Requirements Specification
(Functional Analysis)

Group 19

BiteBudget

1. INTRODUCTION

1.1. Objective and motivation

This document contains a detailed description of the system's needs, setting out what it will do and how it is expected to do so.
Use cases, functional requirements, and non-functional requirements together complete the system description for the "BiteBudget" application.

The primary objective of this system is to assist users in creating personalized weekly meal plans based on their budget, dietary restrictions (e.g., type of diet, calorie intake, protein goals), and preferred supermarket. The application will also generate a corresponding shopping list to streamline grocery shopping.

The motivation behind this project stems from the increasing demand among students and young people for tools that simplify meal planning while addressing specific dietary and financial constraints. By automating this process, the system aims to save users time, reduce food waste, and promote healthy eating habits tailored to individual needs.

This document outlines the functional and non-functional requirements of the system, detailing its expected behavior and performance. It serves as a foundational reference for all stakeholders involved in the development process.

1.2. Audience

This document is aimed at all those involved including:

- Development Team: To understand the technical and functional requirements necessary for building the application.
- Project Managers: To oversee progress and ensure alignment with project goals.
- Stakeholders: To verify that the system meets user needs and business objectives.
- Testers: To design test cases ensuring all requirements are met.
- End Users: To ensure their needs are accurately captured in the system design.

1.3. Definitions

To avoid ambiguity, this section includes definitions of key terms used throughout the document:

- Meal Plan: A structured schedule of meals (breakfast, lunch, dinner, snack) for a week.
- Dietary Restrictions: Specific dietary preferences or limitations, such as vegetarianism, veganism, or high-protein diets.
- Nutritional Profile: Compilation of user’s dietary restrictions, caloric intake, allergies...
- Shopping List: A compiled list of ingredients required to prepare meals in the weekly plan.
- Budget: The maximum amount a user is willing to spend on groceries for one week.
- Supermarket Preference: The user’s preferred store(s) for purchasing groceries.

- Food Inventory: User’s available ingredients in their pantry, including expiration dates.
- Meal Plan Generator: Component of the system responsible for creating weekly meal plans based on the user's nutritional profile, dietary restrictions, budget, and supermarket preferences.
- Recipe Filtering: process that filters available recipes according to the user's dietary restrictions, cooking time, budget, and ingredient preferences to ensure the meal plan matches their needs.

1.4. Related documents

The following documents are related to this requirements specification:

- "Vision and Needs Document" (if available): Outlining high-level goals and user needs.

2. SYSTEM OVERVIEW

2.1. Vision

BiteBudget is a mobile application designed to help young people maintain a healthy diet tailored to their nutritional preferences, lifestyle, and financial budget. The system aims to eliminate common barriers in daily meal planning: limited time, lack of ideas, dietary restrictions, and budget control.

2.2. Scope

The system allows users to define their nutritional profile, including diet type (vegan, vegetarian, keto, etc.), caloric needs, allergies or intolerances, weekly food budget, preferred shopping locations (favorite supermarket, local stores), and available cooking time. Based on this information, BiteBudget generates a personalized daily meal plan and a fully optimized shopping list.

The app provides an end-to-end experience—from meal planning to shopping—fully tailored to each user's lifestyle.

2.3. Context Diagram

2.3.1. Primary Actor:

- **User**: A young person using the app to plan daily or weekly meals according to their lifestyle, dietary needs, and budget.

2.3.2. Secondary Actors:

- **Admin**: For maintaining and updating the databases of products and recipes.
- **Recipe and Nutrition APIs**: External services used to retrieve recipe and nutritional data (e.g., Spoonacular, Edamam). (Future improvement)

2.4. Functional Modules

2.4.1. User Profile

- User registration and login
- Nutrition profile configuration (diet type, calorie intake, allergies, etc.)
- Weekly food budget management
- Preferred store selection and daily cooking time

2.4.2. Meal Planning Engine

- Generation of daily/weekly personalized recipes
- Smart filtering (quick meals, budget-friendly, gluten-free, etc.)
- Nutritional evaluation of the meal plan (total calories, macros, etc.)

2.4.3. Shopping List Generator

- Automatic conversion of selected recipes into a shopping list
- Grouping by categories (vegetables, proteins, etc.)
- Optimization by store or budget constraints

2.4.4. Recipe Browser & Selector

- Display of suggested recipes
- Option to mark favorites, substitute ingredients, or exclude items
- Recipe details (ingredients, steps, preparation time, nutrition info)

2.4.5. Budget Tracker (future enhancement)

- Tracking and recording grocery expenses
- Alerts when exceeding the budget
- Suggestions for saving money

2.4.6. Notifications & Reminders

- Reminders to shop
- Alerts for pending recipes or ingredients nearing expiration

3. SOLUTION USERS

Type or group	Description	Call of Duty
User	A person who interacts with the app to generate customized meal plans based on personal data such as budget, diet, calorie intake, and supermarket preference.	- Enter preferences and budget - Receive recommended meal plans - Receive the shopping list
Administrator	A person responsible for managing the app's backend, including recipes, shopping list, and technical issues.	-Manage recipe and product database - Handle user support requests - Monitor system performance and analytics
Project Manager	Oversees the development process and ensures the solution meets business objectives.	- Review requirements and progress - Coordinate between stakeholders and development team - Approve releases and updates
Tester	Quality assurance personnel responsible for validating the system's functionality and performance.	- Design and execute test cases - Report bugs and issues - Verify requirements are met
Stakeholder	Individuals or groups with a vested interest in the success of the application (e.g., investors, university representatives).	- Provide feedback on requirements and features - Review progress and outcomes - Approve major milestones

4. FUNCTIONAL REQUIREMENTS

This section describes the main functionalities that the application must fulfill. The requirements are divided into a general catalogue, specific security-related functions, and logical groupings based on the app's features.

4.1. CATALOGUE OF FUNCTIONAL REQUIREMENTS

This section defines the functional requirements of the system, structured into grouped functionalities and focused on meeting user needs.

Functional requirement	Version	Priority
RF-001: User registration and login	1.0	High
RF-002: Set diet type and restrictions	1.0	High
RF-003: Set nutritional goals	1.0	High
RF-004: Set weekly food budget	1.0	High
RF-005: Select preferred supermarket	1.0	Med
RF-006: Select preferred cooking time	1.0	Med
RF-007: Generate a weekly meal plan	1.0	High
RF-008: Smart recipe filtering	1.0	Med
RF-009: Nutritional summary of plan	1.0	Med
RF-010: Generate shopping list	1.0	High
RF-011: Categorize ingredients in shopping list	1.0	Med
RF-012: Optimize shopping list for store/budget	1.0	High
RF-013: Display recipes with full details	1.0	High
RF-014: Mark favorites, exclude or substitute items	1.0	Med
RF-015: Budget tracking and alerts	2.0	Low
RF-016: Notifications and reminders	1.0	Med
RF-017: Session logout	1.0	High
RF-018: Remove account and erase info	1.0	High

RF-019: Personal nutritionist chatbot *(experimental)*	2.0	Low
RF-020: User settings management	1.0	Med
RF-021: Data export	2.0	Low
RF-022: Keep track of user food inventory	2.0	Med

4.2. FUNCTIONAL SECURITY REQUIREMENTS

4.2.1. Treatment of access to personal data

The system will store the following information in different files, each one with a different level of security to ensure the user's information remains private.

- User profile (email, preferences, settings) → Security Level: Medium
- Nutrition preferences data → Security Level: High
- Budget and store preferences → Security Level: Medium
- Saved recipes and meal plans → Security Level: Low

Access to personal data will be protected using account-level authentication. Sensitive information such as login credentials will be encrypted.

4.2.2. RF-017: Session logout

The system must provide a manual logout option to prevent unauthorized access from unattended devices.

4.2.3. RF-018: Remove account and erase info

The system must provide an option to remove the account and erase all personal data from the system when they no longer want to use the application.

4.3. USER PROFILE AND PREFERENCES

This module handles user account creation, session management, and configuration of personal preferences related to nutrition, budgeting, and shopping habits.

4.3.1. RF-001: User registration and login

The system must allow users to register with a valid email and password. Users must be able to log in securely and access their personal configurations.

4.3.2. RF-002: Set diet type and restrictions

Users must be able to define their diet (e.g., vegetarian, vegan, omnivorous) and mark any food allergies or ingredient restrictions.

4.3.3. RF-003: Set nutritional goals

Users must be able to configure daily nutritional goals such as target calories, protein, carbs, and fat intake.

4.3.4. RF-004: Set weekly food budget

The system must let users input a weekly budget that will guide recipe and shopping suggestions.

4.3.5. RF-005: Select preferred supermarket

Users can optionally choose one or more preferred stores, which will influence shopping list optimization.

4.3.6. RF-006: Select preferred cooking time

Users must be able to select their typical available time for meal preparation (e.g., <20 mins, up to 1h), which will be used in filtering recipes.

4.3.7. RF-020: User settings management

Users must be able to modify their email, password, measurement units, and notification preferences.

4.4. MEAL PLANNING ENGINE

This module is responsible for generating personalized meal plans based on user preferences and nutritional needs.

4.4.1. RF-007: Generate a weekly meal plan

The system must create a personalized weekly plan that meets the user’s goals and restrictions, including breakfast, lunch, dinner, and optional snacks.

4.4.2. RF-008: Smart recipe filtering

The meal plan must use filters such as cooking time, ingredient restrictions, dietary preferences, and budget to adapt the recipes.

4.4.3. RF-009: Nutritional summary of plan

After generation, the system must display an overview of daily and weekly calorie intake and macronutrient distribution.

4.5. SHOPPING LIST GENERATOR

This module creates a smart shopping list based on selected recipes and user constraints.

4.5.1. RF-010: Generate shopping list

The system must automatically generate a shopping list based on all planned recipes.

4.5.2. RF-011: Categorize ingredients in shopping list

The shopping list must group items into categories (e.g., vegetables, grains, proteins) to ease navigation.

4.5.3. RF-012: Optimize shopping list for store/budget

The shopping list must adapt quantities, brands, or choices to match the preferred store and user's weekly budget.

4.5.4. RF-021: Data export

Users must be able to export their shopping list and/or weekly plan in PDF or plain text format.

4.5.5. RF-022: Keep track of user food inventory

Users will be able to input the ingredients they have and when they buy new ones, so that the app creates a meal plan minimizing food waste and overall costs.

4.6. RECIPE BROWSER & SELECTOR

This module allows users to view, customize, and interact with recipe suggestions.

4.6.1. RF-013: Display recipes with full details

Recipes must display complete information: ingredients, preparation steps, preparation time, nutrition facts, and difficulty.

4.6.2. RF-014: Mark favorites, exclude or substitute items

Users must be able to mark recipes as favorites, exclude specific ingredients, or request substitutions for ones they dislike or can't consume.

4.7. ADDITIONAL FEATURES

This section includes complementary features that improve user experience or are planned as future enhancements.

4.7.1. RF-015: Budget tracking and alerts

The system must allow users to track their spending across shopping lists and raise alerts when the budget is exceeded.

4.7.2. RF-016: Notifications and reminders

Users will receive reminders for shopping, expired items, or pending recipes.

4.7.3. RF-019: Personal nutritionist chatbot (experimental)

The system will include an optional chatbot to answer nutrition-related questions and suggest small changes in plans. It will operate in beta and may not always offer accurate advice.

5. NON-FUNCTIONAL REQUIREMENTS

5.1. CATALOGUE OF NON-FUNCTIONAL REQUIREMENTS

Non-Functional Requirement	Version	Priority
RNF-001: API Interoperability with Nutrition Platforms	2.0	Low
RNF-002: App Stability	1.0	High
RNF-003: Basic Error Handling	1.0	High
RNF-004: Reasonable Response Time	1.0	High
RNF-005: Concurrent Users	2.0	Low
RNF-006: Mobile-first Design	1.0	High
RNF-007: Easy Onboarding	1.0	Med
RNF-008: In-App Help	1.0	Med
RNF-009: Test Coverage	1.0	High
RNF-010: Offline Functionality	1.0	Low
RNF-011: Supported Frameworks	1.0	Med
RNF-012: Readable Code	1.0	High
RNF-013: Multilingual Support	1.0	Med

5.2. INTEROPERABILITY REQUIREMENTS

5.2.1. **RNF-001: API Interoperability with Nutrition Platforms.** The system may optionally integrate APIs (e.g., Spoonacular) to help generate meal plans and retrieve nutrition data.

5.3. RELIABILITY REQUIREMENTS

5.3.1. **RNF-002: App Stability.** The app must function reliably during regular use, with minimal crashes or freezes.

5.3.2. **RNF-003: Basic Error Handling.** If an external service fails (e.g., API timeout), the app should display an error and continue functioning.

5.4. EFFICIENCY REQUIREMENTS

5.4.1. **RNF-004: Reasonable Response Time.** User requests (e.g., getting meal suggestions) must be processed within 2 seconds on average under normal usage.

5.4.2. **RNF-005: Concurrent Users.** The system should support at least 200 concurrent users during peak usage hours.

5.5. USABILITY REQUIREMENTS

5.5.1. **RNF-006: Mobile-first Design.** The user interface must be optimized for mobile devices.

5.5.2. **RNF-007: Easy Onboarding.** New users must be able to understand and use basic app features (budget setting, selecting meals) within 5 minutes.

5.5.3. **RNF-008: In-App Help.** The app should provide access to help content, like tutorials or FAQs.

5.6. SECURITY REQUIREMENTS

5.6.1. **RNF-009: Test Coverage.** At least 60% of the application's logic should be covered by tests to ensure robustness and minimize bugs.

5.7. PORTABILITY REQUIREMENTS

5.7.1. **RNF-010: Offline Functionality.** Users should be able to view saved plans and shopping lists without internet access.

5.8. MAINTAINABILITY REQUIREMENTS

5.8.1. **RNF-011: Supported Frameworks.** Widely supported libraries and frameworks that are easy to maintain have to be used.

5.8.2. **RNF-012: Readable Code.** The codebase must be well-documented, follow best practices, and be understandable by other developers.

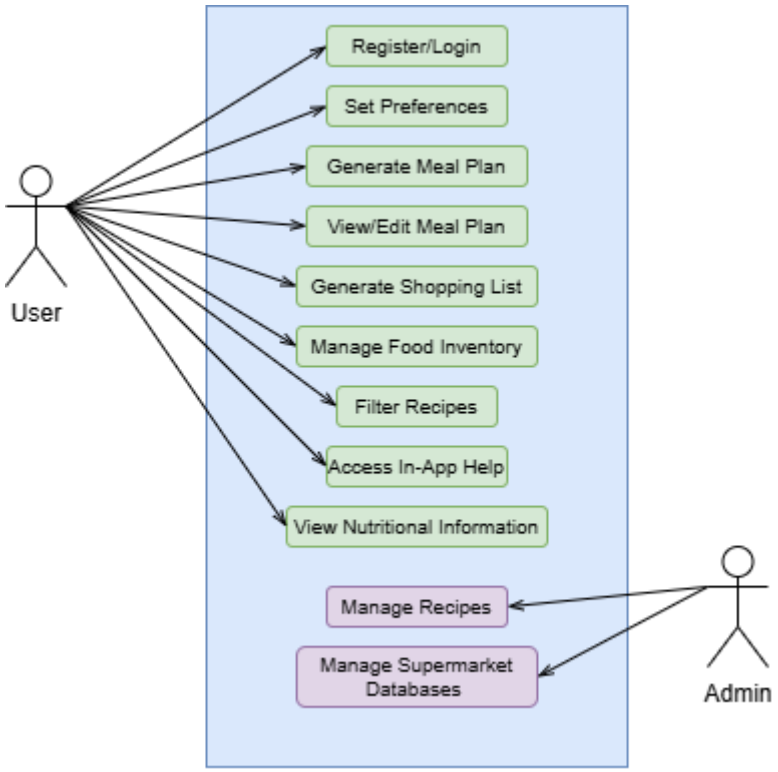
5.9. ACCESSIBILITY REQUIREMENTS

5.9.1. **RNF-013: Multilingual Support.** The app should support multiple languages (e.g., English, Spanish, Catalan)

6. USE CASES

This section includes the set of use cases that describe the external behavior of the system.

6.1. USE CASE DIAGRAMS AND ACTOR HIERARCHY



6.2. CATALOGUE OF ACTORS

6.2.1. User

Type	Primary
Description of their role	Interacts with all core features: setting preferences, generating meal plans, managing inventory, and using shopping lists.
Objectives	Obtain a customized meal plan.
Related Use Cases	Register/Login, Set Preferences, Generate Meal Plan, View/Edit Meal Plan, Generate Shopping List, Manage Food Inventory, Filter Recipes, View Nutritional Information, Access In-App Help

6.2.2. Admin

Type	Secondary
Description of their role	Manages the database of food items, nutrients, and dietary rules. Ensures the system is working correctly and handles any issues the users have.
Objectives	Make sure the system works smoothly and the database is up to date and accurate.
Related Use Cases	Manage Recipes, Manage Supermarket Databases

6.3. CATALOGUE OF USE CASES

Turn your interest ID	Use Case name	Actors
1	Register/Login	User
2	Set Preferences	User
3	Generate Meal Plan	User
4	View/Edit Meal Plan	User
5	Generate Shopping List	User
6	Manage Food Inventory	User
7	Filter Recipes	User
8	View Nutritional Information	User
9	Access In-App Help	User
10	Manage Recipes	Admin
11	Manage Supermarket Databases	Admin

6.4. DESCRIPTION OF USE CASES

6.4.1. Register/Login

Use Case name	Register/Login
Description of the use	Allows the user to access a personalized experience by creating an account or logging in.
Actors participants	User
Basic event flow	1. User opens the app 2. Chooses to register or log in 3. Enters credentials 4. App verifies credentials and opens main dashboard

6.4.2. Set Preferences

Use Case name	Set Preferences
Description of the use	Allow the user to set dietary preferences, nutritional goals, cooking time, and budget.
Actors participants	User
Basic event flow	1. User opens the preferences/settings screen 2. The user enters data such as age, weight, height, goals, supermarket, time, etc. 3. The system saves the data and confirms successful registration.

6.4.3. Generate Meal plan

Use Case name	Generate meal plan
Description of the use	The system analyzes the input data and generates a customized meal plan.
Actors participants	User
Basic event flow	1. User taps "Generate Plan" 2. The system processes preferences to a suitable meal plan 3. The user receives a list of daily meals and their corresponding recipes

6.4.4. View/Edit Meal Plan

Use Case name	View meal Plan
Description of the use	Allows the user to view their current plan and regenerate or replace meals if needed.
Actors participants	User
Basic event flow	1. User opens "My Plan" 2. Meal plan is displayed 3. User can replace a day or regenerate a full plan

6.4.5. Generate Shopping List

Use Case name	Generate Shopping List
Description of the use	Creates a shopping list from the selected meal plan.
Actors participants	User
Basic event flow	1. User taps "Get Shopping List" 2. App compiles ingredients 3. Groups by category and shows list

6.4.6. Manage Food Inventory

Use Case name	Manage Food Inventory
Description of the use	The user can keep track of which ingredients they already have to reduce waste.
Actors participants	User
Basic event flow	1. User opens inventory screen 2. Adds/removes ingredients they have 3. App updates meal suggestions or shopping list

6.4.7. Filter Recipes

Use Case name	Filter Recipes
Description of the use	Filters recipes according to dietary preferences, ingredient limits, or preparation time.
Actors participants	User
Basic event flow	1. User defines filters (e.g., "under 30 min", "no dairy") 2. App shows only recipes that match 3. Filters are saved for future plans

6.4.8. View Nutritional Information

Use Case name	View Nutritional Information
Description of the use	Displays the calories, protein, carbs, and fat of meals and the whole meal plan.
Actors participants	User
Basic event flow	1. User opens meal plan or recipe 2. App shows macro- and micronutrient information

6.4.9. Access In-App Help

Use Case name	Access In-App Help
Description of the use	User can access tutorials or FAQs to learn how to use the app.
Actors participants	User
Basic event flow	1. User taps "Help" or "?" 2. App displays short guides or answers

6.4.10. Manage Recipes

Use Case name	Manage Recipes
Description of the use	Admins can add, modify, or delete recipes from the app database.
Actors participants	Admin
Basic event flow	1. Admin logs into recipe dashboard 2. Creates or edits recipe entries 3. Database is updated

6.4.11. Manage Supermarket Databases

Use Case name	Manage Supermarket Databases
Description of the use	Allows admins to add/remove/edit food items and nutritional data from the database
Actors participants	Admin
Basic event flow	1. Admin logs in to the backend 2. Adds or edits food entries 3. System validates and updates the database