System Documentation

Durchschlag Hannah (K12317156) Enderle Moritz (K12428331)

June 23, 2025

Contents

1 Goal of the System

To support users with food intolerances in assessing whether particular foods align with their unique dietary needs by offering a rating system that ranges from fully suitable for consumption to strongly advised against eating.

- Non-AI Goals:
  - Customizable User Experience: Create profiles detailing food intolerances (e.g., lactose, gluten).
  - Food Input Analysis: Request information about a specific ingredient or even whole recipes.
  - History Tracking: See past requests and instantly revisit the report.
- AI Goals:
  - Ingredient Analysis: Use the user's profile to generate a custom report tailored exactly to the user's preferences.
  - Dish Analysis: Use a common recipe for a requested dish and determine which ingredients should be avoided.

2 Requirements

- Functional Requirements:
  1. EatSafeAI shall require the user to create an account or log into an existing account using Google authentication.
  2. When the user first logs into the app, EatSafeAI shall ask them to enter their intolerances, including personal notes.
  3. If the user has created a profile before and logs back into their account, EatSafeAI shall load the user preferences and not require the user to re-enter their intolerances.
  4. EatSafeAI shall allow the user to search for ingredients or dishes.
  5. When a user requests information about a specific ingredient, EatSafeAI shall provide a compatibility assessment based on the user's food intolerances.
  6. When a user inquires about a specific dish, EatSafeAI shall compile a standard recipe, assess each ingredient's compatibility with the user's food intolerances, and provide an overall evaluation.
  7. EatSafeAI shall provide a list of past requests of the user.
  8. If the user is already registered and opens the app, EatSafeAI shall provide compatible suggestions for recipes that change frequently.
  9. EatSafeAI shall have a Settings page where the user can edit their user profile, including their intolerances, log out, or delete their account.
- Non-Functional Requirements:
  1. EatSafeAI shall ensure data privacy and security for user profiles and history. Type: Attributes
  2. EatSafeAI shall provide responses to user requests within no more than 30 seconds. Type: Performance
  3. EatSafeAI shall be accessible via mobile interfaces. Type: External Interfaces

4. EatSafeAI shall comply with relevant health and data protection regulations. Type: Constraints

- Functional AI-Related Requirements:
  Description: These requirements involve AI for evaluating ingredients (Req5) and dishes (Req6).
  Thresholds:
  - Responses should be delivered within 30 seconds to maintain a seamless user experience and keep users engaged.

  Implementation Plan:
  - Implement a zero-shot agent system powered by a Large Language Model, accessed through the provided API services.
  - Deploy the agent via a FastAPI server running on a centralized server for accessibility.
- Non-Functional AI-Related Requirements:
  1. Explainability (Req5, Req6) Category: Explainability, Transparency and Trust Description: Provide a rationale for assessments (e.g., "Contains [X], a source of [Y], to which you are intolerant").
  2. Provide Safety Hints (Req5, Req6) Category: Safety, Ethics Description: Include disclaimers to caution the user that the information provided may be inaccurate and should always be verified manually.

3 Use Case Descriptions

3.1 Primary Use Cases

UC1: Create User Profile

- **Description:** A user creates a profile by authenticating with Google and entering their food intolerances.
- **Actors:** User with intolerances
- **Stakeholders:** User, Developer/Publisher
- **Pre-conditions:** User has downloaded EatSafeAI and has a Google account.
- **Success condition:** User has a profile in the app.
- **Failure condition:** User cannot sign up and is forced to delete the app.

Main Success Scenario: User opens EatSafeAI → System prompts Google sign-in → User logs in → System displays intolerances screen → User enters intolerances → System saves the profile and redirects to the dashboard. *Exceptions:* Google authentication fails; User exits during login.

UC2: Identify Food Compatibility

- **Description:** The AI compiles a recipe and evaluates ingredient or dish compatibility.
- **Actors:** User with intolerances, AI Agent
- **Stakeholders:** User, Nutritionists/Dietitians

- **Pre-conditions:** User is logged in with a created profile.
- **Success condition:** Successful evaluation and display of the response.
- **Failure condition:** No internet connection.

Main Success Scenario: User enters a query in the search bar → System analyzes ingredient compatibility → System displays "Safe to eat" with a summary (e.g., "No lactose detected"). *Alternative:* User enters a dish (e.g., "butter chicken") → System analyzes dish compatibility → System displays an evaluation (e.g., "Not safe due to dairy"). *Exceptions:* No internet connection displays a "No connection" error.

UC3: View Search History

- **Description:** The user views a list of past food compatibility requests.
- **Actors:** User with intolerances
- **Stakeholders:** User
- **Pre-conditions:** User is logged in and has made prior searches.
- **Success condition:** User views the result of a past request.

Main Success Scenario: User navigates to "History" → System displays search history → User clicks on a past search and instantly accesses the result. *Alternative:* If history is empty, the system displays a hint that no prior searches exist. *Exceptions:* Data retrieval fails; the system tells the user to try again later.

UC4: View Recipe Suggestions

- **Description:** The system suggests compatible recipes based on the user's intolerances.
- **Actors:** User with intolerances
- **Stakeholders:** User, Nutritionists/Dietitians
- **Pre-conditions:** User is logged in with a profile.
- **Success condition:** The user can view a recipe recommended to them.
- **Failure condition:** No recipes can be shown.

Main Success Scenario: User opens the app → System recommends recipes → User clicks on a recipe → The recipe opens in the browser. *Alternative:* User wants new recipes → System regenerates recipes. *Exceptions:* No compatible recipes are found; suggestions disappear from the dashboard.

UC5: Edit Profile

- **Description:** The user edits their user profile or deletes their account.
- **Actors:** User with intolerances
- **Stakeholders:** User, Developer/Publisher
- **Pre-conditions:** User is logged in with an existing profile.
- **Success condition:** The system successfully saves updates or deletes the user's account.
- **Failure condition:** The save fails.

Main Success Scenario: User selects "Edit Profile" → System displays the profile page → User adds an intolerance (e.g., "histamine intolerance") and saves → System confirms the update. *Alternative:* User selects "Delete Account" → System prompts for confirmation → User confirms, and the account is deleted. *Exceptions:* The system displays an "Update failed" error; the user retries.

3.2 Supporting Use Cases

SUC1: Analyze Ingredient Compatibility

- **Description:** The system analyzes individual ingredient safety.
- **Actors:** System (AI Agent)
- **Pre-conditions:** The user has a profile.
- **Success condition:** The system displays an ingredient analysis.
- **Failure condition:** The system cannot display the result.

Flow: System queries API → API builds a prompt for the agent → The agent returns a recommendation → System displays the evaluation.

SUC2: Analyze Dish Compatibility

- **Description:** The AI compiles a recipe and evaluates dish safety.
- **Actors:** System (AI Agent)
- **Pre-conditions:** The user has a profile with intolerances.
- **Success condition:** The system displays a dish evaluation.
- **Failure condition:** Ingredient data is unavailable.

Example Flow: System receives "butter chicken" → AI compiles a recipe (chicken, butter, cream) → AI checks intolerances → Returns "Not safe due to dairy."

SUC3: Retrieve User History

- **Description:** The system retrieves past search history.
- **Actors:** System
- **Pre-conditions:** The user has a profile and prior searches.
- **Success condition:** The system displays the search history.
- **Failure condition:** The system cannot display past searches.

Flow: The system queries the database for past searches → Returns a list of all past searches.

SUC4: Generate Recipe Suggestions

- **Description:** The system generates personalized recipe recommendations.
- **Actors:** System (AI Agent)
- **Pre-conditions:** The user has a profile.
- **Success condition:** The system displays recipe recommendations.
- **Failure condition:** The system cannot show recommendations.

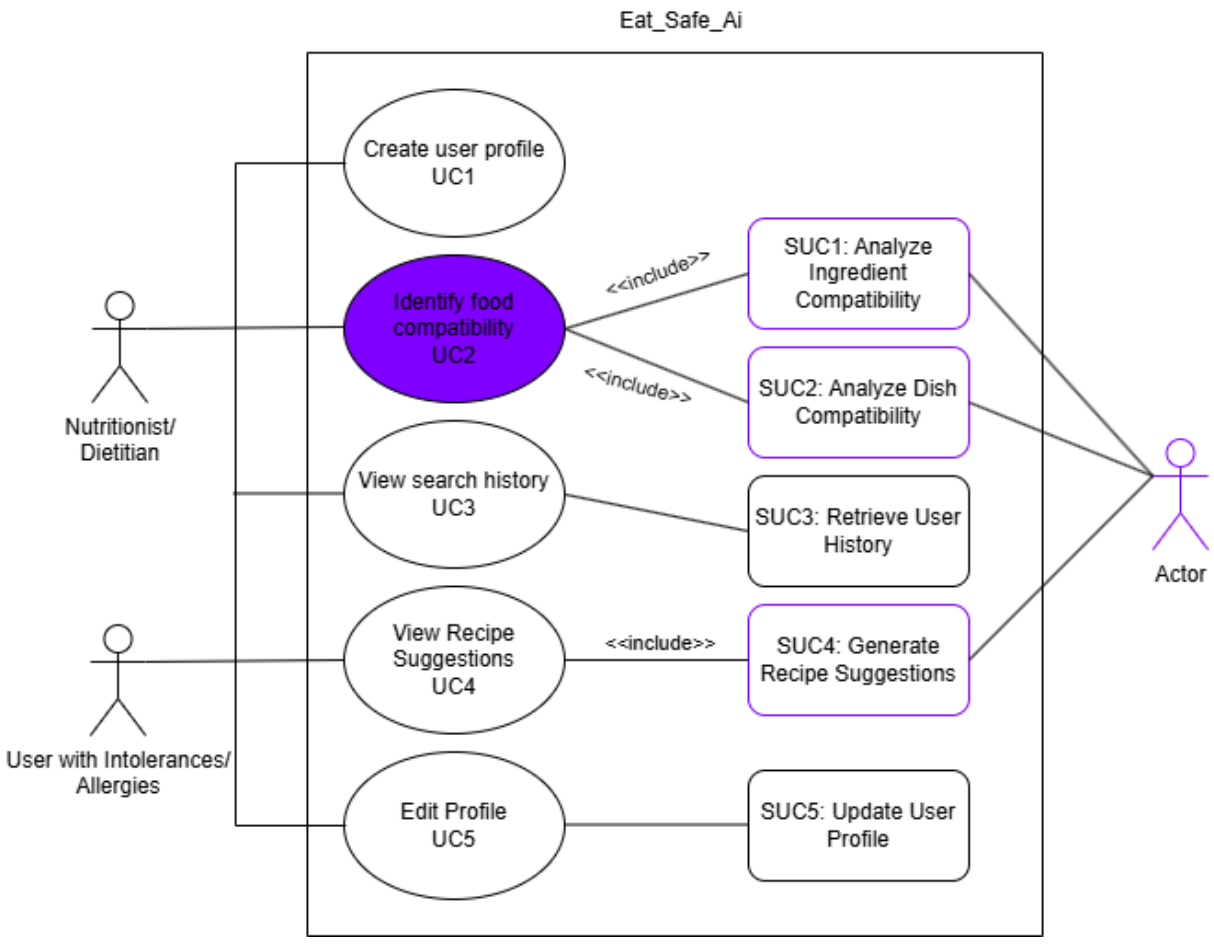
Flow: The app queries the API for recommendations → The API builds a user-specific prompt → The AI agent returns 5 recipes → The system displays the results. *Exceptions:* AI response timeout → The system displays previous recommendations and retries later.

SUC5: Update User Profile

- **Description:** The system updates or deletes profile data.
- **Actors:** System
- **Pre-conditions:** The user has a profile.
- **Success condition:** The profile is successfully updated.
- **Failure condition:** The profile could not be updated.

Flow: The user submits updated intolerances → The system saves the changes. *Alternative:* Account deletion → The system deletes the profile. *Exceptions:* Update fails → Returns an error.

4 Use Case Diagram



5 Implemented Use Cases

In the prototype of EatSafeAI, we have fully implemented the following primary and supporting use cases:

5.1 Primary Use Cases

UC1: Create User Profile

- Users can sign in via their Google Account and immediately enter their personal intolerances.
- The system persists this data so that returning users never need to re-enter their dietary restrictions.

UC2: Identify Food Compatibility

- Users may search for any ingredient or dish.
- For an *ingredient*, EatSafeAI returns a clear compatibility score along with a concise explanation.
- For a *dish*, the app compiles a standard recipe, evaluates each component against the user's profile, and delivers an overall verdict.

UC3: View Search History

- All past compatibility checks are logged in chronological order.
- Users can revisit any previous result instantly by tapping on the history entry.

UC5: Edit Profile

- From Settings, users can update their list of intolerances at any time.
- A "Delete Account" option allows for the permanent removal of all personal data.

5.2 Supporting Use Cases

SUC1: Analyze Ingredient Compatibility

- Under the hood, the AI Agent formulates a query to the LLM, retrieves relevant nutritional data, and returns a targeted safety assessment for a single ingredient.

SUC2: Analyze Dish Compatibility

- When a dish name is entered, the system first reconstructs a canonical recipe, then sequentially checks each ingredient against the user profile, and finally synthesizes an aggregated recommendation.

SUC3: Retrieve User History

- The backend history service fetches and delivers the user's past searches in a single, efficient database query.

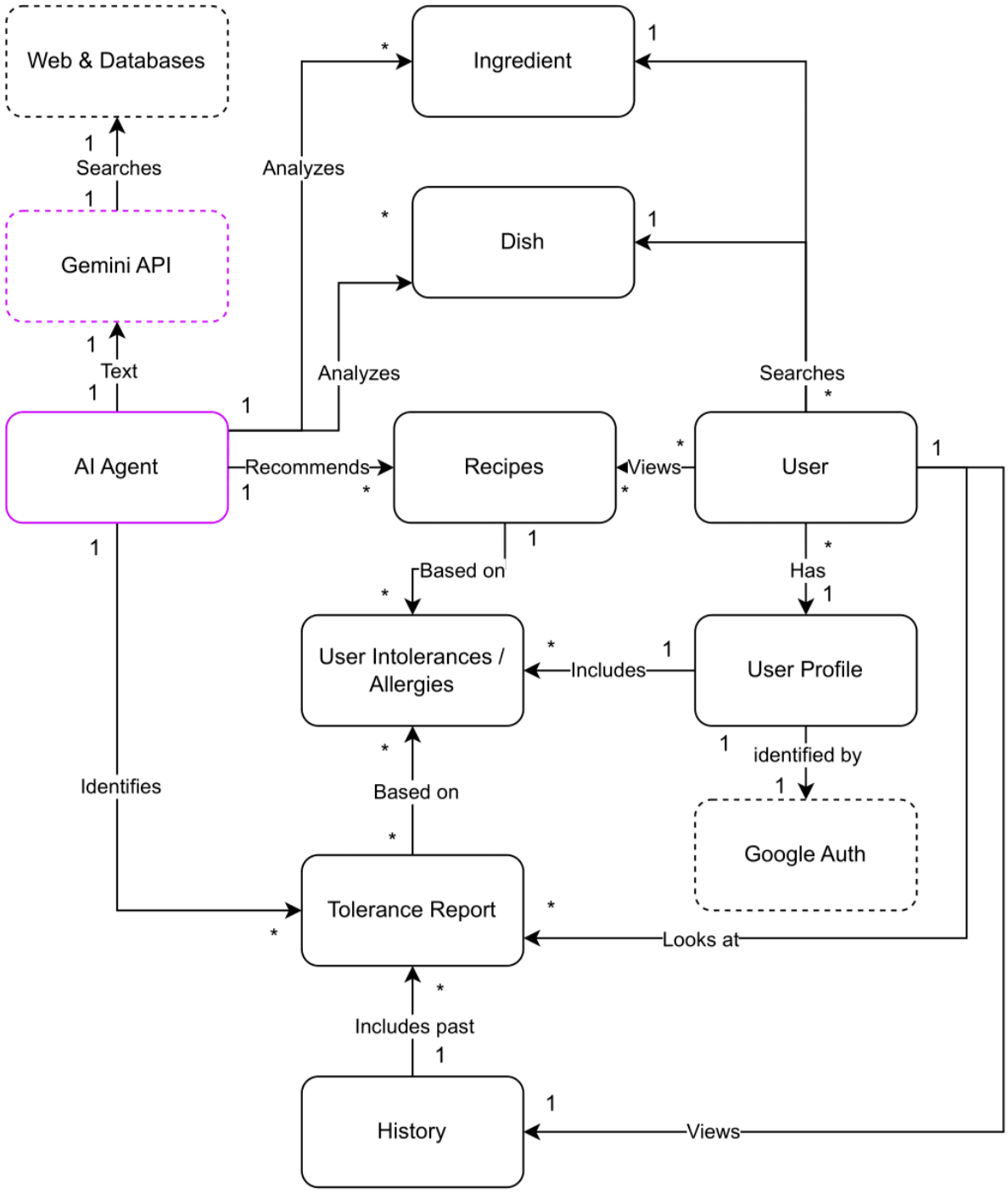
SUC5: Update User Profile

- On profile edits, the system validates and saves new intolerance entries and immediately reflects these updates across all future compatibility checks.

6 Traceability Matrix

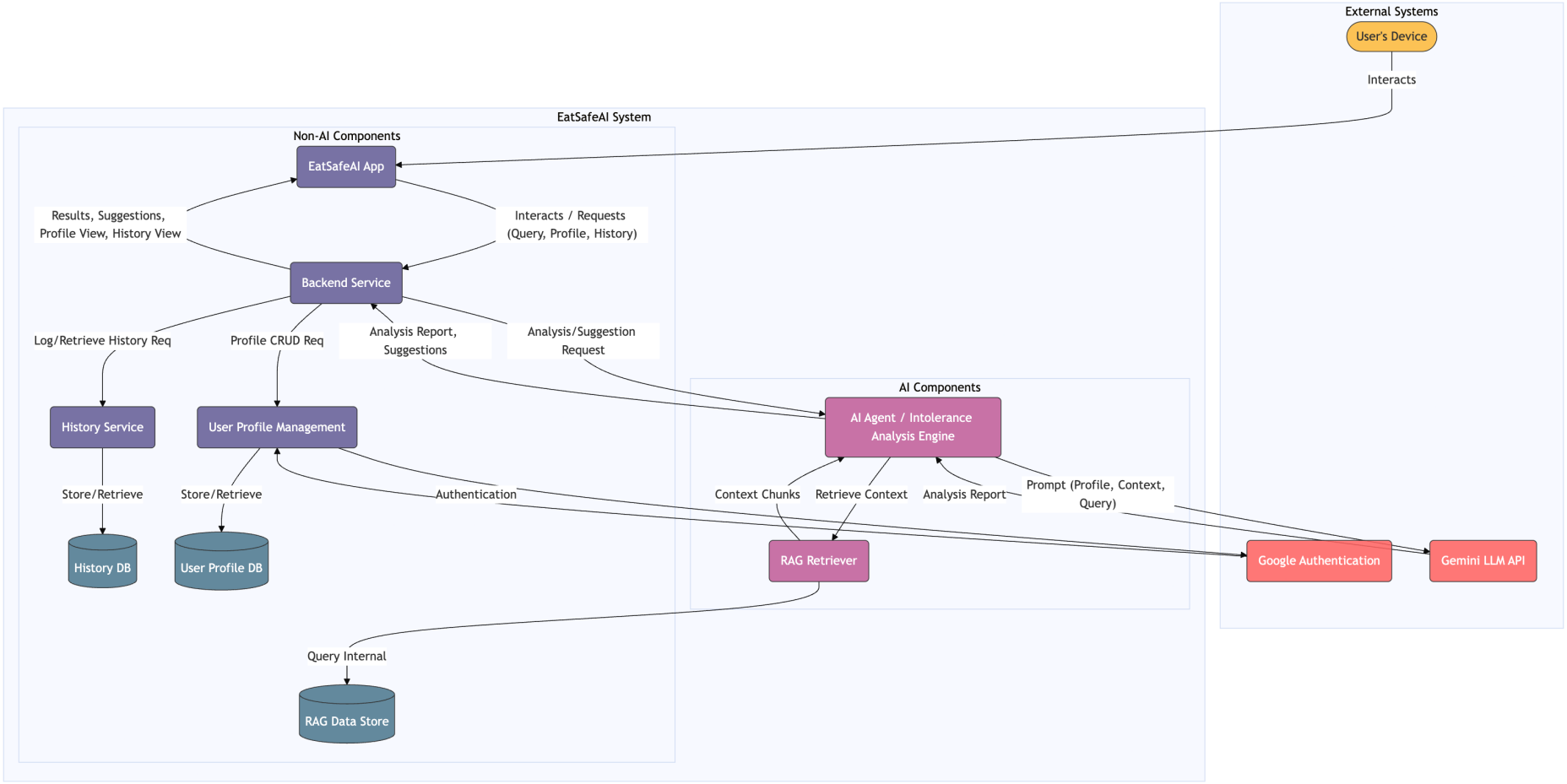
	UC1	UC2	UC3	UC4	UC5	SUC1	SUC2	SUC3	SUC4	SUC5
Req1	✓	-	-	-	-	-	-	-	-	-
Req2	✓	-	-	-	-	-	-	-	-	-
Req3	-	-	-	-	-	-	-	-	-	-
Req4	-	✓	-	-	-	-	-	-	-	-
Req5	-	✓	-	-	-	-	-	-	-	-
Req6	-	✓	-	-	-	-	-	-	-	-
Req7	-	-	-	✓	-	-	-	-	-	-
Req8	-	-	-	-	✓	-	-	-	-	-
Req9	-	-	-	-	-	-	-	-	-	-
NfReq1	✓	-	-	✓	-	-	-	✓	✓	✓
NfReq2	-	✓	-	✓	-	-	✓	✓	✓	✓
NfReq3	✓	✓	-	✓	-	-	✓	✓	✓	✓
NfReq4	-	✓	-	-	-	-	✓	✓	✓	-
AIReq1	-	-	-	-	-	✓	✓	✓	✓	-
AINfReq1	-	✓	-	✓	-	✓	✓	✓	✓	-
AINfReq2	-	✓	-	✓	-	✓	✓	✓	✓	-

7 Domain Model



- - - external resources
—— internal component

8 Architecture Diagram



9 Components Description

Component	Description	Related Requirements
User's Device	Represents the user's interface for interacting with the EatSafeAI system, allowing them to log in, input queries, view their profile, access history, and manage settings.	Req1, Req2, Req3, Req4, Req7, Req9, NfReq3
Google Authentication	External system securely handling user login and account creation.	Req1
Gemini LLM API	External Large Language Model API powering the AI Agent for ingredient and dish evaluation and recipe compilation.	Req5, Req6, AIReq1
EatSafeAI App (UI)	Front-end application providing the user interface for EatSafeAI features, including profile management, search, result display, history viewing, and settings.	Req2, Req3, Req4, Req5, Req6, Req7, Req8, Req9, NfReq3
Backend Service	Central component managing user sessions, handling requests from the UI, orchestrating interactions between services, and ensuring data flow within the system.	Req4, Req5, Req6, Req7, Req8, Req9
User Profile Management	Manages user accounts and profiles, including storing and retrieving intolerance information and personal notes.	Req2, Req3, Req9, NfReq1
History Service	Logs and retrieves user search history for convenience.	Req7
History DB	Database persistently storing user search history.	Req7, NfReq1
User Profile DB	Database securely storing user profile information, including intolerances and personal notes.	Req2, Req3, Req9, NfReq1
RAG Data Store	Holds the information used by the RAG Retriever to provide context for the AI Agent's analysis.	AIReq1
AI Agent / Intolerance Analysis Engine	Core AI component evaluating ingredient and dish compatibility based on user profiles and retrieved context, providing analysis reports and suggestions.	Req5, Req6, Req8, AIReq1, NfAIReq1, NfAIReq2
RAG Retriever	Fetches relevant information from the RAG Data Store to provide the AI Agent with the necessary context for analysis.	AIReq1

10 Design Questions and Answers

1. How important is a correct assessment, and what degree of mistakes is tolerable?
   A correct assessment is crucial; even small mistakes can be dangerous when dealing with food intolerances. Since the evaluation is AI-based, a disclaimer will be included to inform users.
2. Will the app work on different kinds of phones?
   Currently, we are only developing this app for Android. An iOS adaptation can be added later.
3. What logging and monitoring strategies will be implemented across the system to facilitate debugging and identify potential issues?
   There will be a simple logging system on the AI Agent backend. The user app only offers local debugging and will not collect data.
4. How will user feedback on the accuracy of AI-generated assessments be collected and used to potentially improve the system over time?
   The user can rate the assessments using a 5-star rating system and report a problem if they find the answer incorrect. This data can then be used in the future to improve the results.
5. How will the Settings page (Req9) be implemented to ensure the secure and reliable modification and deletion of user data?
   Deleting an account requires confirmation and permanently removes all associated data, with all actions being logged. Changes to the user profile are made by restarting the setup process. Information about email, name, etc., is retrieved from Google authentication.
6. What data sources can be used for the assessment?
   The main data source will be public databases containing nutritional facts about a wide variety of foods. A key example is the publicly available Australian Food Standards database.
7. What authentication methods can be offered?
   The implementation of the app allows for many authentication methods, but as the setup can be very complex, the first goal is to offer only Google authentication. More methods can be easily added later.
8. How can the system react when no information is available about a specific request?
   The system will then search for reliable sources on the internet to find the needed information.
9. How will the system handle regional variations or different naming conventions for the same food item across various data sources?
   Using the LLM (Gemini) automatically resolves this issue, as it can handle misspellings and regional differences effectively. The RAG retrieval and the answer are thus not affected.
10. Can we handle misspellings?
    As above, using the LLM (Gemini) automatically resolves this issue, as it can handle misspellings and regional differences effectively. The RAG retrieval and the answer are thus not affected.