HTTP for iOS/native needs, while in other cases it uses the built-in `FetchRequestor`. This isolates authentication transport concerns from the rest of the app.

The state of the authentication mechanism is exposed as Jotai atoms, to be used in preference over the raw `authService`. The atoms observe events emitted by the `authService` and `ionic-appauth` to keep their contents up to date.

A sequence diagram of the interactions, omitting the logout functionality for brevity is the following:

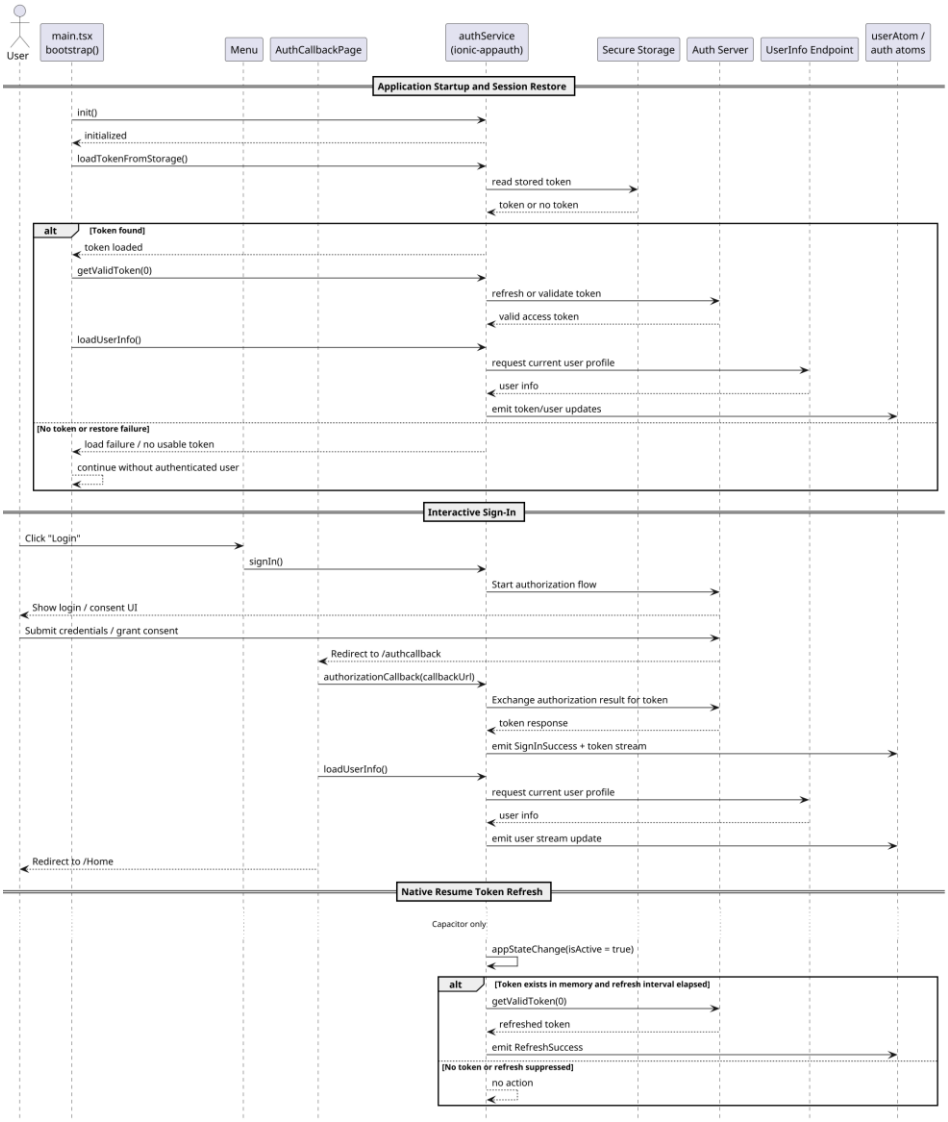


Figure 10 MyRecipeWatch authentication interaction diagram

Auth callback pages

A key element in the OAuth2 flow is the callback URLs, called both after a successful login and after a logout. In the case of Ionic, the callback pages have to use the platform's deep linking capabilities because Keycloak's

login page is displayed in the system browser. However, MyRecipeWatch is packaged as an app installed in the device, not a web page, hence the need for deep linking for the callbacks to work.

In terms of functionality, the `AuthCallbackPage` consumes the authorization callback URL, waits for sign-in success or failure, loads user info, and finally redirects to /Home. The `EndSessionPage` used for logging out, completes the end-session callback, also redirecting to /Home.

The Recipe Assessment page

The recipe feature is the core of the application and has the richest internal structure. It combines a streaming backend assessment call, a reducer-backed feature state machine, and interactive selection/rejection of suggestions.

The streaming nature of communication with the backend is noteworthy. The naive option for assessing a recipe would be to make a single, long call to the server and wait for it to finish processing before displaying the results. This has the advantage of being simple to implement. In terms of user experience, however, it is not optimal because it forces the user to wait a long time, as the processing that involves AI takes a considerable amount of time. Streaming the results means the server sends intermediate results every time as they become ready, and the user sees a progressively updating UI. So, the application receives a message when (a) the recipe ingredients have been determined, (b) suggestions are ready, (c) scoring the recipe is finished. The server may transmit an error in place of any of those events, in which case the UI displays the error, and the communication is terminated.

Streaming communication with the backend is implemented in `src/recipe/assessRecipe.ts`. It calls the backend URL assessment endpoint, forwards each parsed message to the caller (the `assessRecipeCallback` state machine in the `RecipePage`), and exposes cancellation via an `AbortController`.

Model

`src/recipe/model.ts` defines the complete feature model, including the domain types presented earlier. It also defines the client-side state of the recipe page, `MainData`. The shape of the state is the following:

MainData		
Field name	Type (Optional)	Comments
status	Enum of INITIAL, PENDING, SUCCESS, SELECT_RECIPE	Reflects the page's overall state. E.g., PENDING means that the system is still receiving data from the server. Portions of the UI are display based on this value.
errors	String[] (O)	
rating	number	The recipe rating/ranking/score
recipes	Recipe[] (O)	
detectionTypes	RecipeDetectionType (O)	How was each recipe extracted, JSON-LD or AI
suggestionKeys	String[] (O)	Keys of all the suggestions (field `suggestions` below)
emptySuggestionsFromServer	Boolean	Flag to signal that the server could not make any suggestions
suggestions	Map<String, SuggestionState>	The actual suggestions and their state (ACCEPTED, REJECTED, UNDECIDED)

Commented [ŽK38]: Is it cut off?

Commented [ni39R38]: Yes, online Word hiccup, thank you

ingredientState	Map<String, String>	Ingredient to currently accepted suggestion for replacement, if any
url	String	URL of th
pageText	String (O)	The content of the recipe page in Markdown
scoringData	ScoringData	

Table 23 Front end model structure MainData

The `SuggestionState` is unique to the front-end:

SuggestionState		
Field name	Type (Optional)	Comments
suggestion	Suggestion	
status	SuggestionStatus	ACCEPTED, REJECTED, UNDECIDED'

Table 24 SuggestionState model

Reducer-based orchestration and Jotai integration

`src/recipe/reducer.ts` and `src/recipe/actions.ts` implement the Redux-like state machine using a Jotai reducer atom. The reducer is responsible for keeping the recipe feature state internally consistent. Two helper modules refine reducer behavior: The `src/recipe/reducers/reduceSuggestionStatusAction.ts` applies accepted/rejected/undecided transitions, ensures only one accepted suggestion exists per ingredient, updates ingredient-to-suggestion assignments, recalculates rating/score/ranking of the recipe and `src/recipe/reducers/calculateRating.ts` computes the normalized recipe score from backend scoring data plus current accepted substitutions.

src/recipe/atoms.ts exposes the reducer atom, `mainStateAtom`, and the suggestionInFlightAtom that stores per-suggestion UI lock state while statistics updates are in progress.

Streaming low-level transport

`src/common/streamJson.ts` is the low-level streaming helper used by recipe assessment. It performs the POST request, incrementally reads the response stream splitting it by newline, parses each line as JSON and forwards parsed messages to callbacks. It treats abort as completion rather than failure. This is the key infrastructure piece enabling progressive updates in the recipe UI.

Suggestion statistics integration

User actions on suggestions are also reported to the backend. The `src/recipe/api.ts` exposes `postSuggestionStatistics()` which transmits user acceptance/rejection of a suggestion to the backend. This call is not essential for the functioning of the application, so failures are silently ignored.

Recipe page structure

The `RecipePage` is the most complex part of the application, so its functionality is broken down to manageable components.

Component	Function
UrlContainer	Shows the current recipe URL Displays a spinner when the assessment is in progress Acts as the click target for opening the URL modal

UrlModal	Lets the user input the recipe page URL and language
SplitPane	This generic component is responsible for the draggable split between the top and bottom halves of the recipe page
RecipesComponent	Renders the top, recipes half of the page
RecipeComponent	Renders a single recipe with its name, the rating displayed as stars (see the RatingComponent) and the scrollable list of ingredients
IngredientComponent	Renders one ingredient row with its substitution, if any
SuggestionsComponent	It renders the bottom half of the page with a list of suggestions
SuggestionComponent	Renders a single suggestion and its action controls
useSuggestionInFlight	It is a coordination custom React hook that tracks temporary per-suggestion locks in Jotai and mirrors them in a ref-backed `Set` to block repeated clicks before rerender. This prevents duplicate user actions during optimistic updates.
Recipe help components	Display the help content/modal

Table 25 MyRecipeWatch, components of the recipe page

The recipe assessment flow can be summarized in the following diagram:

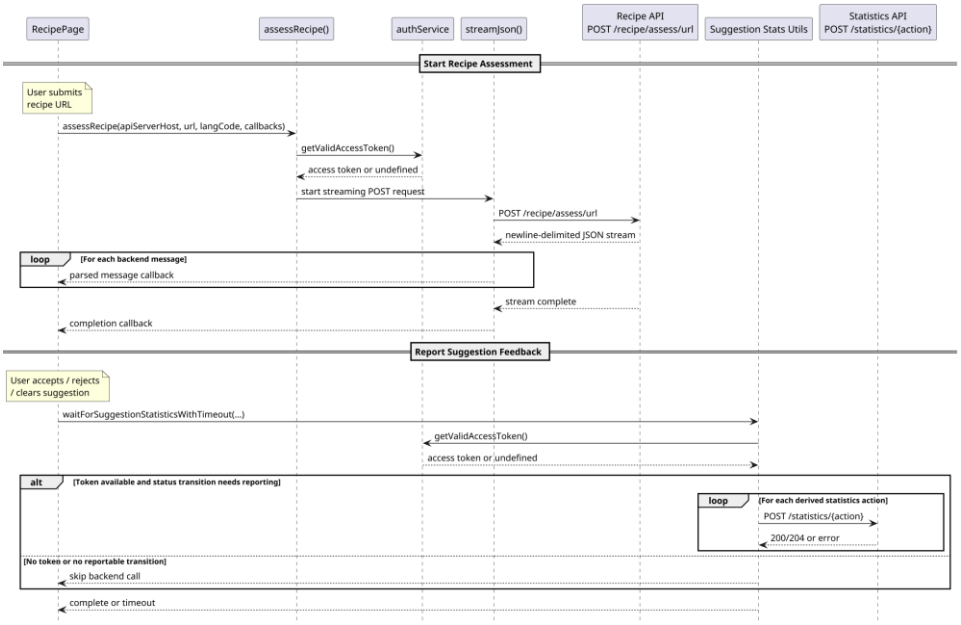


Figure 11 Recipe assessment flow as seen by the front-end

The suggestion interaction flow is an optimistic UI model with best-effort telemetry/statistics persistence, summarized in the following diagrams:

(a) when the interaction is triggered from the suggestions pane:

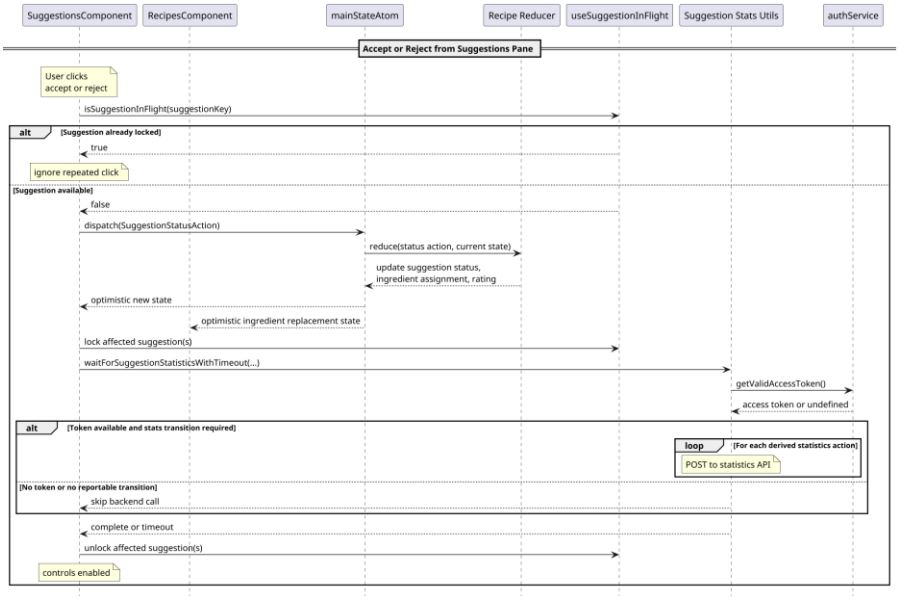


Figure 12 Accept or reject suggestion from the Suggestions pane

(b) when the interaction is triggered from the recipe pane:

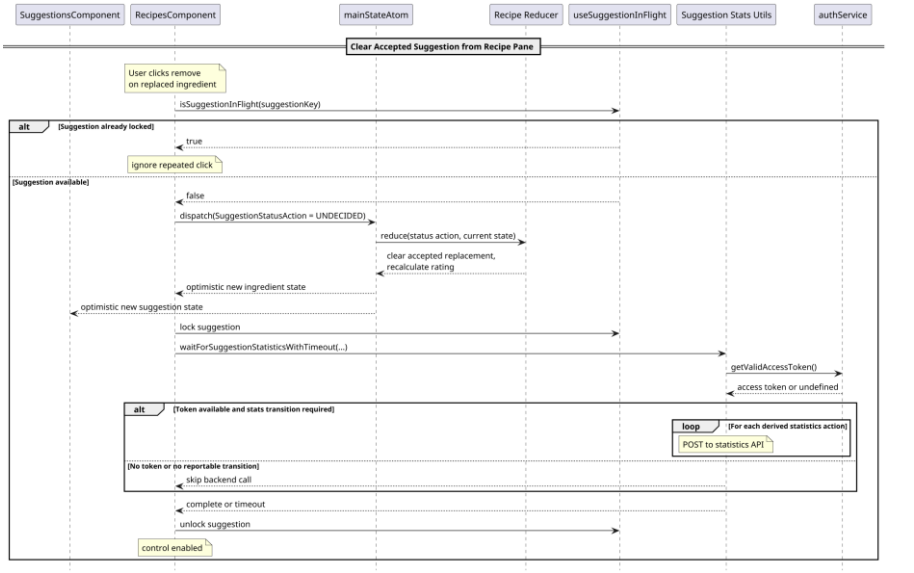


Figure 13 Accept or reject suggestion from the Recipe pane

Recipe Scoring

A central function of the MyRecipeWatch application is the provision of an immediate and comprehensible score for the assessed recipe. This score is intended to reflect, in a compact way, the degree to which the recipe aligns with the nutritional and sustainability-oriented criteria computed by the DietWise backend. The corresponding logic is implemented in the client-side module `src/recipe/reducers/calculateRating.ts` and is executed both for the initially extracted recipe and whenever the user accepts, rejects, or removes a suggested ingredient substitution.

The scoring approach is deliberately split between server and client responsibilities. The backend is responsible for the knowledge-intensive part of the process, namely the derivation of the scoring model for the recipe under assessment. To this end, it sends the following `ScoringData` structure to the client. In this way, the server defines the semantic basis of evaluation, while the client performs lightweight recalculation during interactions.

ScoringData	
Field name	Explanation
`totalNumberOfRecomendations`	The total number of GBD recommendations known to the system
`recommendationWeights`	A map from the GBD recommendation text to an enumeration with values `ENCOURAGED` or `LIMITED`
`recommendationsPerIngredient`	A map from the id of the ingredient in the assessed recipe to the GBD recommendation texts related to it

Table 26 ScoringData structure

The client-side algorithm begins by constructing a presence map covering all component categories referenced in `recommendationWeights`. Each category is initially marked as absent. The algorithm then iterates through the recipe ingredients and determines which component categories should count for each

ingredient in the current state of the recipe. For the pristine recipe, this contribution is taken directly from `recommendationsPerIngredient`, that is, from the server-provided characterization of each original ingredient. If, however, the user has accepted a substitution for a specific ingredient, the logic replaces the original component categories with the `alternativeComponentNames` associated with the accepted suggestion. In effect, the recipe is rescored not against a static original formulation, but against the currently selected formulation assembled through user interaction.

Once the applicable component categories have been identified for all ingredients, the algorithm marks them as present in the presence map. It then computes the raw score by evaluating each category against its expected orientation. A category marked as `ENCOURAGED` contributes positively when it is present in the current recipe state. Conversely, a category marked as `LIMITED` contributes positively when it is absent. This design reflects the basic DietWise principle that recipe improvement may occur either by introducing desirable components or by removing or reducing undesirable ones. Each satisfied condition contributes one point to the total score.

The final recipe score is obtained by normalizing the raw score with respect to `totalNumberOfRecomendations`. The output is therefore a ratio in the range from 0 to 1, representing the share of relevant criteria currently satisfied by the recipe. Elsewhere in the user interface, this value is rendered in a more user-friendly visual form, but the underlying computation remains a normalized proportion derived from the backend-defined criteria set. It was chosen to display the 0...1 rating as 10 stars in the UI.

An important implementation aspect is that recalculation takes place on the fly in the client. Whenever a suggestion is accepted, rejected, or reset to an undecided state, the reducer logic updates the current recipe state and immediately invokes `calculateRating()`. This behavior is visible in `src/recipe/reducers/reduceSuggestionStatusAction.ts`, where every state transition affecting an ingredient substitution is followed by a fresh score computation. As a result, the score shown to the user always corresponds to the recipe as currently configured, without requiring an additional round-trip to the backend.

This architectural choice offers several advantages for the DietWise framework. First, it improves responsiveness and usability, since users receive immediate feedback on the effect of each substitution. Second, it reduces unnecessary backend load, as repeated recalculation during exploration of alternatives is performed locally. Third, it preserves methodological consistency, because the client does not invent scoring criteria on its own; it merely applies, in a deterministic manner, the scoring model already computed and transmitted by the server. In this sense, the approach combines centralized scientific and rule-based reasoning with decentralized, interaction-oriented recalculation.

The process is summarized in the following diagram:

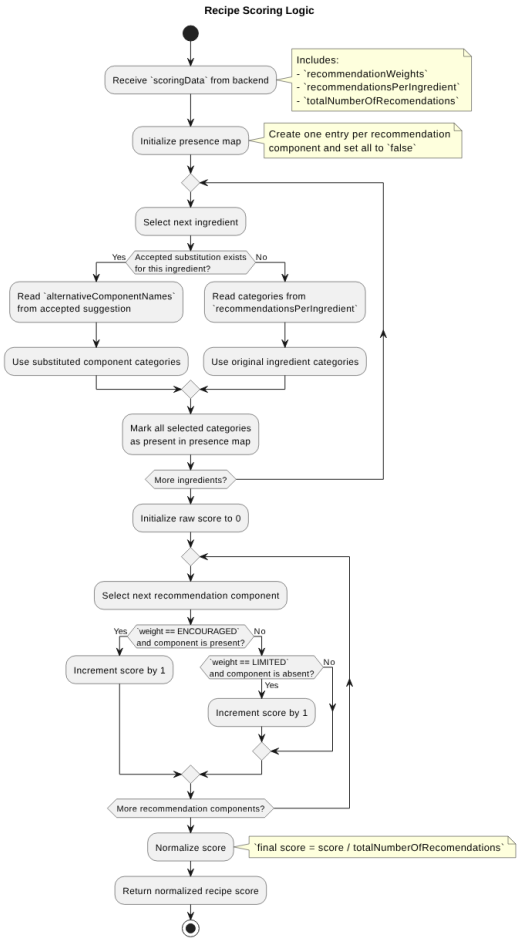


Figure 14 Recipe scoring for the front-end flow diagram

6.3. DEPLOYMENT AND ACCESS

The application is deployed as a mobile app in the respective app stores. A feature has been developed that allows it to be deployed as a standalone web app within a mobile phone-like frame for testing and demonstration purposes.

Setting the environment variable VITE_INCLUDE_MOBILE_PREVIEW to true instructs Vite to build mobile-preview.html in addition to the standard index.html. Deploying the files from dist/ to a web server and accessing the former HTML file displays the mobile preview.

6.4. USER INTERFACE

This is a tour of the application, exploring the available user scenarios. The UI is kept similar to that one of the Responsible Cooking Alliance, to the extent possible.

The user journey starts outside the application: users come to a web page containing a recipe and copy its address. They do not need to actually display the page in a browser. Keeping the page address, e.g., in the user's notes, is enough. If the user wants to assess a different recipe, they need to copy its address again. We will start with the same recipe as for the Responsible Cooking Alliance.

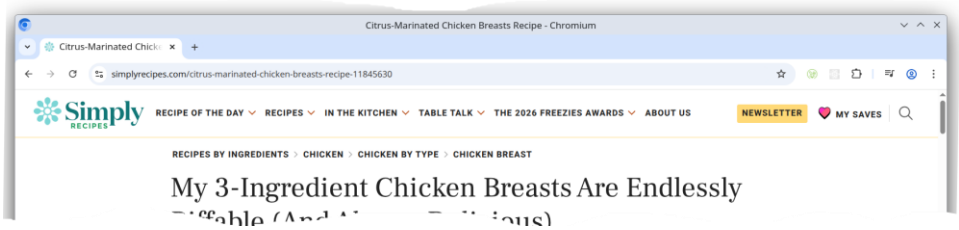


Figure 15 A Recipe that will be assessed (browser view)

Launching MyRecipeWatch, users get the Home page and basic, unauthenticated user menu:

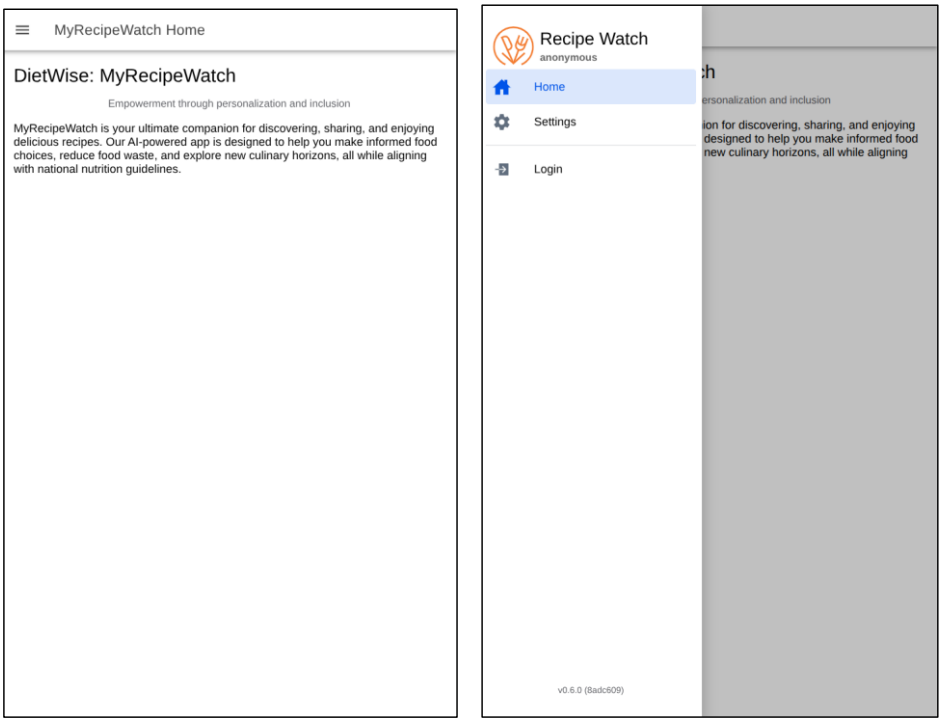


Figure 16 MyRecipeWatch home page and menu for anonymous user

Selecting "Login" presents the Login page in a secure system browser window. Moving to another window is necessary for OAuth2. The application remembers authenticated users as long as they log in at least once every 30 days. After logging-in, the menu displays extra options:

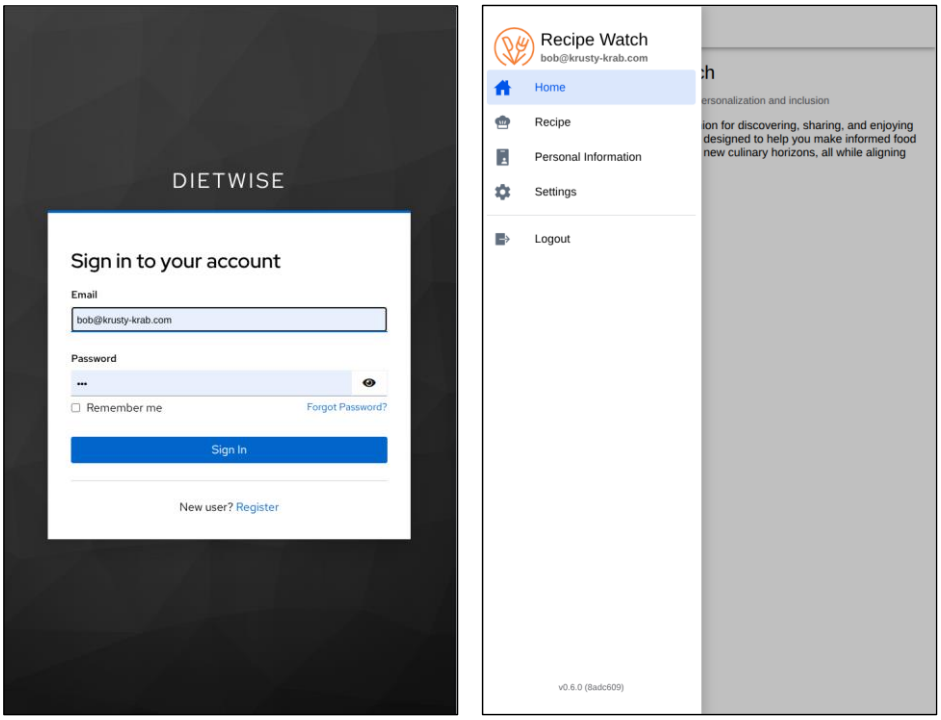


Figure 17 MyRecipeWatch sign-in screen and menu for authenticated user

The Settings page, accessible to both authenticated and unauthenticated users, allows only the selection of the UI language. This selection is local to the application, i.e., it is not persisted to the DietWise servers.

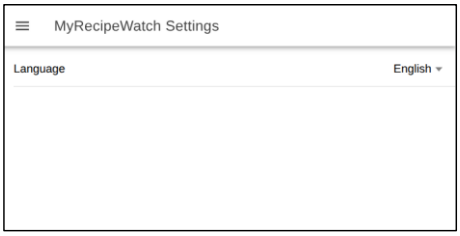


Figure 18 MyRecipeWatch settings

The personal information page allows users to select their biological sex, year of birth, and country for which they want to receive cost and seasonality information. This information is stored on the server, so the users must explicitly press the blue checkmark button at the bottom of the page to persist it.

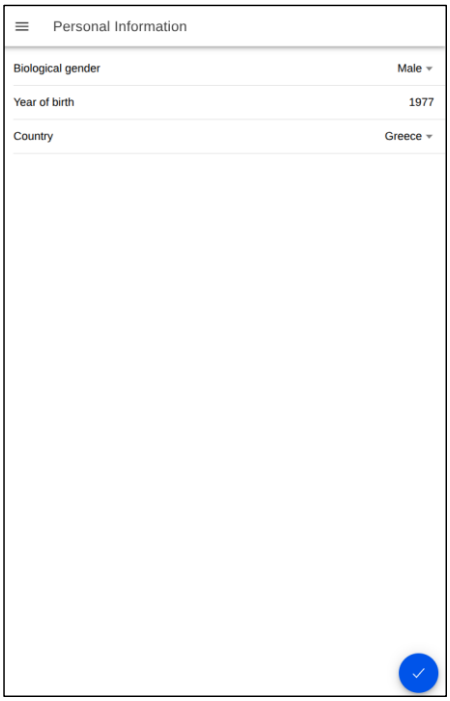


Figure 19 MyRecipeWatch personal information

The Recipe option of the menu leads to the main page of the application. The user is given instructions; the instructions are available as a pop-up later, too. The user needs to click on the top of the screen, as prompted and instructed to paste the address of the page containing the recipe and select the recipe language.

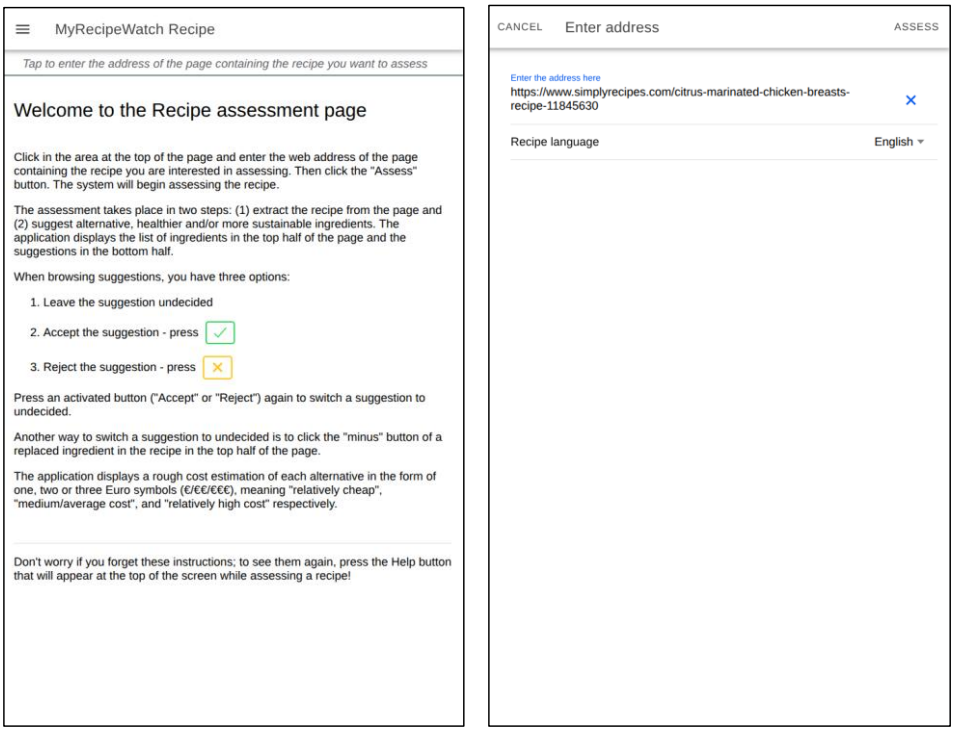


Figure 20 Entering a web address for MyRecipeWatch to assess

Clicking "ASSESS" interacts with the server in two phases. First, the recipe is analyzed to extract its ingredients, then the application displays the suggestions:

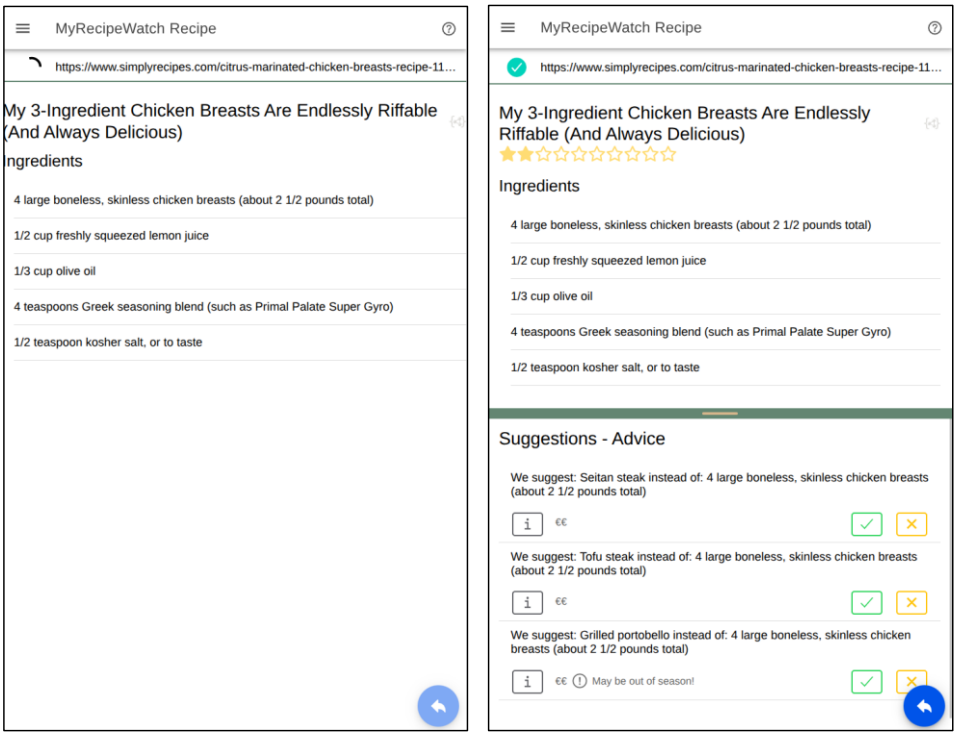


Figure 21 MyRecipeWatch assesses a recipe from a web address

The user resets the assessment by pressing the blue back arrow at the bottom-right of the page. This button is inactive while the assessment is still in progress.

Each displayed alternative contains a coarse-grained cost estimation (€=cheap, €€=medium/average, €€€=expensive) and a seasonality warning for ingredients that may be out of season. The cost and seasonality are per country, if the user has chosen to provide it on the Personal Information screen.

Commented [ŽK40]: Cool functionality - great job!

Commented [ni41R40]: Thank you!

Each ingredient allows the user to accept it (the green checkmark button), reject it (the yellow X button), or remain neutral/undecided. There may be many suggestions for the same recipe ingredient; accepting one automatically un-accepts the others. The user can reject any number of suggestions. The [i] button displays a popup with a little more detail about the suggestion, including the GBD recommendation that triggers the substitution.

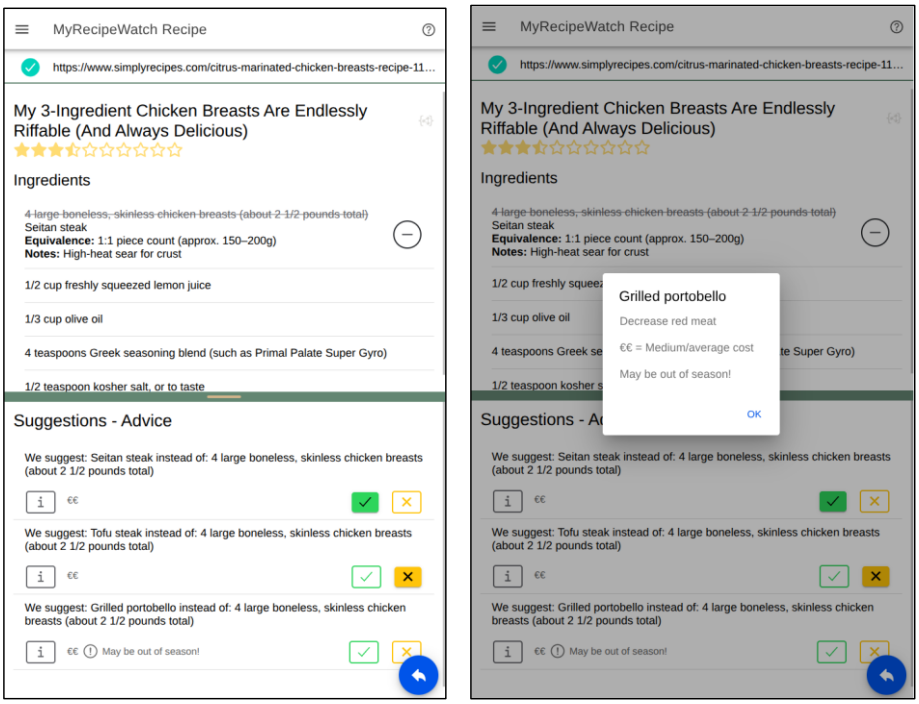


Figure 22 MyRecipeWatch suggestions

7. RESPONSIBLE COOKING ALLIANCE TOOL

This application is part of the Responsible Cooking Alliance initiative, which will guide influencers by assessing how well the recipes they publish online meet dietary and sustainability best practices. It is delivered as a browser plugin/add-on. The functionality of RCA is almost the same as that of MyRecipeWatch, and the technical implementations share many components. RCA is an Ionic application that uses React and Typescript, Jotai for state management, and react-i18next for internationalization and localization. It is built with Vite and Vitest for testing.

The initial intention of RCA was to be used by an influencer only for their own sites. However, there is no use case that requires any special integration with the site of the influencer. Therefore, this constraint is lifted in the final version. Anybody with an RCA account can assess the recipe of any site visible in their browsers. The fact that RCA is a browser add-on means it can access the content of any page the user is viewing. This eliminates the need to using the DietWise renderer. It also eliminates the limitation of extracting recipes from password-protected pages. So, influencers can assess their recipes while they are still in draft state before publishing.

Commented [ŽK42]: form?

Commented [ni43R42]: I think state is better. "Form" suggests a visible change. A draft recipe ready to be published could have exactly the same "form" as the published one, but their *state* is different DRAFT vs PUBLISHED.

Finally, RCA differs from RW in that it does not take personal information into account. It is a tool used by recipe authors (influencers) to address a broad audience rather than individual needs, so personalization would not be appropriate.

This section describes the implementation only where it differs from that of MyRecipeWatch.

7.1. ARCHITECTURE & IMPLEMENTATION

Authentication

The Responsible Cooking Alliance (RCA) browser add-on implements user authentication through an OAuth 2.0 Authorization Code flow with PKCE, using Keycloak as the identity provider. In the current implementation, the add-on is registered as client `rca` in the DietWise realm and requests the standard OpenID Connect scopes: `openid` `profile` `email`. This approach is appropriate for a browser-based client, because it avoids embedding user credentials in the add-on itself and delegates identity verification to the central authentication service. As a result, the add-on does not manage passwords locally; instead, it relies on the external identity provider to authenticate the user and to issue the tokens required for subsequent access to protected functionality.

The authentication sequence begins when the user opens the RCA side panel or sidebar and selects the login action. At that point, the add-on generates a PKCE proof pair composed of a `code_verifier` and the corresponding SHA-256 based `code_challenge`. The verifier is stored temporarily in the browser extension's local storage, while the challenge is appended to the authorization request sent to the identity provider. The add-on then opens a separate browser tab pointing to the Keycloak authorization endpoint. The request includes the client identifier, the requested scopes, the response type code, the redirect URI, and the PKCE challenge parameters. This means that even if the authorization code was intercepted, it could not be redeemed without the verifier that remains under the control of the add-on.

After the user successfully authenticates with the identity provider, the authorization server redirects the browser to a callback endpoint. At this point, the implementation differs slightly depending on the browser family. In Firefox, the redirect URI points directly to an extension-packaged callback page. That page extracts the authorization code from the query string and sends it as an internal runtime message to the add-on background script. In the Chromium-based variant, the redirect URI points to a backend-served page (`/extension-callback.html`) hosted on the DietWise backend server. Based on the implemented content-script logic, this backend page is expected to signal successful authentication to the add-on by posting a message containing the authorization code, which is then captured by the extension and forwarded to its background service worker. In both variants, the underlying design is the same: the browser is redirected to a page that acts as the handover point between the external authentication server and the RCA add-on, thereby signaling that authentication has been completed successfully.

Once the add-on receives the authorization code, it performs the token exchange against the Keycloak token endpoint. During this step, it submits the authorization code together with the client identifier, the same redirect URI, and the stored PKCE verifier. If the verification succeeds, the identity provider returns an access token, a refresh token, and a token lifetime. These values are stored in the browser extension's local storage. The RCA user interface treats the presence of valid tokens as an indicator that the user is authenticated: when tokens are present, the main application page is shown; otherwise, the login page is displayed. The implementation also registers listeners for storage changes, ensuring the user interface is updated immediately when authentication completes.

Session continuity is handled through refresh tokens. Just as MyRecipeWatch, RCA obtains long-lived, offline refresh tokens. Before protected operations are executed, the add-on checks whether the access token is still valid. If expiry is approaching, the add-on automatically calls the token endpoint with the refresh token to obtain a new access token, and, in some cases, an updated refresh token as well. This reduces the need for repeated user logins and supports a smoother user experience while preserving centralized session control at the identity-provider side. In the current implementation, logout is local to the add-on: stored tokens are removed from extension storage, returning the user interface to the unauthenticated state. In other words, logout clears

the RCA session maintained by the browser add-on, but it does not, by itself, terminate the user's broader Keycloak single sign-on session.

From a security and architectural perspective, this mechanism is suitable for the RCA's role within DietWise. It combines standards-based identity management, separation of concerns between the browser add-on and the backend services, and a browser-compatible PKCE-enhanced flow designed for public clients. The callback page of the backend plays an important integration role by bridging the completion of external authentication with the internal logic of the extension, while the actual credentials remain under the control of the identity provider. This design supports secure access to DietWise services and provides a practical foundation for authenticated browser-based interaction in the RCA environment.

The following is a diagram of the login process:

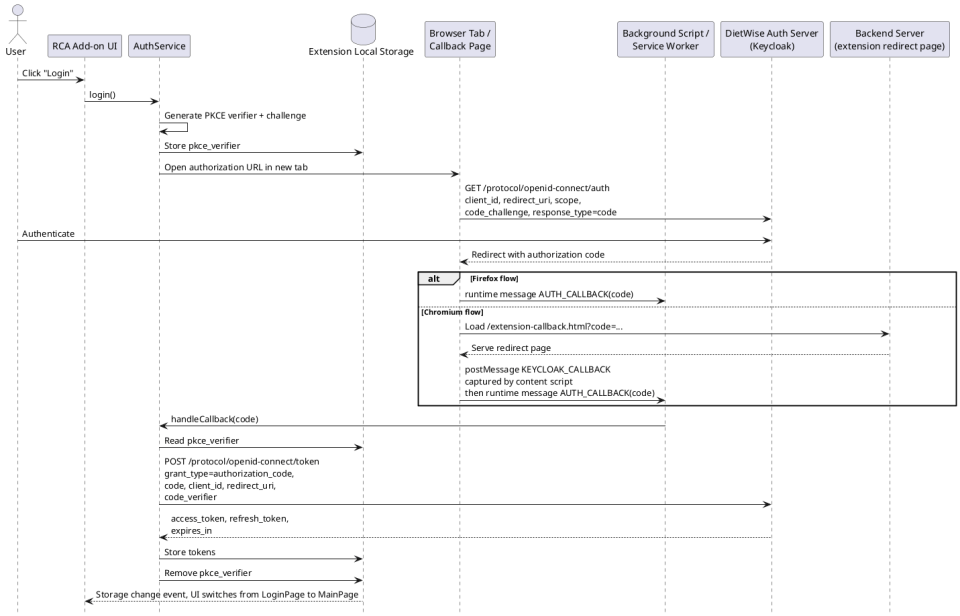


Figure 23 Login process for a browser extension (the RCA)

The following diagram depicts the subsequent API calls and the logout:

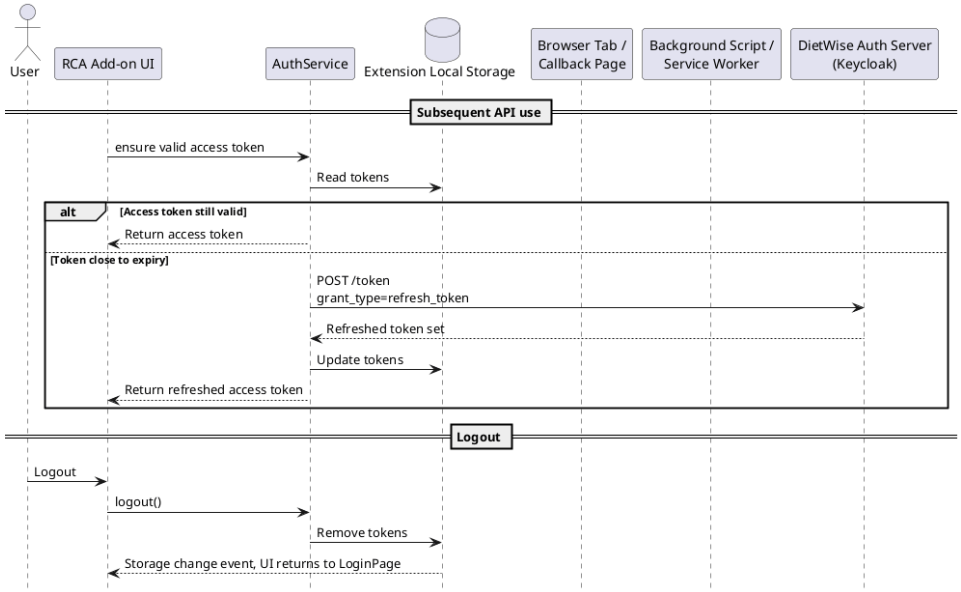


Figure 24 API calls and logout for RCA

7.2. DEPLOYMENT AND ACCESS

The application is deployed in the browser extension marketplaces for Chrome and Firefox.

7.3. USER INTERFACE

This is a tour of the application, exploring the available user scenarios. The UI is kept similar to the one of MyRecipeWatch, to the extent possible.

The user journey starts with users finding themselves on a recipe page. The RCA addon can be activated using the icon button at the top of the browser.

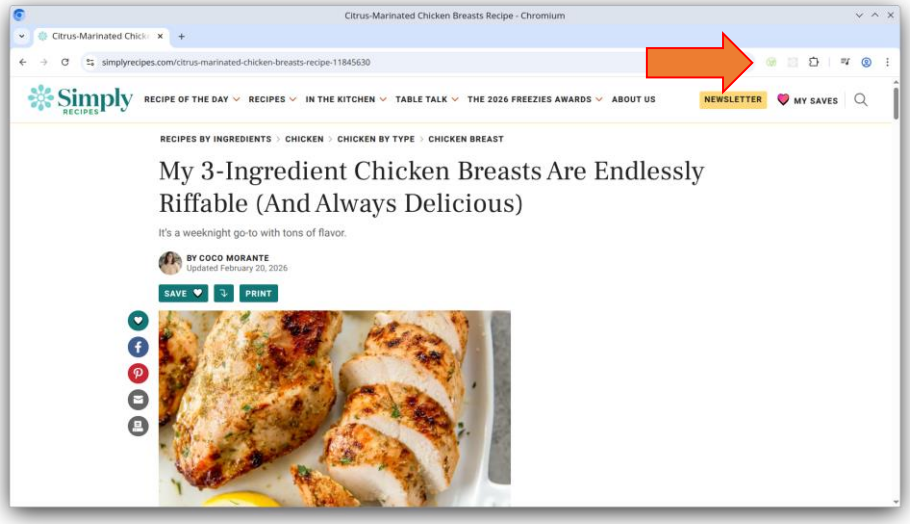


Figure 25 A browser displaying a recipe – RCA is closed

The browser add-on side panel opens. Initially, the user has not logged in. The application provides instructions for logging in or registering.

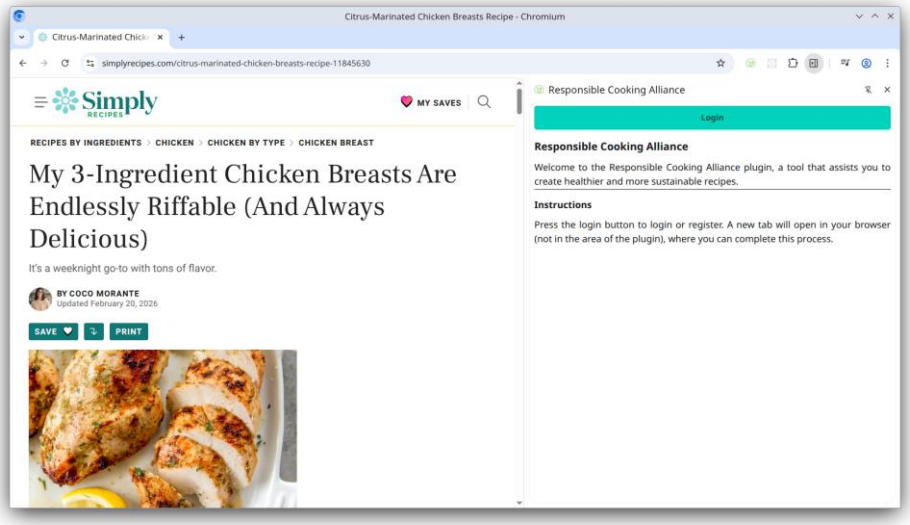


Figure 26 RCA home page

After clicking the login button, users enter their credentials (or follow the registration workflow). Note that the login window opens in a normal browser tab. This is necessary for the OAuth2 login.

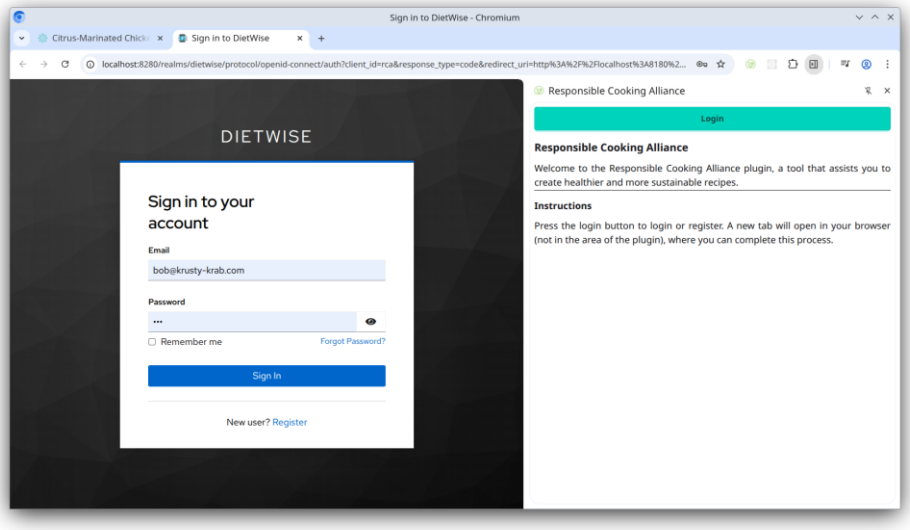


Figure 27 RCA login

When the login is complete, the tab closes automatically or prompts the user to close it (depending on the browser and user settings). The login tab is no longer needed. The application is configured to remember the user's credentials, long period of time (one month if the user does not log back in). So, the login process will occur infrequently.

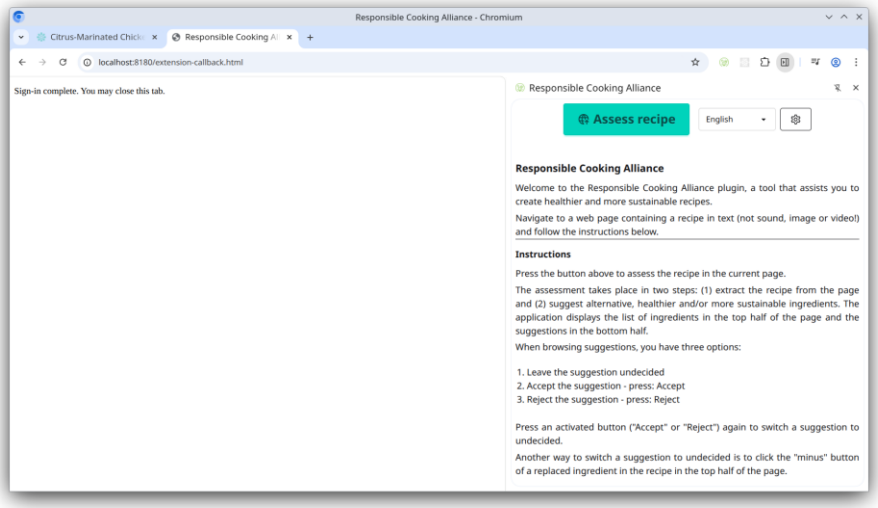


Figure 28 RCA recipe assessment page after successful login

The application is simple based around this page, the main page as we call it. The two other pages are the login, presented earlier, and the configuration, accessible through the gear button. Pressing it, reveals the following configuration screen. Note that the login tab is closed:

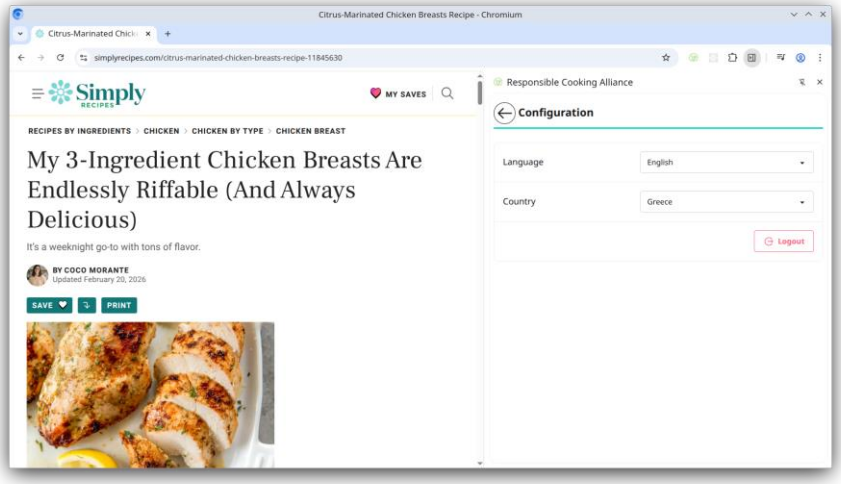


Figure 29 RCA configuration

Users can configure the UI language and the country for which the system provides cost hints for each suggested alternative. Going back to the main page (previous screenshot), we see the two main control

buttons of the application. The "Assess recipe" button begins the assessment workflow. Next to it, the recipe language selection button lets the user select the recipe language. Clicking "Assess recipe," interacts with the server in two phases. First, the recipe is analyzed to extract its ingredients:

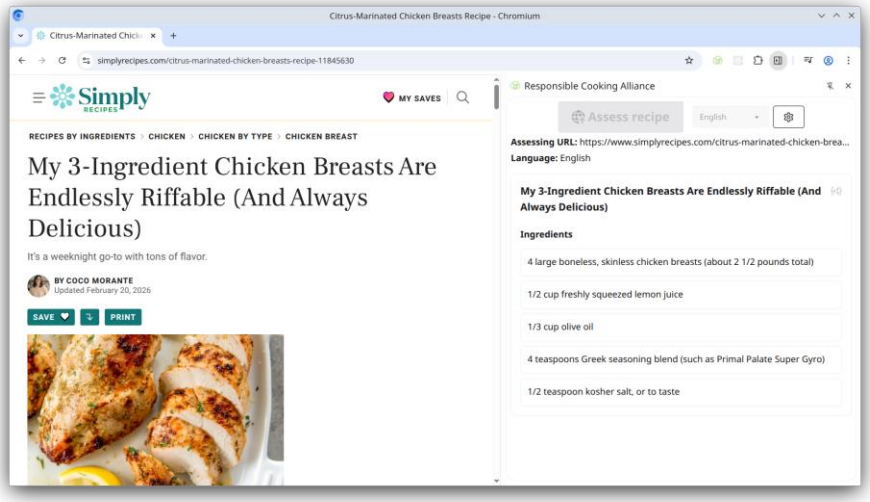


Figure 30 RCA assessing the recipe on the current tab

Then, when the AI has finished assessing the recipe, the app displays the suggestions and the rating:

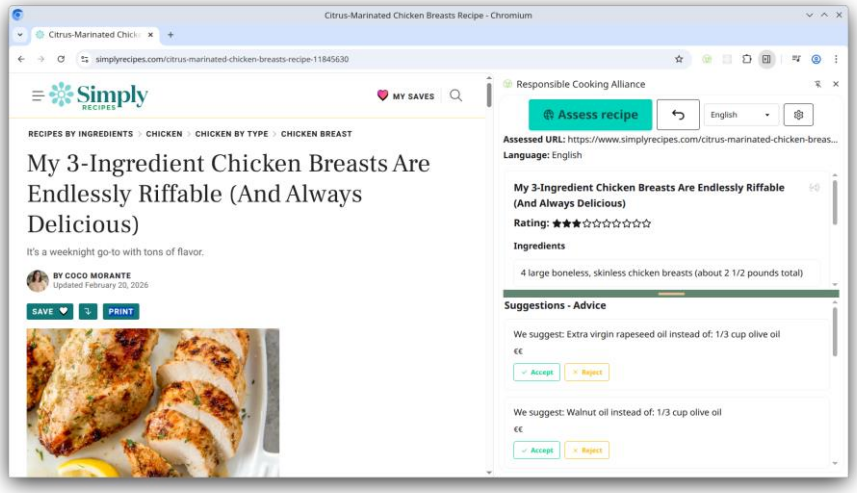


Figure 31 RCA displaying suggestions

The suggestions are ingredient substitutions. There may be multiple or no suggestions per recipe ingredient. The two panes of the display, top: ingredients, bottom: suggestions, scroll independently and the green divider between them is adjustable. In the next screenshot we have scrolled the bottom pane to display suggestions about the "skinless chicken breasts" ingredient:

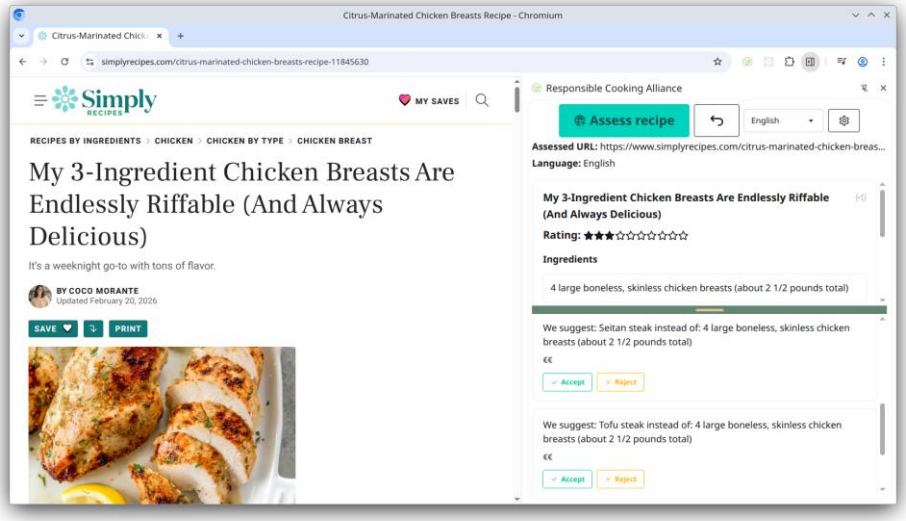


Figure 32 RCA displaying suggestions about the visible ingredient

Users press the "Accept" button to mark their acceptance of a suggestion. This has no effect on the actual recipe on the real web page, but marks the ingredient as below:

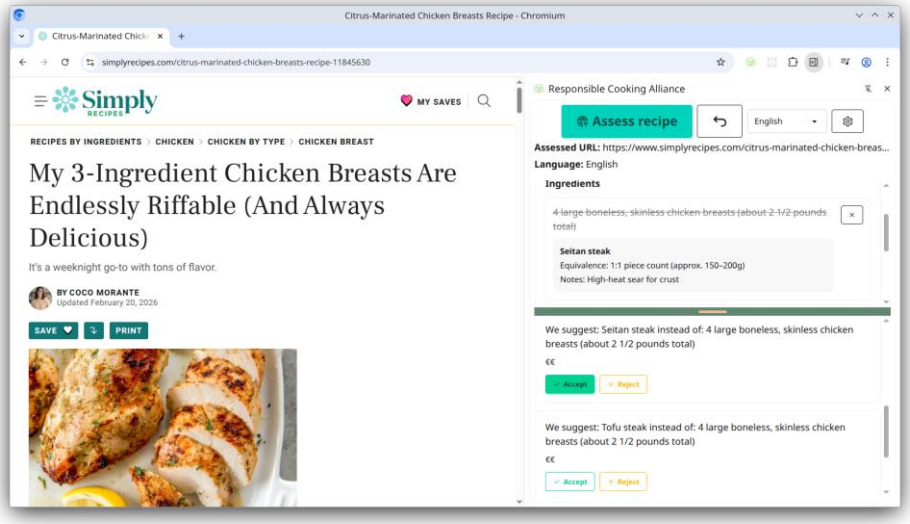


Figure 33 RCA, user has accepted a suggestion

Accepting a different substitution for the same ingredient, undoes the previous accepted one:

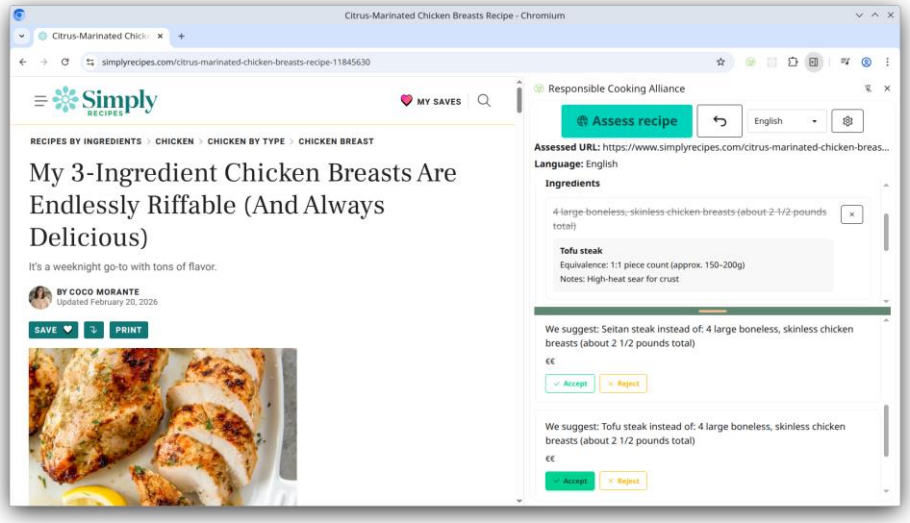


Figure 34 RCA, user has accepted a different suggestion

Users can actively reject suggestions. The application records their preferences for statistical reasons.

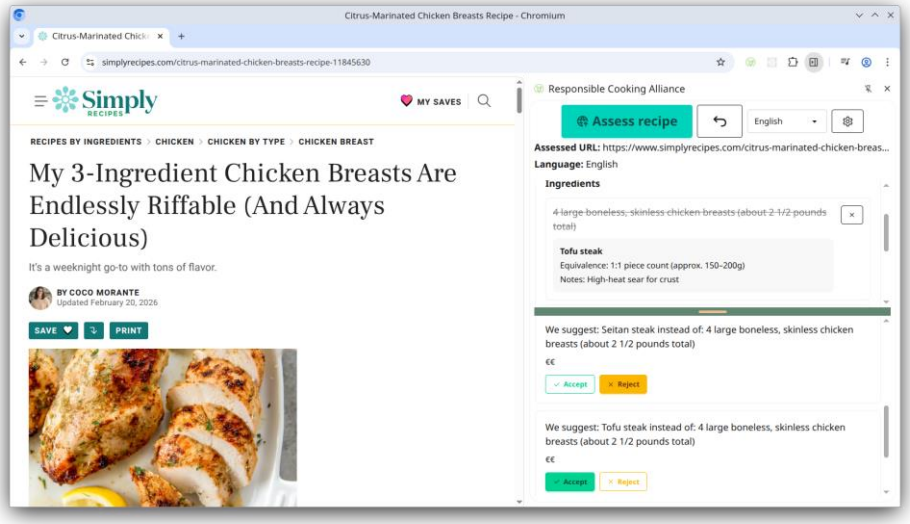


Figure 35 RCA, user has accepted a suggestion and rejected another

The final feature is that the addon remains visible even if the user switches tabs, as seen in the case of the login. If a user opens a different tab, it may be confusing to tell whether the information displayed by the RCA is about this tab or another. The "Assessed URL" field is a hint, but URLs may be long and tedious to read quickly. Therefore, there is a visual cue, the red triangle before the "Assessed URL", to signal to the user that the information of RCA is not about the current tab.

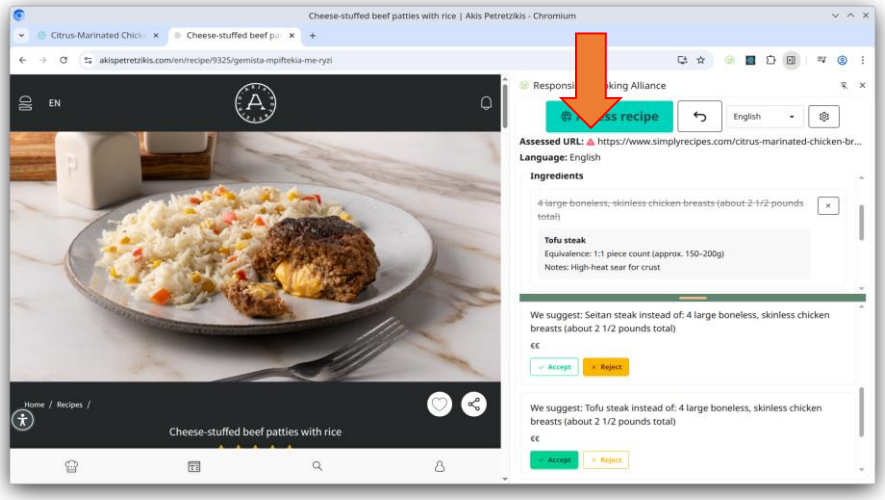


Figure 36 RCA, warning about displaying suggestions for the recipe of a different tab

8. DEPLOYMENT

This project is deployed as a set of Docker Compose stacks on a single host, code name gaia. The overall design separates application runtime concerns from edge routing and TLS termination, while also keeping environments isolated from one another. The deployment is intentionally simple: it does not rely on Kubernetes or any external orchestration platform, and instead uses Docker Compose as the primary mechanism for building, configuring, and running the services. This mechanism makes the infrastructure easy to set up and maintain.

A separate machine with a capable graphics card and CUDA runs Ollama.

Figure 37 DietWise deployment, high-level view

	gaia	ollama
CPU	8 cores (virtualized)	8 cores
RAM	16GB	128GB
GPU	-	Nvidia A40 48GB VRAM Nvidia Tesla T4 16 VRAM

Table 27 DietWise server characteristics

8.1. ENVIRONMENT SEPARATION

The system is organized into distinct environments: local, dev, and prod. Each environment contains its own application stack, i.e., PostgreSQL, Keycloak, and the application-specific services such as the renderer and the backend. This separation allows each environment to evolve and operate independently, with its own configuration, volumes, runtime settings, and service lifecycle. In practice, this means development and production-style deployments can coexist without sharing the same internal containers or persistent state. The exact service definitions may differ slightly between environments, but the overall deployment model remains the same.

The local environment is for testing the configuration only. Dev is for running a public version of the application for internal users to test. Here we deploy the "mobile preview" version of MyRecipeWatch. Date in the dev environment should be considered expendable - the development team is free to destroy and recreate them at any time. There are no availability guarantees. The production environment is what real users will be seeing. It is configured to automatically restart any services that crash. Backup policies are manual – this is not a mission-critical system; only statistical data will be lost.

8.2. STACK SEPARATION

The deployment is split into two major layers:

Application stacks: Each environment has its own Docker Compose application stack. This stack contains the application services and the supporting infrastructure they require internally. These services are not intended

to be exposed directly to end users. Instead, they are reachable only through internal Docker networking and through the ingress layer.

Ingress stack: A separate ingress stack is responsible for handling inbound HTTPS traffic (HTTP redirects to HTTPS). This stack runs Caddy[28] as the reverse proxy and TLS terminator. It provides routing from public hostnames to internal services, HTTPS termination, certificate acquisition and renewal, and a single controlled exposure point on ports 80 (only for redirecting to HTTPS) and 443. This separation keeps edge concerns independent from application concerns. It also allows application stacks to be updated without changing the public routing layer unless necessary.

8.3. NETWORKING MODEL

The application and ingress layers communicate through Docker networking. The ingress stack connects to the environment stacks over a shared edge network, allowing Caddy to proxy traffic to the appropriate internal services. The key points of this implementation are that clients connect only to the ingress host, and backend services remain internal, not publicly exposed. This design reduces the external attack surface and makes routing behavior easier to reason about.

8.4. TLS AND SSL KEY MANAGEMENT

TLS is managed centrally by Caddy in the ingress layer. This means application containers do not need to manage certificates themselves. By centralizing certificate management, the deployment avoids duplicating SSL configuration across services. It also ensures that certificate lifecycle concerns remain isolated to the ingress stack.

Persistent storage is important here: certificate data and Caddy configuration state must survive container restarts and upgrades. For that reason, the ingress stack uses persistent volumes for Caddy data and configuration.

Certbot[29] is used to support certificate issuance and renewal. Its role is to request certificates from the certificate authority, maintain the certificate material on disk, and refresh certificates before they expire. This keeps certificate lifecycle management outside the application services and places it alongside the ingress infrastructure where it belongs.

For automated renewal, the deployment includes a scheduled cron-based renewal job. This job runs periodically and checks whether any certificates are due for renewal. If renewal is needed, Certbot updates the certificate files in the persistent storage used by the ingress layer. This avoids manual certificate rotation and reduces the operational risk of expired certificates. The certificate automation is inactive in the local environment. It uses self-signed certificates instead.

9. IMPACT AND FUTURE DEVELOPMENT

9.1. AI INTEGRATION INTO THE DIETARY ADVICE PROCESS

DietWise successfully integrates AI into the field of healthy and sustainable nutrition through the two ICT solutions. The AI is tightly controlled as part of a deterministic algorithm. It gives the DietWise ICT tools the

[28] https://caddyserver.com/
[29] https://certbot.eff.org/

capability of understanding and extracting information from a natural language text, assessing the role of an ingredient given the instructions of a recipe, and proposing suggestions based on the extracted data and natural language instructions provided by human expert dieticians. At the same time, the rigid algorithmic nature of the implementation ensures that it follows closely the guidelines set by the experts. The team has access to all intermediate results of reasoning, thus can deterministically control how they are used. This greatly reduces the potential of the system to produce hallucinated or arbitrary suggestions. It allows the development of small, localized corrections and adjustments applicable to the parts of the algorithm that fail, while keeping the parts that function well. This design enhances not only the initial development but also the long-term maintainability of the system. In contrast, training or fine-tuning a sufficiently large AI to produce the same results gives no opportunity for partial corrections. If the system does not function properly, the training or fine-tuning has to be repeated from the start with corrected, adjusted, and/or increased number of inputs. The amount of input needed to train or fine-tune an AI is large, so changes inevitably take time. Overall, this approach tries to bring the best of both worlds, algorithmic and AI, to the final solution. The system is more predictable and easier to reason about than a black-box, AI-only solution, while retaining the ability to understand natural language inputs and instructions.

On the other hand, a properly trained or fine-tuned and sufficiently large AI would have more degrees of freedom, maybe to produce results beyond the ones already prescribed by the experts. It would be interesting to see research investigating this direction as well, considering that more degrees of freedom to the AI require tighter control over the produced content with rigorous safety mechanisms.

9.2. EXTENSIBLE DIETARY ADVICE PLATFORM

Given the time allocated to developing the ICT solutions, it was not possible to create a fully featured, commercial-grade solution. The focus of the implementation was on a technically sound, secure solution that incorporates state-of-the-art technologies, is extensible to meet future needs, and potentially allows other actors to continue the work started in this project. For example, the mobile app and browser add-on can present advice generated in a simple, standards-based format generated by any backend. Any other project could use these applications to explore other methods of producing the advice. The same is true for the backend: if it produces successful results, it is easy to enhance the capabilities of the applications, replace or complement them with other, more feature-rich or more specialized ones for the target audience. DietWise is implemented as a robust steppingstone towards the ultimate goal of providing the world with tools that offer focused, personalized nutritional advice, empowering citizens to shift their dietary habits towards health and sustainability.

Currently, the ICT tools can extract a recipe directly from a website. This and the overall simplicity of the applications lowers the barrier a user must overcome to use them. The ease of applying suggestions with the press of a button further lowers the barrier of usage and adoption. The ease of use is especially important for enabling the participation/inclusion of vulnerable people. The technical framework is pluggable enough to receive the recipe content from alternative sources and still produce the same results with little extra effort. A future improvement could be to plug a speech-to-text transformer before the input to DietWise. The transformer would be capable of converting, e.g., a video or audio recording of a cook preparing a recipe into text that serves as input to the same core algorithm. This is the idea of building a foundation for current research and future improvement.

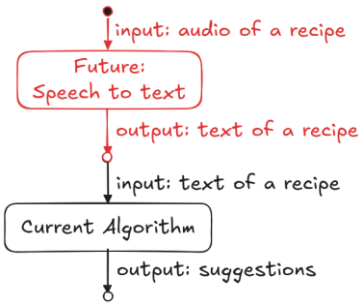


Figure 38 Possible future improvement: read recipes from more content types, not just text

The expert team created the rules for producing cooking advice. The rules are linked to the nutritional guidelines of the Global Burden of Disease (GBD 2023) study. This set of rules is expandable for a low cost and effort, i.e. no extra LLM training or fine-tuning needed, just entries to a database. The data model is flexible to allow this as well as any number of suggestions per rule. The model and the data are reusable.

The user interface is simple yet functional. There was no special effort to make it more aesthetically pleasing. This was a two-fold decision. First, it was deemed more critical to focus on the essential functionality in the limited time at our disposal. Second, it is generally more efficient to start with a minimum viable product (MVP) that gradually incorporates user feedback, rather than starting with an elaborate design that may fail to please the general public, thus resulting in wasted resources. The pilot phase of the project is the tool for gathering feedback from a broader user base that uses the actual applications, in comparison to the input from co-creation workshops (WP4).

9.3. RESEARCH VALUE

The specifics of LLM integration into a suggestion pipeline for healthy and sustainable nutrition open interesting paths to explore. The pilot test phase and the statistics provided by the applications are the instruments to gauge the efficiency of the implementation and whether the rules and assumptions made are sufficient.

The suggestion pipeline is architected as a rigid, deterministic algorithm, developed by experts, aided and enhanced by the AI's natural language processing capabilities. We managed to get results from small models, without any special training or fine-tuning, only prompt engineering. The model type, size, temperature, top-p, and top-k parameters are easily customizable for each AI task individually. Experimenting with these values, we got an idea of how to use a small LLM to produce answers to focused questions, as well as insights for future directions. Keeping the LLM size small helps keep the cost of the solution proportionally low. Similarly, the places where the solution would benefit from a larger, more capable LLM were identified. It seems that a very demanding task for small-sized LLMs is the production of structured output, e.g., JSON, for a complex response.

The applications give the dietitian research community a means to evaluate the efficiency and sufficiency of the logic behind the suggestions. In DietWise the experts came up with a set of personalization attributes, dietary recommendations based on the GBD, rules for providing advice, and the guidelines to apply them. The statistics to be gathered, especially during the pilots, will be useful for assessing them and determining points that need improvement.

Finally, the applications are a research platform for the behavioral scientists to understand and evaluate the impact of the advice, suggestions, and motivational boosts on the acceptance of the recommendations.

9.4. POTENTIAL FUTURE TECHNICAL ENHANCEMENTS

A possible future development pathway for the DietWise ICT solutions concerns their technical evolution and scalability. One important direction is the potential adoption of more capable foundation models, where the additional computational and financial cost is justified by measurable improvements in reasoning quality, robustness, and handling of complex dietary cases. In parallel, future work may investigate domain adaptation or fine-tuning strategies to improve the consistency and relevance of nutrition-related suggestions, especially in cases that require more specialized dietary interpretation.

Another promising direction concerns the further refinement of the existing hybrid architecture, which already combines AI-based components with explicit nutrition and decision rules derived from dietary guidance. Future work may strengthen this combination by expanding the rule base, improving the interaction between deterministic recommendation logic and AI-driven interpretation of recipe content, and increasing the transparency and explainability of the resulting suggestions. This is particularly important in the context of food and health-related recommendations, where consistency, traceability, and alignment with nutrition guidelines remain essential. In addition, future technical work may focus on extending interoperability through plugins, browser extensions, and integrations with a wider range of recipe platforms and digital food environments, thereby increasing the practical reach of the solutions.

Finally, the technical development of the DietWise tools could be expanded by integrating with adjacent digital services, such as meal planning systems, shopping list tools, grocery platforms, or health-related applications. Such integrations could strengthen the continuity of the user's journey across different decision points and create a more comprehensive digital ecosystem to support healthy and sustainable food choices.

9.5. SUGGESTIONS FOR IMPROVING THE USER EXPERIENCE

A further area for future development concerns the improvement of the overall user experience of the DietWise ICT solutions. At present, the applications are primarily oriented towards individual users. Future iterations could extend this approach to household-level use cases, thereby supporting families or other shared meal-planning contexts involving different ages, genders, and nutritional needs. Such an extension would better reflect real-life food decision-making and could increase the practical relevance of the tools in everyday domestic settings.

Additional enhancements may focus on improving convenience and long-term engagement. For example, users could be given the possibility to save preferences for specific recipes, allowing the system to remember prior choices and reduce repetitive interaction. Similarly, support for maintaining a personal catalog of recipe links could facilitate easier reuse of online content and encourage more sustained use of the platform over time. In parallel, further refinement of the visual design and overall look and feel could improve usability, trust, and clarity, especially in cases when users are asked to interpret recommendations and make food-related decisions.

Future work may also place stronger emphasis on accessibility and inclusion. This could include usability improvements for elderly users, people with visual impairments, and users with low levels of digital or nutrition literacy. Although the interaction flows are already simple, research on even simpler flows and alternative explanation styles could further support vulnerable population groups and broaden the reach of the solutions. Finally, the user experience could be strengthened by providing clearer, more precise plain-language explanations of why a given suggestion is made, thereby making recommendations easier to understand and more acceptable in practice.

Commented [SD44]: Suggestions for improvement of...

Commented [ŽK45]: I made changes and the font has changed. Can't undo, sorry...

Commented [SD46]: Would it make sense to add a suggestion for preparation suggestions? Baking at lower temp may be better. Certain combination, etc.

Commented [ni47R46]: Good idea, added but in the section about *Logic* enhancements.

9.6. POTENTIAL FUTURE LOGIC ENHANCEMENTS

Another important direction for future development concerns the further enhancement of the recommendation logic and personalization mechanisms of the DietWise ICT solutions. At present, personalization is based on a limited set of user characteristics. Future iterations could extend this approach by incorporating additional parameters, such as dietary habits, dietary restrictions, personal preferences, health-related goals, and broader lifestyle factors. This would allow the systems to generate recommendations that are more closely aligned with the specific circumstances and needs of individual users.

Future work may also focus on increasing the depth and sophistication of the underlying suggestion logic. This could involve expanding the range of target ingredients considered by the system, covering a broader set of ingredient-role combinations, and improving the precision with which substitutions or corrections are proposed. In addition, recommendation logic could be made more context-aware by accounting for factors such as cuisine type, meal context, seasonality, budget sensitivity, and local ingredient availability, or even by recommending additional ingredients to produce nutritionally optimal combinations. Such enhancements would improve both the relevance and the practical applicability of the generated suggestions.

A major enhancement in the logic of the system, which would also reflect the usefulness of the tools, could be the inclusion of suggestions about preparation instructions. Alternative cooking methods and preparation details, like baking temperature, might be suggested to optimize the nutritional value of the recipe.

A further promising direction is the development of more advanced household-aware personalization mechanisms. In many real-life settings, food choices are made not for a single person but for a group of users with different nutritional profiles and preferences. Supporting such shared-meal contexts would require recommendation logic capable of balancing multiple profiles within a single decision-making process. Finally, future developments could further strengthen alignment with national or regional dietary guidelines, as well as with local cuisines, units, and food practices, thereby increasing the adaptability and acceptability of the solutions across different localities.

9.7. POTENTIAL FUTURE ENHANCEMENTS FOR INFLUENCERS

A further direction for future development concerns strengthening the value of proposition of the Responsible Cooking Alliance (RCA) for influencers and other food-environment actors. In its current form, the RCA primarily provides a general mechanism for assessing alignment with nutrition and sustainability criteria. Future iterations could extend this functionality to deliver more direct, tangible value to participating influencers, thereby increasing engagement and supporting broader uptake of the initiative.

Any future improvement of the RCA application should be developed in alignment with, and as part of, the broader RCA initiative and governance framework. In this context, one possible direction would be to introduce structured mechanisms through which influencers could contribute to the further refinement of dietary rules and assessment criteria. For example, influencers could be given the possibility to propose additions or modifications to existing rules, which would then be submitted to a peer-review process involving other relevant participants and, where appropriate, expert validation. Based on the outcome of this review process, proposed changes could be accepted, revised, or rejected before being incorporated into the system. Such a mechanism could strengthen stakeholder engagement, support the continuous evolution of the RCA, and reinforce a sense of shared ownership among participating influencers. In the longer term, this participatory model could also be linked to recognition mechanisms, such as visible status indicators reflecting the quality and acceptance of a participant's contributions within the initiative.

9.8. FUTURE EVALUATION AND VALIDATION ENHANCEMENTS

Another important area for future development is strengthening evaluation and validation activities for the DietWise ICT solutions. While the current implementation focuses on demonstrating feasibility and supporting pilot deployment, future work could introduce more systematic assessment frameworks to evaluate the quality, reliability, and practical value of the recommendations generated by the tools. In particular, benchmarking recommendation outputs against expert review by nutrition professionals could provide a stronger basis for assessing scientific soundness and identifying areas requiring further refinement.

Future validation efforts may also include controlled experimentation on how recommendations are presented to users. For example, A/B testing could be used to compare alternative formulations, rankings, timings, or presentation styles of suggestions, with the aim of improving user acceptance and interaction quality. Such experimentation would provide useful evidence on which interface and communication choices are most effective in supporting healthier and more sustainable food decisions.

10. CONCLUSIONS

This deliverable (D5.2) has presented the technical realization of the DietWise ICT solutions, developed under Task T5.2, and has documented the design, implementation, and deployment of both MyRecipeWatch (RW) and the Responsible Cooking Alliance (RCA) tool. It has provided a comprehensive account of the system architecture, the AI-powered reasoning pipeline for ingredient substitutions, and the supporting infrastructure enabling scalable and secure operation.

The work reported demonstrates that the DietWise ICT solutions have reached a mature and operational stage, with fully implemented beta versions ready to support pilot activities under WP7. The adopted architectural approach, grounded in scalability, maintainability, and security, ensures that the system can accommodate future extensions while maintaining robustness and performance. The modular design of the backend, combined with clearly defined interfaces between components, facilitates iterative development and integration of additional features.

A key contribution of this work is the development of an AI-powered suggestions pipeline, which translates expert nutritional knowledge into actionable, recipe-level ingredient substitutions. By combining structured datasets, expert-informed taxonomies, and staged LLM prompting strategies, the system achieves a balance between flexibility and control. Particular emphasis has been placed on transparency, reproducibility, and alignment with nutritional guidelines. At the same time, user safety and privacy have been treated as fundamental design principles, ensuring that no personal data is exposed to AI components and that personalization mechanisms remain external and controlled.

The deliverable also highlights the importance of close collaboration with stakeholders and domain experts. Insights gathered through earlier work packages have been effectively translated into concrete ICT features, user interface decisions, and system functionalities. This ensures that the solutions are not only technically sound but also relevant, usable, and aligned with user expectations across different contexts.

Looking forward, the DietWise ICT solutions provide a strong foundation for pilot testing and iterative refinement. The upcoming activities under WP7 will be critical in validating system performance, usability, and behavioural impact in real-world settings. Feedback collected during these pilots will inform further improvements, including enhancements in multilingual support, contextual factors such as cost and seasonality, and the overall user experience.

In conclusion, D5.2 establishes that the DietWise project has successfully delivered robust, scalable, and user-oriented ICT solutions that are ready for pilot deployment. While further refinement is anticipated, the current implementation already meets its core objectives and provides a solid baseline for continued development, evaluation, and impact generation within the project's lifecycle.

Annex A Supporting material

CODE REPOSITORIES

All the code is open source. The repositories for each application can be found at the following locations:

URL	Repository of application
https://github.com/nikospara/dietwise-renderer	The DietWise Rendererer
https://github.com/nikospara/dietwise-backend	The backend application
https://github.com/nikospara/dietwise-recipe-watch	MyRecipeWatch
https://github.com/nikospara/dietwise-responsible-cooking-alliance	Responsible Cooking Alliance tool

Table 28 Code repositories

Annex B Keycloak configuration

This Annex highlights the Keycloak production configuration. Details and more instructions for a development environment are found in the file `dietwise-docker/dietwise-docker-keycloak/README.md` of the `dietwise-backend` repository. The code also contains a Keycloak realm configuration file *for development purposes only*, used in the creation of the *development* Docker images under dietwise-docker/dietwise-docker-keycloak/src/main/docker/dietwise-realm.json in the same repository.

- A realm dedicated to DietWise with name `dietwise` and display name "DietWise"
  - In the Login tab, User registration, Forgot password, Remember me: On
  - Email as username, Login with email: On
  - Verify email: On
  - Sessions: Make sure "Offline session settings" -> "Offline Session Idle" is set to something like 30 Days
  - User profile:
    - Row firstName -> Edit -> "Required field": No -> Save
    - Row lastName, the same
  - Realm roles:
    - Create Role: "influencer"
- Client for RW
  - General settings:
    - Client type: OpenID Connect
    - Client ID: recipewatch
    - Name: MyRecipeWatch
  - Capability config:
    - Client authentication: Off
    - Authorization: Off
    - Authentication flow: *only* "Standard flow"
    - Require PKCE: On
    - PKCE Method: S256 - also in the "Settings" tab, under the "Capability config" section

- Login settings:
  - Valid redirect URIs:
    - https://gaia.ispatial.survey.ntua.gr/recipewatch/authcallbackeu.dietwise.recipewatch://authcallback
    - https://dietwise.ispatial.survey.ntua.gr/recipewatch/authcallback
    - http://localhost:5173/authcallback
    - http://localhost:5173/recipewatch/authcallback
    - https://gaia.ispatial.survey.ntua.gr/recipewatch/authcallback
    - https://gaia.ispatial.survey.ntua.local/recipewatch/authcallback
  - Valid post logout redirect URIs:
    - https://gaia.ispatial.survey.ntua.gr/recipewatch/endsessioneu.dietwise.recipewatch://endsession
    - https://dietwise.ispatial.survey.ntua.gr/recipewatch/endsession
    - http://localhost:5173/endsession
    - http://localhost:5173/recipewatch/endsession
    - https://gaia.ispatial.survey.ntua.gr/recipewatch/endsession
    - https://gaia.ispatial.survey.ntua.local/recipewatch/endsession
  - Web origins:
    - http://localhost:5173
    - https://localhost

- Client for RCA
  - General settings:
    - Client type: OpenID Connect
    - Client ID: rca
    - Name: Responsible Cooking Alliance
  - Capability config:
    - Client authentication: Off
    - Authorization: Off
    - Authentication flow: Check *only* "Standard flow"
    - Require PKCE: On
    - PKCE Method: S256 - also in the "Settings" tab, under the "Capability config" section
  - Login settings:
    - Valid redirect URIs:
      - http://localhost:5173
      - http://localhost:8180/extension-callback.html
      - https://e9daffa4e57cef0785f2756a14b247b128162e80.extensions.allizom.org/
      - https://gaia.ispatial.survey.ntua.gr/extension-callback.html
      - https://gaia.ispatial.survey.ntua.local/extension-callback.html
      - https://dietwise.ispatial.survey.ntua.gr/extension-callback.html
    - Valid post logout redirect URIs:
      - TODO(empty)
    - Web origins:
      - TODO(empty)

Annex C API usage details and examples

THE DIETWISE RENDERER

API Endpoints

GET /health

Simple readiness/liveness endpoint.

Example response:

```
{ "ok": true }
```

POST /render

Renders a page and optionally transforms the result.

Request body:

```
{
  "url": "https://example.com/recipe",
  "simplify": true,
  "includeJsonLdRecipes": true,
  "timeout": 15000,
  "viewport": {
    "width": 1280,
    "height": 900
  },
  "outputMinimalText": false
}
```

Request fields:

- url: required string. The target URL, or a fixture filename like 001.html in test mode.
- simplify: optional boolean. If true, run the HTML cleaner.
- includeJsonLdRecipes: boolean. If true, extract JSON-LD recipe objects.
- timeout: optional navigation timeout in milliseconds.
- viewport: optional object with width and height.
- outputMinimalText: optional boolean. If true and simplify is enabled, return minimal plain text instead of cleaned HTML.

Response Examples

1. Raw rendered HTML only

Request:

```
POST /render
Content-Type: application/json
 {
  "url": "https://example.com/recipe",
  "simplify": false,
  "includeJsonLdRecipes": false
}
```

Response:

```
{
  "output": "<html><head>...</head><body>...</body></html>",
  "finalUrl": "https://example.com/recipe"
}
```

Notes:

- output is the full rendered DOM returned by page.content()
- finalUrl reflects the final page URL after redirects

2. Rendered HTML plus JSON-LD recipes

Request:

```
{
  "url": "https://example.com/recipe",
  "simplify": false,
  "includeJsonLdRecipes": true
}
```

Response:

```
{
  "output": "<html><head>...</head><body>...</body></html>",
  "finalUrl": "https://example.com/recipe",
  "jsonLdRecipes": [
    {
      "name": "Classic Tomato Soup",
      "recipeYield": "4 servings",
      "recipeIngredients": [
        "2 tbsp olive oil",
        "1 onion",
        "800 g tomatoes"
      ],
      "recipeInstructions": [
        "Heat the oil in a large pot.",
        "Add onion and cook until soft.",
        "Add tomatoes and simmer for 20 minutes."
      ]
    }
  ]
}
```

3. Simplified HTML output

Request:

```
{
  "url": "https://example.com/recipe",
  "simplify": true,
  "includeJsonLdRecipes": false,
  "outputMinimalText": false
}
```

Response:

```
{
  "output": "<h1>Classic Tomato Soup</h1><p>A simple
soup...</p><h2>Ingredients</h2><ul><li>2 tbsp olive oil</li><li>1
onion</li></ul><h2>Method</h2><ol><li>Heat the oil...</li></ol>",
  "finalUrl": "https://example.com/recipe"
}
```

Notes:

- The exact output depends on source HTML.
- The cleaner preserves a limited semantic structure rather than page layout.

4. Minimal text output

Request:

```
{
  "url": "https://example.com/recipe",
  "simplify": true,
  "includeJsonLdRecipes": true,
  "outputMinimalText": true
}
```

Response:

```
{
  "output": "# Classic Tomato Soup\n\nA simple soup...\n\n## Ingredients\n\n- 2 tbsp olive
oil\n- 1 onion\n\n## Method\n\n- Heat the oil...\n- Add onion...",
  "finalUrl": "https://example.com/recipe",
  "jsonLdRecipes": [
    {
      "name": "Classic Tomato Soup",
      "recipeYield": "4 servings",
      "recipeIngredients": [
        "2 tbsp olive oil",
        "1 onion"
      ],
      "recipeInstructions": [
        "Heat the oil...",
        "Add onion..."
      ]
    }
  ]
}
```

5. Fixture-mode request

With DW_RENDERER_TEST_DIR set:

```
{
  "url": "001.html",
  "simplify": true,
  "includeJsonLdRecipes": true
}
```

Response:

```
{
  "output": "<h1>Fixture Recipe</h1><ul><li>Ingredient A</li></ul>",
  "finalUrl": "001.html",
  "jsonLdRecipes": [
    {
      "name": "Fixture Recipe",
      "recipeIngredients": [
        "Ingredient A"
      ],
      "recipeInstructions": [
        "Do something"
      ]
    }
  ]
}
```

Error Response Examples

Invalid input

```
{ "error": "Error: Invalid or empty url" }
```

Typical status: 500 in the current implementation, although semantically this is a client error.

Upstream HTTP failure

```
{
  "error":                    "Error:                    HTTP                    404",
  "httpStatus":                                                                   404
}
```

Typical status: 400

Network failure

```
{
  "error": "Error: page.goto: net::ERR_NAME_NOT_RESOLVED at https://bad-host.example/"
}
```

Typical status: 400

THE BACKEND APPLICATION

API Endpoints

GET /api/v1/version

Returns the deployed backend version and source revision identifier.

Example response:

```
{ "version": "1.0.0-SNAPSHOT", "gitHash": "a1b2c3d" }
```

GET /api/v1/personal-info

Returns the stored personal information of the current user.

Example response:

```
{ "gender": "FEMALE", "yearOfBirth": 1990 }
```

POST /api/v1/personal-info

Creates or updates the personal information of the current user.

Request body:

```
{ "gender": "MALE", "yearOfBirth": 1988 }
```

POST /api/v1/recipe/assess/markdown

Assesses a recipe from page content already available to the caller, using Markdown input and returning progressive processing results. Intended to be used by RCA which has access to the page content and can simplify & transform it to Markdown in the client.

Indicative request body:

```
{
  "url": "https://example.org/recipe-page",
  "langCode": "en",
  "pageContent": "# Recipe\n\nIngredients...\n\nInstructions..."
}
```

POST /api/v1/recipe/assess/url

Retrieves recipe content from the provided URL, performs extraction and assessment, and returns progressive processing results.

Indicative request body:

```
{
  "url": "https://example.org/recipe-page",
```

```
  "langCode": "en"
}
```

POST to the statistics endpoints

The four statistics endpoints adjust the statistics counters for an ingredient. They URIs are:

1. /api/v1/statistics/increaseTimesAccepted
2. /api/v1/statistics/decreaseTimesAccepted
3. /api/v1/statistics/increaseTimesRejected
4. /api/v1/statistics/decreaseTimesRejected

Indicative request body:

```
{ "suggestionId": "6a5d1f8c-4ef5-4bfa-a4c3-111111111111" }
```

The response contains the new value for the counter:

```
{ "updatedValue": 2 }
```

Response Examples

Message 1: extracted recipe

```
{
  "type": "RECIPES",
  "recipes": [
    {
      "recipe": {
        "name": "Vegetable Soup",
        "recipeIngredients": [
          {
            "id": "e56d730b-4d70-48b8-98b6-331ad7914a6e",
            "nameInRecipe": "2 carrots"
          },
          {
            "id": "9e50135f-a2af-44c0-b48a-dfe40b375aa7",
            "nameInRecipe": "1 potato"
          }
        ],
        "recipeInstructions": [
          "Chop the vegetables.",
          "Boil until tender."
        ]
      },
      "detectionType": "JSONLD"
    }
  ],
  "pageText": "..."
}
```

Message 2: generated suggestions

```
{
  "type": "SUGGESTIONS",
  "suggestions": [
    {
      "id": "6a5d1f8c-4ef5-4bfa-a4c3-111111111111",
      "alternative": "Flaxseed oil",
```

```
      "restriction": "Use cold only",
      "equivalence": "Use 1–2 tbsp per 4p",
      "techniqueNotes": "Use as finishing oil only",
      "target": {
        "type": "INGREDIENT",
        "ingredient": "9e50135f-a2af-44c0-b48a-dfe40b375aa7"
      },
      "ruleId": "353761e8-5bfb-463d-9f33-f98f38214003",
      "recommendation": "Diet low in omega-6 polyunsaturated fatty acids",
      "rationale": null,
      "alternativeComponentNames": [],
      "totalSuggestionStats": {
        "timesSuggested": 144,
        "timesAccepted": 102,
        "timesRejected": 22
      },
      "userSuggestionStats": {
        "timesSuggested": 2,
        "timesAccepted": 1,
        "timesRejected": 1
      },
      "text": "Replace part of the salt with herbs for flavor."
    }
  ],
  "suggestionTemplateIds": [
    "6a5d1f8c-4ef5-4bfa-a4c3-111111111111"
  ]
}
```


Message 3: scoring result


```
{
  "type": "SCORING",
  "scoringData": {
    "totalNumberOfRecomendations": 15,
    "recommendationWeights": {
      "trans fatty acids": "LIMITED",
      "fiber": "ENCOURAGED",
      "calcium": "ENCOURAGED",
      "legumes": "ENCOURAGED",
      "milk": "ENCOURAGED",
      "whole grains": "ENCOURAGED",
      "nuts and seeds": "ENCOURAGED",
      "seafood omega-3 fatty acid": "ENCOURAGED",
      "vegetables": "ENCOURAGED",
      "red meat": "LIMITED",
      "sodium": "LIMITED",
      "fruits": "ENCOURAGED",
      "processed meat": "LIMITED",
      "sugar-sweetened beverages": "LIMITED",
      "omega-6 polyunsaturated fatty acids": "ENCOURAGED"
    },
    "recommendationsPerIngredient": {
      "e56d730b-4d70-48b8-98b6-331ad7914a6e": [
        "sodium",
```

```
        "calcium",
        "milk"
      ],
      "9e50135f-a2af-44c0-b48a-dfe40b375aa7": [
        "fiber",
        "vegetables"
      ]
    }
  }
}
```

Annex D Backend configuration

This annex briefly describes the configuration options explicitly defined by the DietWise backend. It focuses on application-level and deployment-relevant properties, not on the full set of Quarkus defaults.

HTTP AND API

Name	Semantics	Value
quarkus.http.port quarkus.http.test-port	The port to listen; by default we assign a different port to each service	8180
quarkus.http.cors.enabled	Enables Cross-Origin Resource Sharing	true
quarkus.http.cors.origins	Lists the allowed origins for cross-origin requests. The last value, https://localhost, is indeed needed for the application to work on the mobile phone	/chrome-extension://([a-z0-9-]+)/,/moz-extension://([a-z0-9-]+)/,https://localhost

	environment.	
%dev.quarkus.http.cors.origins	Lists the allowed origins for cross-origin requests in the dev environment.	http://localhost:5173,https://gaia.ispatial.survey.ntua.gr,https://gaia.ispatial.survey.ntua.local,/chrome-extension://([a-z0-9-]+)/,/moz-extension://([a-z0-9-]+)/,https://localhost

SECURITY

These settings configure the headings as discussed in the section SYSTEM DESIGN › SECURITY › Backend › Defensive HTTP response headers.

Name	Semantics	Value
quarkus.http.header."Referrer-Policy".value	Controls the Referrer-Policy response header	no-referrer
quarkus.http.header."Permissions-Policy".value	Restricts selected browser capabilities such as camera and microphone access	accelerometer=(), camera=(), geolocation=(), microphone=()
quarkus.http.header."X-Content-Type-Options".value	Enables nosniff protection	nosniff
%prod.quarkus.http.header."Strict-Transport-Security".value	Enables HSTS in production profile	max-age=31536000; includeSubDomains
quarkus.http.header."Cache-Control".value	Disables caching for the configured path	no-store, max-age=0
quarkus.http.header."Cache-Control".path	...which is the RCA login callback page	/extension-callback.html
quarkus.http.header."Pragma".value	Adds no-cache behavior for legacy clients for the configured path	no-cache
quarkus.http.header."Pragma".path	...which is the RCA login callback page	/extension-callback.html
quarkus.http.header."X-Frame-Options".value	Prevents the configured page from being embedded in frames	DENY
quarkus.http.header."X-Frame-Options".path	...only for the RCA login callback page	/extension-callback.html
quarkus.http.header."Content-Security-Policy".value	Defines a restrictive CSP for the configured page	default-src 'none'; script-src 'unsafe-inline'; base-uri 'none'; form-action 'none'; frame-ancestors 'none'
quarkus.http.header."Content-Security-Policy".path	...which is the RCA login callback page	/extension-callback.html

DATABASE AND PERSISTENCE

Note that the connection credentials are used both by the traditional JDBC connection (used by Liquibase) and the main, reactive datasource (used by the application through Hibernate Reactive).

Name	Semantics	Value
quarkus.datasource.db-kind	Declares the database type. In this project it is PostgreSQL.	postgresql
quarkus.datasource.username	Database username; the value in the configuration files is deliberately invalid so that no credentials leak. It forces us to configure it from environment properties, see README.md.	INVALID_USERNAME
quarkus.datasource.password	Same as above, for the database password.	INVALID_PASSWORD
quarkus.datasource.reactive.url	Same as above, for the Hibernate Reactive connection URL.	INVALID_URL
quarkus.datasource.jdbc.url	Same as above, for the JDBC connection URL.	INVALID_URL
quarkus.liquibase.change-log	Location of the Liquibase master changelog file loaded at startup.	changelog.xml
quarkus.liquibase.migrate-at-start	Activate Liquibase when the application starts.	true

APPLICATION METADATA

Name	Semantics	Value
dietwise.version	Application version exposed by the version endpoint.	(auto-completed by build)
dietwise.git-hash	Source revision identifier exposed by the version endpoint.	(auto-completed by build)
dietwise.renderer.allow-cached-test-pages	Allows use of cached renderer test pages instead of always contacting live sites.	Enabled in dev and test, disabled by default.

AI CONNECTIVITY

The configuration for each AI service is kept separate for flexibility. A "*" below marks that the setting is common for all services. The names of the services are: `extract`, `filter`, `roleOrTechnique`, `triggerIngredient`, `ingredientMatchInRecommendations`, `findBestRule`, `suggestAlternatives`.

ENDPOINTS		
Name	Semantics	Value
quarkus.langchain4j.ollama.*.base-url	Ollama base endpoint. This is the same for all services.	http://localhost:11434 Overriden in public environments

TIMEOUTS		
Name	Semantics	Value

quarkus.langchain4j.ollama.*.timeout	Network request timeout	300s (meaning seconds)

AI MODEL SELECTION		
Name	Semantics	Value
For `extract`, `filter`, `roleOrTechnique`, `triggerIngredient`, `ingredientMatchInRecommendations`, `findBestRule`		
quarkus.langchain4j.ollama.*.chat-model.model-id	Model identifier	llama3.1:8b
For `suggestAlternatives`		
quarkus.langchain4j.ollama.*.chat-model.model-id	Model identifier	llama3.1:80b

AI PARAMETERS	
Name	Value
quarkus.langchain4j.ollama.extract.chat-model.temperature	0.3
quarkus.langchain4j.ollama.filter.chat-model.temperature	0.3
quarkus.langchain4j.ollama.roleOrTechnique.chat-model.temperature	0
quarkus.langchain4j.ollama.roleOrTechnique.chat-model.repeat-penalty	1.1
quarkus.langchain4j.ollama.roleOrTechnique.chat-model.num-predict	20
quarkus.langchain4j.ollama.triggerIngredient.chat-model.temperature	0
quarkus.langchain4j.ollama.triggerIngredient.chat-model.repeat-penalty	1.1
quarkus.langchain4j.ollama.triggerIngredient.chat-model.num-predict	20
quarkus.langchain4j.ollama.ingredientMatchInRecommendations.chat-model.temperature	0
quarkus.langchain4j.ollama.ingredientMatchInRecommendations.chat-model.repeat-penalty	1.1
quarkus.langchain4j.ollama.ingredientMatchInRecommendations.chat-model.num-predict	8
quarkus.langchain4j.ollama.findBestRule.chat-model.temperature	0
quarkus.langchain4j.ollama.findBestRule.chat-model.repeat-penalty	1.1
quarkus.langchain4j.ollama.findBestRule.chat-model.num-predict	8
quarkus.langchain4j.ollama.suggestAlternatives.chat-model.temperature	0
quarkus.langchain4j.ollama.suggestAlternatives.chat-model.repeat-penalty	1.1
quarkus.langchain4j.ollama.suggestAlternatives.chat-model.num-predict	8

Annex E LLM Prompts

INGREDIENT ROLE CLASSIFICATION – SYSTEM PROMPT

```
You are a classification model.

Task: determine the role or technique of an ingredient in a recipe.
```

```
Context: This classification feeds a lookup system. The RoleOrTechnique value will be used
to match this ingredient against a database of food alternatives. Choose the value that
most precisely describes how THIS ingredient functions in the recipe — not the dish
category, not surrounding ingredients. Precision matters: a wrong role will retrieve
irrelevant alternatives.

You are given:
- a list of allowed RoleOrTechnique values
- an ingredient
- the recipe instructions

You must choose the single best matching value from the list of allowed RoleOrTechnique
values.

Strict output rules:
- Output EXACTLY one value from the list of allowed RoleOrTechnique values.
- Output only the value.
- Do not output explanations.
- Do not output punctuation.
- Do not output quotes.
- Do not output multiple values.
- Do not invent new values.
- If no value clearly matches, output: unknown

Critical classification rule:
- Classify based on what THIS ingredient does in the recipe, not what is done around it.
- If an ingredient is cooked IN fat, it is not the fat itself.
- If an ingredient is added as a garnish at serving, it is a topping, not a protein or
sauce.

Here are a few examples:

# Example 1

## User message
Allowed RoleOrTechnique values:
- minced in sauce
- steak centerpiece
- cubes stew
- flavoring
- sandwich fill
- bread pizza
- pasta
- brine pack
- seasoning
- sauce addin
- chili burgers
- sauté fat
- finish oil
- cream swap
- staple
- bread
- baking fat
- topping
- swap in
- beverage
- stirfry protein
- veg boost
- broth base
- curry cubes
- sauce enricher
- roux binder
```

```
ingredient: butter

instructions:
- Melt butter in a pan and sauté the onions until soft.

Select the roleOrTechnique value.

Output only the value.

## Assistant message

sauté fat

# Example 2

## User message
Allowed RoleOrTechnique values:
- minced in sauce
- steak centerpiece
- cubes stew
- flavoring
- sandwich fill
- bread pizza
- pasta
- brine pack
- seasoning
- sauce addin
- chili burgers
- sauté fat
- finish oil
- cream swap
- staple
- bread
- baking fat
- topping
- swap in
- beverage
- stirfry protein
- veg boost
- broth base
- curry cubes
- sauce enricher
- roux binder

ingredient: olive oil

instructions:
- Drizzle olive oil over the pasta just before serving.

Select the roleOrTechnique value.

Output only the value.

## Assistant message

finish oil

# Example 3

## User message
Allowed RoleOrTechnique values:
- minced in sauce
```

```
- steak centerpiece
- cubes stew
- flavoring
- sandwich fill
- bread pizza
- pasta
- brine pack
- seasoning
- sauce addin
- chili burgers
- sauté fat
- finish oil
- cream swap
- staple
- bread
- baking fat
- topping
- swap in
- beverage
- stirfry protein
- veg boost
- broth base
- curry cubes
- sauce enricher
- roux binder

ingredient: onion

instructions:
- Add 2 tablespoons of olive oil, the onion and carrot. Sauté for 3-4 minutes.

Select the roleOrTechnique value.

Output only the value.

## Assistant message

flavoring

# Example 4

## User message
Allowed RoleOrTechnique values:
- minced in sauce
- steak centerpiece
- cubes stew
- flavoring
- sandwich fill
- bread pizza
- pasta
- brine pack
- seasoning
- sauce addin
- chili burgers
- sauté fat
- finish oil
- cream swap
- staple
- bread
- baking fat
- topping
- swap in
- beverage
```

```
- stirfry protein
- veg boost
- broth base
- curry cubes
- sauce enricher
- roux binder

ingredient: feta cheese

instructions:
- Serve with capers, some grated feta cheese, fresh oregano, freshly ground pepper.

Select the roleOrTechnique value.

Output only the value.

## Assistant message

topping
```

MATCH INGREDIENT TO TRIGGER – SYSTEM PROMPT

```
 You are a classification model.

Task: Classify the ingredient into ONE trigger ingredient value.

Context: This classification feeds a lookup system that retrieves predefined healthy
alternatives for this ingredient. The trigger ingredient value must reflect what this
ingredient genuinely IS — not a superficially related category. If the ingredient does not
closely match any value, output unknown. A wrong value retrieves irrelevant alternatives;
unknown is always safer than a forced match.

You will be given:
- the allowed trigger ingredient values
- an ingredient name
- the ingredient's role/technique in a recipe

You must choose the single best matching value from the list of allowed trigger ingredient
values.

Strict output rules:
- Output EXACTLY one value from the allowed list.
- Output only the value.
- Do not output explanations.
- Do not output punctuation or quotes.
- Do not output multiple values.
- Do not invent new values.
- If no value clearly matches, output: unknown
- If the ingredient is salt, olive oil or water output: unknown.

Here are a few examples:

# Example 1

## User message
Allowed trigger ingredient values:
- Beef
- Pork
- Bacon/lardons
- Luncheon meat
- White flour
- White pasta
```

```
- Canned tuna
- Soy sauce
- Any meat sauce
- Minced meat
- Butter
- General fat choice
- Low-dairy sauce
- White rice
- Refined bread
- Margarine (non-HO)
- Salad topping
- Protein choice
- SSB
- Low-dairy breakfast
- Roerbak proteïne
- Pasta dishes
- Stock cube
- Lamb
- Cream
- White couscous

ingredient: spaghetti

roleOrTechnique: pasta

Select the trigger ingredient value.

Output only the value.

## Assistant message

White pasta

# Example 2

## User message
Allowed trigger ingredient values:
- Beef
- Pork
- Bacon/lardons
- Luncheon meat
- White flour
- White pasta
- Canned tuna
- Soy sauce
- Any meat sauce
- Minced meat
- Butter
- General fat choice
- Low-dairy sauce
- White rice
- Refined bread
- Margarine (non-HO)
- Salad topping
- Protein choice
- SSB
- Low-dairy breakfast
- Roerbak proteïne
- Pasta dishes
- Stock cube
- Lamb
- Cream
- White couscous
```

```
ingredient: olive oil

roleOrTechnique: finish oil

Select the trigger ingredient value.

Output only the value.

## Assistant message

General fat choice

# Example 3

## User message
Allowed trigger ingredient values:
- Beef
- Pork
- Bacon/lardons
- Luncheon meat
- White flour
- White pasta
- Canned tuna
- Soy sauce
- Any meat sauce
- Minced meat
- Butter
- General fat choice
- Low-dairy sauce
- White rice
- Refined bread
- Margarine (non-HO)
- Salad topping
- Protein choice
- SSB
- Low-dairy breakfast
- Roerbak proteïne
- Pasta dishes
- Stock cube
- Lamb
- Cream
- White couscous

ingredient: beef mince

roleOrTechnique: minced in sauce

## Assistant message

Minced meat
```

DETERMINE INGREDIENT COMPOSITION – SYSTEM PROMPT

```
You are a constrained classifier.

Task: determine the composition of an ingredient that is part a recipe from a list of
components.

You are given:
```

```
- a list of allowed component names with an optional explanation in parentheses
- the name of the ingredient

You must choose all the components that apply to the ingredient.

Strict output rules:
- Output one component per line.
- Output only the component name, not the explanation.
- Do not output explanations.
- Do not output punctuation.
- Do not output quotes.
- Do not output multiple values in the same line.
- Do not invent new values.

Here are some examples:

# Example 1

## User message
Components:
- processed meat (any meat preserved, flavored, or modified via methods like salting,
curing, smoking, or adding chemical preservatives to extend shelf life; includes bacon,
sausage, hot dogs, ham, salami, and deli meats)
- red meat
- sodium
- sugar-sweetened beverages
- trans fatty acids
- calcium
- fiber
- fruits
- legumes
- milk
- nuts and seeds
- omega-6 polyunsaturated fatty acids (nutrients found in vegetable oils like soybean,
corn, sunflower, nuts, and seeds)
- seafood omega-3 fatty acids
- vegetables
- whole grains

Ingredient: Feta/ Evaporated milk

## Assistant message

sodium
calcium
milk

# Example 2

## User message
Components:
- processed meat (any meat preserved, flavored, or modified via methods like salting,
curing, smoking, or adding chemical preservatives to extend shelf life; includes bacon,
sausage, hot dogs, ham, salami, and deli meats)
- red meat
- sodium
- sugar-sweetened beverages
- trans fatty acids
- calcium
- fiber
- fruits
- legumes
- milk
```

```
- nuts and seeds
- omega-6 polyunsaturated fatty acids (nutrients found in vegetable oils like soybean,
corn, sunflower, nuts, and seeds)
- seafood omega-3 fatty acids
- vegetables
- whole grains

Ingredient: Spinach

## Assistant message

fiber
vegetables
```

DETERMINE BEST FITTING RECOMMENDATION – SYSTEM PROMPT

```
 You are a selection model.

Task: identify the single best fitting rule from a list of filtered database entries for a
given ingredient.

Context: This selection feeds a lookup system. The chosen rule id will be used to retrieve
predefined healthy alternatives for the ingredient. The entries have already been pre-
filtered by trigger ingredient — your task is to rank them by fit and return the id of the
best match. Precision matters: a wrong rule retrieves irrelevant alternatives.

You are given:
- the ingredient name as it appears in the recipe
- the ingredient's role or technique in the recipe
- the ingredient's dietary components
- a list of filtered database entries, each with an id, a recommendation and a role or
technique

You must choose the single best matching entry and output its id.

Selection criteria — apply in this order:
1. Role or technique match: prefer the entry whose role most closely matches the
ingredient's roleOrTechnique.
2. Dietary component relevance: if still tied, prefer the entry whose recommendation is
most relevant to the ingredient's dietaryComponents.

Strict output rules:
- Output EXACTLY one id from the list of filtered database entries.
- Output only the id value.
- Do not output explanations.
- Do not output punctuation.
- Do not output quotes.
- Do not output multiple values.
- Do not invent new values.
- If no entry clearly matches on any criterion, output the id of the first entry in the
list.

Here are a few examples:

# Example 1

## User message
ingredient: 4 slices of bacon
roleOrTechnique: sauté fat
```

```
triggerIngredient: bacon/lardons
dietaryComponents:
- processed meat
- sodium

Filtered db entries:
- id: 1
   - recommendation: Decrease processed meat
   - role: flavoring
- id: 2
   - recommendation:
  - role:

Select the id of the best fitting entry.
Output only the id.

## Assistant message

1

# Example 2

## User message
ingredient: butter
roleOrTechnique: sauté fat
triggerIngredient: Butter
dietaryComponents:
- omega-6 polyunsaturated fatty acids

Filtered db entries:
- id: 1
  - recommendation: Decrease saturated fat
  - role: baking fat
- id: 2
   - recommendation: Decrease saturated fat
   - role: sauté fat

Select the id of the best fitting entry.
Output only the id.

## Assistant message

2

# Example 3

## User message
ingredient: 2 baguettes
roleOrTechnique: bread pizza
triggerIngredient: refined bread
dietaryComponents:
-

Filtered db entries:
- id: 1
  - recommendation: Increase fiber
  - role: bread

Select the id of the best fitting entry.
Output only the id.

## Assistant message
```

SUGGEST ALTERNATIVES – SYSTEM PROMPT

```
You are a culinary nutrition assistant.

Task: given an ingredient that needs to be substituted in a recipe, select the best
alternatives and return them.

Context: You are the final step of a healthy eating recommendation pipeline. A curated
expert database has already been consulted and a set of candidate alternatives has been
retrieved. Your job is to evaluate these candidates and return the most suitable ones.
You are given:
- the ingredient name as it appears in the recipe
- the ingredient's role or technique in the recipe
- a list of candidate alternatives, each with a name, an optional explanation, and
optional restrictions
- optional equivalence notes (quantity / ratio guidance)
- optional technique notes (cooking method adaptations)

You must return the suitable alternatives, according to the restrictions and the role or
technique of the ingredient to be replaced.

Output rules:
- Output ONLY the name of each suitable alternative.
- Return between 1 and 3 alternatives. Prefer fewer, higher-confidence results over many
uncertain ones.
- Omit candidates that have a restriction that makes them clearly unsuitable given the
role or technique.
- Do not output null values — use an empty string "" if a field has no content.
- One alternative name per line.

Classification rules:
- The alternative is ALWAYS a name from the candidate list. Never invent a value here.

Here are a few examples:


# Example 1

## User message
We need to substitute the ingredient 4 10-12-inch flour tortillas
Its role in the recipe is -
The allowed substitutes are:
- Whole grain flour (blend 50%)
  - Restrictions: Hydration + proofing adjust
  - Equivalence: Start 30-50% blend
  - Technique notes: Increase hydration +5-10%
- Pulse flour blend (20-30%)
  - Restrictions: Texture changes
  - Equivalence: Start 30-50% blend
  - Technique notes: Increase hydration +5-10%
- Spelt (partial)
  - Restrictions: Gluten content differs
  - Equivalence: Start 30-50% blend
  - Technique notes: Increase hydration +5-10%
```

```
## Assistant message

Whole grain flour (blend 50%)
Spelt (partial)
```

Annex F ASSESSMENT LIST FOR TRUSTWORTHY AI

Authors: Rosaly Severijns (KU Leuven; Coordination Team), Nefeli Kousta (ICCS; AI system developer) & Nikos Paraskevopoulos (ICCS; AI system developer)

Date last version: 30/04/2026

This list is an ethical assessment of the digital tools developed in the EU Horizon project DietWise (Grant agreement ID: 101181692): MyRecipeWatch and the Responsible Cooking Alliance. As both rely on the same algorithm, only one assessment is completed for both. In MyRecipeWatch, users can enter recipe weblinks, and a rules-based AI system will give suggestions for ingredient replacements to foster healthy cooking practices. The Responsible Cooking Alliance is a web extension tool that provides ingredient replacement recommendations to influencers' recipes, based on the same algorithm. The AI system is rules-based and human-controlled and can only deliver output from a dataset curated by nutritional experts.

REQUIREMENT #1 Human Agency and Oversight

1.1 Human agency and autonomy	
Is the AI system designed to interact, guide or take decisions by human end-users that affect humans or society?	The AI tool is designed to provide healthy ingredient suggestions for human cooking decisions but cannot interact in a conversational sense or take decisions. Humans use it as a recommendation tool and maintain autonomy on whether to integrate the AI's suggestion in their recipes or not.
Could the AI system generate confusion for some or all end-users or subjects on whether a decision, content, advice or outcome is the result of an algorithmic decision?	The risk of confusion about whether suggestions are algorithmically generated is mitigated through onboarding materials and informed consent, which clearly explain that recommendations are produced by a rules-based AI system. The popup explanation feature reinforces this by providing transparent reasoning for each suggestion.
Are end-users or other subjects adequately made aware that a decision, content, advice or outcome is the result of an algorithmic decision?	Yes, the main concept of the tool is that an AI model extracts recipe text from a web-link and generates ingredient recommendations, based on a carefully curated database by dieticians. End-users are informed about and give consent to this goal upon downloading the tool.
Could the AI system generate confusion for some or all end-users or subjects on whether they are interacting with a human or AI system? Are end-users or subjects informed that they are interacting with an AI system?	The AI model cannot interact with users and has only one function: to give ingredient recommendations. End-users are informed of the basis of the recommendations upon downloading the tool.

Could the AI system affect human autonomy by generating over-reliance by end-users? Did you put in place procedures to avoid that end-users over-rely on the AI system?	No. The AI model only gives suggestions to improve recipes. Humans still decide autonomously to accept or reject the suggestion, and act in real life to implement the suggestion. The system cannot implement the suggestion itself. Relying on the system for suggestions would only mean that people eat more healthily.
Could the AI system affect human autonomy by interfering with the end-user's decision-making process in any other unintended and undesirable way? Did you put in place any procedure to avoid that the AI system inadvertently affects human autonomy?	The system could potentially affect user autonomy by nudging dietary choices towards a specific evidence-based framework, which may not align with all users' cultural preferences, personal values or individual health circumstances. However, this risk is mitigated by the accept/reject mechanism which preserves full user control over every suggestion, and by the system's transparent communication of the nutritional reasoning behind each recommendation. Users are never prevented from following their original recipe. Onboarding materials clearly frame the system as a supportive tool, and the voluntary nature of personalization inputs reinforces user agency throughout the interaction.
Does the AI system simulate social interaction with or between end-users or subjects?	No.
Does the AI system risk creating human attachment, stimulating addictive behaviour, or manipulating user behaviour? Depending on which risks are possible or likely, please answer the questions below: o Did you take measures to deal with possible negative consequences for end-users or subjects in case they develop a disproportionate attachment to the AI System? o Did you take measures to minimise the risk of addiction? o Did you take measures to mitigate the risk of manipulation?	The risk of human attachment or addictive behavior is low, because the tool is used as a supplement to an already existing task or hobby (cooking with recipes) and people cannot fully rely on the app to do this for them. Moreover, as people learn which ingredients can be replaced, they will remember this themselves and likely use the tool less over time rather than become dependent.
1.2 Human oversight	
Please determine whether the AI system (choose as many as appropriate): o Is a self-learning or autonomous system; o Is overseen by a Human-in-the-Loop; o Is overseen by a Human-on-the-Loop; o Is overseen by a Human-in-Command.	The system is not a self-learning or autonomous system, as it is rules-based and does not update itself without human intervention. All three human oversight models are applicable to varying degrees: Human-in-the-Loop: End-users exercise human-in-the-loop oversight through the accept/reject mechanism and intervening in every individual recommendation before it is applied to their recipe. They can also notify us of inappropriate suggestions through the easily accessible contact form. Inappropriate here means that the tool may confuse roles/ techniques of using ingredients, making suggestions unsuitable for the recipe in question. Due to the controlled nature of the system, it cannot

	suggest ingredients outside our expert-curated dataset. Human-on-the-Loop: The university development team and nutritionists oversee the system during its design cycle and monitor its operation post-launch through iterative reviews triggered by user feedback. Human-in-Command: The university development team retains full command over the overall activity of the system, including the ability to deactivate individual rules or shut down the system entirely.
Have the humans (human-in-the-loop, human-on-the-loop, human-in-command) been given specific training on how to exercise oversight?	Formal training on exercising oversight has not been organised. However, the nutritionists involved in iterative reviews have relevant domain expertise to assess the quality and appropriateness of recommendations. The computer science team has the technical expertise to monitor, maintain and intervene in the system's operation. An external ethical advisor brings specific expertise and feedback. Onboarding/introduction materials provide end-users with training/information on how to use the accept-reject mechanism and notify us of 'strange' recipe suggestions.
Did you establish any detection and response mechanisms for undesirable adverse effects of the AI system for the end-user or subject?	Before dissemination, the tool will be thoroughly pre-tested, including on usability and undesirable outputs of the AI model (wrong recipe extraction, unrealistic or inappropriate replacement suggestions). Moreover, after launching the tool, users can use the contact form to notify us about potentially inappropriate outputs, which the tool owners will then examine.
Did you ensure a 'stop button' or procedure to safely abort an operation when needed?	In the case of the tools in question, the contact form described above is sufficient. Even if the tool gives inappropriate outputs, this will not cause direct harm to users and hence a 'stop button' is not needed.
Did you take any specific oversight and control measures to reflect the self-learning or autonomous nature of the AI system?	This question is not applicable, as the system is not self-learning or autonomous. It is a rules-based system that does not update itself without explicit human intervention.

REQUIREMENT #2 Technical Robustness and Safety

Resilience to Attack and Security	
Could the AI system have adversarial, critical or damaging effects (e.g. to human or societal safety) in case of risks or threats such as design or technical faults, defects, outages, attacks, misuse, inappropriate or malicious use?	The system poses a low risk. Dataset errors or AI system behavior could lead to inappropriate swap suggestions, particularly for users with allergies or medical conditions, though user accept/reject control and expert nutritionist validation mitigate this. However, as the tools can only suggest ingredients from our carefully expert-curated dataset, there is no risk of proposing a non-edible, dangerous ingredient.

	The main misuse risk is users or influencers treating suggestions as clinical advice, mitigated by informed consent and the planned pre-testing and feedback system. Malicious URL submission could affect the parsing component but not directly corrupt recommendations. Dataset tampering would be the most serious security risk, making dataset integrity and access control critical safeguards. The risk of this is low as the dataset the tools run on is kept securely behind multi-factor authentication within the developing institution.
Is the AI system certified for cybersecurity (e.g. the certification scheme created by the Cybersecurity Act in Europe) or is it compliant with specific security standards?	The system has not yet obtained cybersecurity certification under the EU Cybersecurity Act or any equivalent scheme, as the system is currently in development. However, we comply with standard web application security practices, including secure data storage and input validation.
How exposed is the AI system to cyber-attacks? o Did you assess potential forms of attacks to which the AI system could be vulnerable? o Did you consider different types of vulnerabilities and potential entry points for attacks such as: Data poisoning (i.e. manipulation of training data); Model evasion (i.e. classifying the data according to the attacker's will); Model inversion (i.e. infer the model parameters)	The system's rules-based, non-learning architecture substantially limits cyber-attack exposure. Since there is no trainable model, model evasion and model inversion attacks are not applicable. Data poisoning is mitigated through strict access controls on the dataset the model runs on and the fact that dataset updates are managed by the university development team rather than being open to external input.
Did you put measures in place to ensure the integrity, robustness and overall security of the AI system against potential attacks over its lifecycle?	Yes, several measures are in place or planned. Dataset integrity is protected through restricted access, with updates managed exclusively by the university development team and validated by expert nutritionists. The URL parsing component is secured through input validation and sanitization. The code has been reviewed for security and vulnerability prior to launch. Post-launch, the system will be iteratively monitored and maintained by the university team, with nutritionist reviews triggered by user feedback. As the possible tool output is completely controlled by the expert dataset, the developers can easily adjust rules to allow rapid response to inappropriate suggestions without requiring full system shutdown.
Did you red-team/pentest the system?	No independent penetration test was performed.
Did you inform end-users of the duration of security coverage and updates? o What length is the expected timeframe within which you provide security updates for the AI system?	Planned for when we have fixed the exact security update schedule. Dedicated disclaimers will be provided within the application screens.
General Safety	
Did you define risks, risk metrics and risk levels of the AI system in each specific use case? o Did you put in place a process to continuously measure and assess risks? o Did you inform end-users and subjects of existing or potential risks?	Risks of the system include failure of the AI system to extract recipe text, and failure of the AI system to provide accurate ingredient suggestions (i.e., that fit the recipe/ match the trigger ingredient). We stress that since suggestions are provided from a curated white list, the system is not capable of producing harmful output.

Commented [RS48]: Ruling out URLs designed by hackers etc

Commented [RS49]: ICCS? I guess this asks if we will test risk at cyber attacks by trying to 'attack' the system ourselves

Commented [ne50R49]: exactly, but we won't do that. Needs independent team with the specific expertise.

Did you identify the possible threats to the AI system (design faults, technical faults, environmental threats) and the possible consequences? o Did you assess the risk of possible malicious use, misuse or inappropriate use of the AI system? o Did you define safety criticality levels (e.g. related to human integrity) of the possible consequences of faults or misuse of the AI system?	Threats are limited to the topics discussed above.
Did you assess the dependency of a critical AI system's decisions on its stable and reliable behaviour? (o Did you align the reliability/testing requirements to the appropriate levels of stability and reliability?)	The system is not classified as a critical AI system, as its output serves as advice, and final decisions remain with the user. Nevertheless, the stability and reliability of the system's recommendations are considered important given their potential impact on users' dietary choices. The rules-based and deterministic architecture ensures stable and consistent outputs that cannot go beyond the ingredients in our dataset. Pre-launch testing will validate the reliability of recommendations across diverse recipe types and user contexts. The contact form provides ongoing monitoring of reliability post-launch.
Did you plan fault tolerance via, e.g. a duplicated system or another parallel system (AI-based or 'conventional')?	No, there is no fault tolerance for the AI answers. The system is not critical enough to justify the additional cost and complexity, both for creating and maintaining it.
Did you develop a mechanism to evaluate when the AI system has been changed to merit a new review of its technical robustness and safety?	Since the system is completely human-controlled and does not update itself, changes to the system occur exclusively through deliberate decisions by the development team.
Accuracy	
Could a low level of accuracy of the AI system result in critical, adversarial or damaging consequences?	As the system relies on a human-controlled dataset, the risk of inaccurate output is low. It can only recommend ingredients from the dataset. Inaccuracy in this case would mean that the tools misunderstand the role/type of ingredient and therefore recommend ingredients that are 'inappropriate' for the recipe. The only harmful risk would be that the system recommends an ingredient the user is allergic to. The risk of damaging consequences from such inaccuracies is low, as humans control the final decisions they make in terms of cooking.
Did you put in place measures to ensure that the data (including training data) used to develop the AI system is up-to-date, of high quality, complete and representative of the environment the system will be deployed in?	Yes, the data the AI model runs on is carefully curated and iteratively checked by nutritional experts. As this is completely controlled by the tool developers, the system will not self-generate new training data automatically, ensuring high quality.
Did you put in place a series of steps to monitor, and document the AI system's accuracy?	The tool is developed and tested in different steps: • Development phase. The tool developers and project partners test the tool iteratively and note whether the system makes mistakes. • Pre-test phase. A limited number of end-users and dieticians are involved to test an almost-ready version

Commented [RS51]: @ICCS anything else you think of?
Commented [ne52R51]: No, the threats are the ones mentioned.
Commented [RS53]: ICCS? I guess this asks if the system falls back on an offline suggestion or something when something goes wrong? Not sure.
Commented [ne54R53]: We could add the fallback mechanism of manual recipe insertion ???
Commented [ni55R53]: No need for fault tolerance.

	of the tool and document its accuracy (i.e., appropriateness of suggestions for the recipe). • Pilot phase. After the usability tests and improvements, a pilot phase with additional evaluation of usability and accuracy is conducted among a larger group of end-users (~5,000 in three different countries). • After launching the tools, accuracy is monitored and adjusted through users' feedback.
Did you consider whether the AI system's operation can invalidate the data or assumptions it was trained on, and how this might lead to adversarial effects?	This is not applicable as the AI system cannot go beyond the output/text suggested in our expert-curated dataset.
Did you put processes in place to ensure that the level of accuracy of the AI system to be expected by end-users and/or subjects is properly communicated?	Yes. Informed consent during account registration informs people of potential accuracy risks and levels. In addition, the contact form system fosters a transparent environment in which end-users can give feedback.
Reliability, Fall-back plans and Reproducibility	
Could the AI system cause critical, adversarial, or damaging consequences (e.g. pertaining to human safety) in case of low reliability and/or reproducibility? (o Did you put in place a well-defined process to monitor if the AI system is meeting the intended goals? o Did you test whether specific contexts or conditions need to be taken into account to ensure reproducibility?)	The system's potential to cause critical or damaging consequences in case of low reliability is assumed to be low. Users always keep the autonomy over final decisions. The rules-based and deterministic architecture ensures reproducibility by design, with identical inputs consistently producing identical outputs. Pre-launch testing across diverse recipe types and in the three target countries verifies reproducibility across different contexts and conditions.
Did you put in place verification and validation methods and documentation (e.g. logging) to evaluate and ensure different aspects of the AI system's reliability and reproducibility? o Did you clearly document and operationalise processes for the testing and verification of the reliability and reproducibility of the AI system?	The system's rules-based and deterministic architecture ensures reproducibility by design. To ensure reliability, the following measures are in place or recommended: interaction data including submitted URLs, generated suggestions, and user accept/reject decisions are logged for research purposes, and this logging also serves as a system monitoring function. Testing protocols are formally documented prior to launch, covering functional performance across different recipe formats, websites and languages. Post-launch, logs are periodically reviewed by the development team to identify inconsistencies or failure patterns.
Did you define tested failsafe fallback plans to address AI system errors of whatever origin and put governance procedures in place to trigger them?	Yes, fallback measures are defined for the primary failure scenarios. In case of system outage or URL parsing failure, an error will be displayed to the user. In case of an identified inappropriate suggestion, rules can quickly be changed or added without requiring a full system shutdown, minimizing disruption while protecting user safety. The university development team holds authority to shut down the system if needed, triggered by user complaints received through the feedback system or by findings from nutritionist reviews.

Did you put in place a proper procedure for handling the cases where the AI system yields results with a low confidence score?	Since the system is rules-based and deterministic rather than probabilistic, confidence scores are not applicable.
Is your AI system using (online) continual learning? o Did you consider potential negative consequences from the AI system learning novel or unusual methods to score well on its objective function?	No.

REQUIREMENT #3 Privacy and Data Governance

Privacy	
Did you consider the impact of the AI system on the right to privacy, the right to physical, mental and/or moral integrity and the right to data protection?	Yes. The impact on privacy is mostly threatened by users making an account and providing information (email address). Moreover, the tools collect data on usage, and users can voluntarily fill in gender and age information for personalization of the recommendations. Personally identifiable information (Pii), i.e. only the email, is handled by Keycloak, a tool trusted generally by the world. The email is kept in a different DB than other information. Our system does not store IP addresses or locations, nor the recipes it assesses (potentially copyright content).
Depending on the use case, did you establish mechanisms that allow flagging issues related to privacy concerning the AI system?	End-users will not directly face privacy issues as they do not interact with the AI system in a conversational way. The use is limited to receiving recommendations. However, in the case a concern arises, they can use our contact form to alert the tool owners.
Data Governance	
Is your AI system being trained, or was it developed, by using or processing personal data (including special categories of personal data)?	No.
Did you put in place any of the following measures some of which are mandatory under the General Data Protection Regulation (GDPR), or a non-European equivalent? Data Protection Impact Assessment (DPIA); Designate a Data Protection Officer (DPO) and include them at an early state in the development, procurement or use phase of the AI system; Oversight mechanisms for data processing (including limiting access to qualified personnel, mechanisms for logging data access and making modifications); Measures to achieve privacy-by-design and default (e.g. encryption, pseudonymisation, aggregation, anonymisation); Data minimisation, in	As the data we collect and use via the tools is very minimal, we do not have a Data Protection Impact Assessment. However, data handling is carefully outlined in DietWise's Data Management Plan and the assigned Data Manager within the project oversees data activities. The Data Manager has been involved to set appropriate standards for gathering and storing the data under GDPR regulations. All data collected can be seen and controlled only by the tool owners. We only collect strictly necessary personal data (email addresses) to contact users and merge research data with tool system data. All else (i.e., gender and age) is voluntary and not essential for the tools to run. Background mechanisms make sure that the personal data is stored encrypted and separately from a separate pseudonymization key.

particular personal data (including special categories of data).	
Did you implement the right to withdraw consent, the right to object and the right to be forgotten into the development of the AI system?	The AI system itself does not store or train on data provided by the users. However, in the informed consent upon registering for the tool, the user is asked the proper permission for gathering the data. Also, users can request account and data deletion via the contact form.
Did you consider the privacy and data protection implications of data collected, generated or processed over the course of the AI system's life cycle?	Yes, privacy and data protection implications have been considered. The system collects and stores recipe-specific data (URL, title), ingredient swap acceptance/rejection interactions, app activity data, and voluntary age and gender inputs for research purposes. As the system is developed and deployed in the EU, it is subject to GDPR requirements. Key measures in place include: data collection is limited to what is necessary for research purposes; age and gender input is voluntary, reducing the amount of personal data collected; users are informed of data collection and its purpose through the informed consent procedure at registration. There is a clear data management plan devised by the DietWise project.
Did you consider the privacy and data protection implications of the AI system's non-personal training-data or other processed non-personal data?	The system's core dataset is a curated, non-personal nutritional dataset built and validated by expert nutritionists, based on GBD dietary risk factors. As this dataset contains no personal data, it does not carry direct privacy implications.
Did you align the AI system with relevant standards (e.g. ISO25, IEEE26) or widely adopted protocols for (daily) data management and governance?	The system aims to align with widely adopted standards for data management and governance relevant to its EU context. Key applicable standards include ISO/IEC 27001 for information security management, ensuring secure storage and access control of the curated dataset and user interaction data. GDPR compliance governs the collection, storage and processing of personal data including age, gender and interaction logs. ISO 8000 principles for data quality are relevant to ensuring the integrity and accuracy of the curated nutritional dataset.

REQUIREMENT #4 Transparency

Traceability	
Did you put in place measures that address the traceability of the AI system during its entire lifecycle?	Yes, traceability measures are in place or planned across the system lifecycle. Since the system is rules-based, every recommendation can be traced back to a specific rule and dataset entry and t**he team has access to all intermediate results of reasoning, thus can deterministically control how they are used**. The reasoning behind each suggestion is visible to users through the popup explanation feature, ensuring transparency at the point of interaction. Interaction data including submitted URLs, generated suggestions and accept/reject decisions are logged. Output quality will be improved post-launch due to the user contact form and periodical reviews by the university development team and nutritionists.
Explainability	

Did you explain the decision(s) of the AI system to the users?	Yes, the system provides explanations for each ingredient swap suggestion through a popup screen that users can open for each recommendation. This popup presents the reasoning behind the suggestion in plain language, for example explaining that a swap is recommended due to reduced salt content. Users are therefore informed of the rationale behind each recommendation before deciding to accept or reject it.
Do you continuously survey the users if they understand the decision(s) of the AI system?	A formal continuous survey mechanism for assessing user comprehension of the system's decisions is not currently in place. A simple comprehension assessment is incorporated into the pre-test and pilot phases prior to wider dissemination, verifying that the popup explanations are clear and understandable across different user groups.
Communication	
In cases of interactive AI systems (e.g., chatbots, robo-lawyers), do you communicate to users that they are interacting with an AI system instead of a human?	Not applicable.
Did you establish mechanisms to inform users about the purpose, criteria and limitations of the decision(s) generated by the AI system?	Yes, several mechanisms are in place to inform users about the purpose, criteria and limitations of the system. The informed consent procedure at registration communicates the system's purpose, its basis in GBD dietary risk factors, and relevant disclaimers. The popup explanation feature communicates the specific criteria behind each individual suggestion. The informed consent and onboarding materials explicitly address technical limitations and potential risks, including that recommendations are general in nature, not tailored to individual medical conditions or allergies, and do not substitute professional nutritional or medical advice. For influencers, additional guidance is provided on how to transparently communicate the tool's role to their audiences.

REQUIREMENT #5 Diversity, non-discrimination and fairness

Avoidance of unfair bias	
Did you establish a strategy or a set of procedures to avoid creating or reinforcing unfair bias in the AI system, both regarding the use of input data as well as for the algorithm design?	Yes, several measures are in place to avoid creating or reinforcing unfair bias. The curated dataset is built on GBD dietary risk factors, which are globally derived and reviewed by expert nutritionists, providing a scientifically grounded and broadly representative foundation. The system is pre-tested in the three target countries prior to launch to verify that recommendations are culturally appropriate and relevant in each local context. Swap suggestions focus on general ingredients that are broadly common across European cuisines, minimizing the risk of culturally inappropriate recommendations. Personalization, limited to age and gender, is grounded in scientific evidence and is a voluntary feature. The contact form allows users to report suggestions they find inappropriate or irrelevant, providing a continuous mechanism to identify and address potential bias in

	outputs post-launch. Dataset updates are validated by expert nutritionists before deployment, ensuring that any new data introduced does not reinforce existing biases or introduce new ones.
Did you consider diversity and representativeness of end-users and/or subjects in the data? (o Did you test for specific target groups or problematic use cases? o Did you research and use publicly available technical tools, that are state-of the-art, to improve your understanding of the data, model and performance? o Did you assess and put in place processes to test and monitor for potential biases during the entire lifecycle of the AI system (e.g. biases due to possible limitations stemming from the composition of the used data sets (lack of diversity, non-representativeness)? o Where relevant, did you consider diversity and representativeness of end-users and or subjects in the data?)	Yes, diversity and representativeness have been considered throughout the system design. The GBD dietary risk factors provide a broad evidence base, further validated by expert nutritionists. Pre-launch testing in Greece, Lithuania and Belgium specifically assesses whether recommendations are appropriate across different local dietary cultures. Personalization based on age and gender accounts for the most relevant demographic differences in nutritional needs but can also be switched off by the user if they wish. Known limitations regarding individual circumstances beyond age and gender are communicated to users through onboarding materials and informed consent.
Did you put in place educational and awareness initiatives to help AI designers and AI developers be more aware of the possible bias they can inject in designing and developing the AI system?	No specific initiatives were put in place, but the DietWise project generally has a large awareness of bias due to its focus on vulnerable populations. Moreover, the external expert board and ethics advisor ensure a mechanism for critical advice related to fairness and bias.
Did you ensure a mechanism that allows for the flagging of issues related to bias, discrimination or poor performance of the AI system? (o Did you establish clear steps and ways of communicating on how and to whom such issues can be raised? o Did you identify the subjects that could potentially be (in)directly affected by the AI system, in addition to the (end-)users and/or subjects?)	Yes, a contact form system is in place. Flagged issues are reviewed by the university development team and by nutritionists where relevant. Clear communication on how and to whom issues can be raised is provided through the onboarding materials.
Is your definition of fairness commonly used and implemented in any phase of the process of setting up the AI system? (o Did you consider other definitions of fairness before choosing this one? o Did you consult with the impacted	As the AI system only has one goal and cannot flexibly interact with users, we tailored the fairness definition accordingly. We defined fairness as *the equal quality and appropriateness of dietary swap recommendations across all user groups, regardless of age, gender, nationality or dietary culture*. This definition is operationalized through several mechanisms: the

communities about the correct definition of fairness, i.e. representatives of elderly persons or persons with disabilities? o Did you ensure a quantitative analysis or metrics to measure and test the applied definition of fairness? o Did you establish mechanisms to ensure fairness in your AI system?)	dataset covers a broad range of ingredients relevant across the three target countries, voluntary personalization accounts for age and gender differences in nutritional needs if desired, and pre-launch testing in Greece, Lithuania and Belgium assesses recommendation quality across different user contexts. Throughout the pilot studies, recommendation acceptance rates will be compared across demographic groups to identify any performance disparities. The contact system provides a continuous post-launch mechanism to monitor fairness in practice.
Accessbility and Universal Design	
Did you ensure that the AI system corresponds to the variety of preferences and abilities in society?	The system is designed to accommodate a variety of user preferences and abilities. The accept/reject mechanism respects individual dietary preferences and cultural practices. The popup explanation feature provides reasoning in plain language, making the system accessible to users without nutritional expertise. Age and gender personalization accounts for different nutritional needs across demographic groups. Pre-launch testing in the three target countries assesses accessibility and appropriateness across different user contexts. It is acknowledged that the system does not currently accommodate users with specific medical conditions or disabilities beyond age and gender, communicated upon registration.
Did you assess whether the AI system's user interface is usable by those with special needs or disabilities or those at risk of exclusion? (o Did you ensure that information about, and the AI system's user interface of, the AI system is accessible and usable also to users of assistive technologies (such as screen readers)? o Did you involve or consult with end-users or subjects in need for assistive technology during the planning and development phase of the AI system?)	The team previously organized co-creation sessions with vulnerable populations to assess the usability and requirements of such a tool. Moreover, during the pre-testing and pilot phase, specific attention will be paid to the usability of the interface. Pilot studies will purposely focus on groups at risk of vulnerability such as those with a low socio-economic status, migrants, adolescents and elderly.
Did you ensure that Universal Design principles are taken into account during every step of the planning and development process, if applicable?	Universal Design principles were not explicitly adopted as a formal framework throughout the design and development process. However, the principles that are relevant to digital products ('Equitable use', 'Flexibility and use', 'Simple and intuitive use', 'Perceptible information', and 'Tolerance for error') are taken into account in the fairness and ease of use/information as explained above.
Did you take the impact of the AI system on the potential end-users and/or subjects into account? (o Did you assess whether the team involved in	Yes, the impact on end-users has been considered throughout development. Pre-launch testing in the three target countries will involve engagement with the target users (citizens and influencers). Users with medical conditions, allergies or dietary

building the AI system engaged with the possible target end-users and/or subjects? o Did you assess whether there could be groups who might be disproportionately affected by the outcomes of the AI system? o Did you assess the risk of the possible unfairness of the system onto the end-user's or subject's communities?)	restrictions are identified as potential groups that can be impacted, which is mitigated through informed consent, clear communication of limitations, and the accept/reject mechanism. User feedback is used to monitor impacts post-launch.
Stakeholder participation	
Did you consider a mechanism to include the participation of the widest range of possible stakeholders in the AI system's design and development?	Yes, an important feature of the DietWise project is to engage stakeholders in the pre-tests and larger pilot tests of the tools. This includes (vulnerable) citizens, community/ welfare organizations, dietitians and schools. Moreover, the project partners include NGOs and institutions that influence the countries' health policies. Stakeholders are involved through co-creation sessions, pre-testing and pilot studies.

REQUIREMENT #6 Societal and Environmental Well-being

Environmental Well-being	
Are there potential negative impacts of the AI system on the environment?	The system's environmental impact is minimal. As a rules-based mobile app and web extension, it does not require computationally intensive model training or inference, resulting in low energy consumption.
Where possible, did you establish mechanisms to evaluate the environmental impact of the AI system's development, deployment and/or use (for example, the amount of energy used and carbon emissions)? (o Did you define measures to reduce the environmental impact of the AI system throughout its lifecycle?)	A formal mechanism to evaluate the environmental impact of the system has not been established, given the system's limited computational requirements and small scale.
Impact on Work and Skills	
Does the AI system impact human work and work arrangements?	The system's impact on human work is limited and largely positive. The primary users are home cooks and influencers, for whom the system serves as a supportive tool rather than a replacement for human decision-making. For influencers, the system may streamline the process of developing healthier recipe variations, augmenting rather than replacing their creative work. The system does not automate any professional nutritional advisory role, as recommendations are general in nature and explicitly not a substitute for professional advice. Upon successful evaluation, the tool may serve as a supplement to the work of dietitians, as a

	recommendation for dietary improvement, but in no way will the tools replace dietitians or medical professionals.
Did you pave the way for the introduction of the AI system in your organisation by informing and consulting with impacted workers and their representatives (trade unions, (European) work councils) in advance?	Not applicable.
Did you adopt measures to ensure that the impacts of the AI system on human work are well understood? (o Did you ensure that workers understand how the AI system operates, which capabilities it has and which it does not have?)	This question is largely not applicable to the current system, as it does not directly impact human work arrangements. As dietitians may recommend the tool to their clients as a supplement to their professional practice, clear communication materials about the system's capabilities and limitations are included in the tool information upon registration to support informed and appropriate use in this context.
Could the AI system create the risk of de-skilling of the workforce? (o Did you take measures to counteract de-skilling risks?)	The risk of de-skilling is not applicable. In cases where the system could be used in a professional context (e.g., recommended by dietitians), it will be a complement to nutritional advice rather than a replacement.
Does the system promote or require new (digital) skills? (o Did you provide training opportunities and materials for re- and up-skilling?)	The system is designed to be simple and intuitive, requiring no specific digital or nutritional skills to use. Onboarding information is provided upon registration. The project includes a specific focus on vulnerable groups, which will receive additional onboarding sessions and nutritional classes to support effectively navigating the system regardless of digital literacy levels.
Impact on Society at large or Democracy	
Could the AI system have a negative impact on society at large or democracy? (o Did you assess the societal impact of the AI system's use beyond the (end-)user and subject, such as potentially indirectly affected stakeholders or society at large? o Did you take action to minimize potential societal harm of the AI system? o Did you take measures that ensure that the AI system does not negatively impact democracy?)	The system's broader societal impact is assessed as largely positive, promoting healthier dietary choices grounded in evidence-based nutritional science. No significant negative impact on democracy or society at large is anticipated. The system does not collect or process data in ways that could influence political opinions or democratic processes, and its recommendations are strictly limited to the dietary domain. The contact form provides a continuous mechanism to monitor and address any unintended broader societal impacts post-launch.

REQUIREMENT #7 Accountability

Auditability	
Did you establish mechanisms that facilitate the AI system's auditability (e.g. traceability of the development process, the sourcing of training data	Yes, several mechanisms are in place. The rules-based architecture ensures that every recommendation can be traced back to a specific rule and dataset entry, making the decision-making process transparent and auditable. Interaction data

and the logging of the AI system's processes, outcomes, positive and negative impact)?	including submitted URLs, generated suggestions and accept/reject decisions are logged, providing a traceable record of system outputs and user responses. The contact form records user-reported issues, contributing to an ongoing log of system performance and impact. All pre-testing and piloting protocols and results are documented. Together these mechanisms ensure that the system's development process, data sourcing, operational logic and outcomes are traceable and available for external review if required.
Did you ensure that the AI system can be audited by independent third parties?	Yes, the same mechanisms as described above can be shared with third-party auditors if necessary.
Risk Management	
Did you foresee any kind of external guidance or third-party auditing processes to oversee ethical concerns and accountability measures? (o Does the involvement of these third parties go beyond the development phase?)	Yes, external guidance and oversight are in place throughout the project lifecycle. An independent external ethical advisor, and an External Expert Advisory Board is involved for the duration of the project, providing ongoing ethical oversight beyond the development phase. The board includes computer scientists and app developers. Additionally, the project is guided by EU regulations and monitored by EU project officers, providing a further layer of independent accountability.
Did you organise risk training and, if so, does this also inform about the potential legal framework applicable to the AI system?	No formal risk training program has been organized for the development team. However, the team is generally aware of the potential risks associated with the system and the applicable legal framework, including GDPR and the EU AI Act, through their academic and professional backgrounds. The involvement of an external ethical advisor and EU project officers provides additional guidance on legal and regulatory requirements throughout the project lifecycle.
Did you consider establishing an AI ethics review board or a similar mechanism to discuss the overall accountability and ethics practices, including potential unclear grey areas?	There is no specifically dedicated AI ethics review board, but the external ethics advisor, and the external advisory board which includes AI experts ensures discussions on accountability and ethics practices take place. The external experts will also receive and review this ALTAI assessment.
Did you establish a process to discuss and continuously monitor and assess the AI system's adherence to this Assessment List for Trustworthy AI (ALTAI)?	The ALTAI will be checked and revised every 6 months throughout the duration of the DietWise project.
Did you establish a process for third parties (e.g. suppliers, end-users, subjects, distributors/vendors or workers) to report potential vulnerabilities, risks or biases in the AI system? (o Does this process foster revision of the risk management process?)	Yes, several mechanisms are in place for third parties to report potential vulnerabilities, risks or biases. End-users can report concerns through the contact form. Reported issues are reviewed by the university development team and nutritionists where relevant, directly informing dataset updates. The external ethical advisor and EU project officers provide an additional channel through which broader risks and ethical concerns can be raised and addressed.
For applications that can adversely affect individuals, have redress by design mechanisms been put in place?	Even though the risk is low, redress mechanisms are built into the system design. Users can reject any individual swap suggestion through the accept/reject mechanism, ensuring immediate

	recourse at the point of interaction. For broader concerns, the contact system provides a direct channel to report inappropriate suggestions to the development team. The team can deactivate specific rules or suggestions without requiring a full system shutdown.