Calories Calculation Application

This document outlines the functional and non-functional requirements for the project, aimed at developing a calorie and expenditure tracking application. *The functional requirements define what the system must do, while the non-functional requirements specify the system's expected qualities and constraints.*

Functional / Non-Functional Requirements

FR1: **Account Management**

- Create a new account and allow users to sign in.

FR2: **Add a New Food Entry**

- Each food entry must include the following information:
  - Date/time when the food was taken
  - Food/product name (e.g., Apple, ice cream, meat)
  - Calorie value (numeric)

FR3: **Calorie Threshold Management**

- Set a daily calorie threshold limit of 2,500 calories.
- Allow users to view the days when this limit was reached.
- Display a warning message when the daily calorie limit is exceeded.

FR4: **Monthly Expenditure Tracking**

- Allow users to enter the price for each food entry.
- Display a warning message when a user spends more than €1,000 in a month. The warning should be displayed similarly to the calorie limit warning.

FR5: **Weekly Summary Notification**

- When the user logs in, display a report that includes the following:
  - Total calories consumed per day for the week.
  - The number of days the calorie threshold was exceeded.
  - Total expenditure for the week.

FR6: **Food Entry Filtering**

- Enable users to filter food entries by date range (from date/to date).

FR7: **Admin Role Features**

- Admins should have a dedicated screen to view and manage all food entries (create, read, update, delete).
- Admins should access a reporting screen with the following statistics:
    - The number of entries added in the last 7 days versus the week before (including the current day).
    - The average number of calories per user for the last 7 days.
    - A list of users who exceeded the monthly price limit in the previous month.
- Regular users must not access the admin reporting screen or its data.

1. **NFR1**: **Usability**
    - The application should be intuitive to use, with an easy-to-understand interface.
    - All interactions must be completed in fewer than three clicks.
2. **NFR2**: **Target Platform**
    - The application must be developed in Java.
3. **NFR3**: **Version Control**
    - Use Git for version control and host the project on GitHub.

Additional constraints:

- The application can be created as a **web/desktop** application

Target Environment

The application should be demonstrated in Java.

For this assignment, you must build teams of 4 members working **together**.

Scenarios

Scenario for FR1: Account Management

- A user registers with their details (name, email, password) and successfully logs in to access their dashboard.

Scenario for FR2: Add a New Food Entry

- The user adds a food entry with the date/time, food name, and calorie value, and saves it. The entry is then logged in their food list.

Scenario for FR3: Calorie Threshold Management

- A user logs food entries throughout the day. When the total calories exceed 2,500, the system shows a warning: *"Daily calorie limit exceeded."*

Scenario for FR4: Monthly Expenditure Tracking

- A user logs multiple food prices in a month. Upon exceeding €1,000, a warning is displayed: *"Monthly spending limit exceeded."*

Scenario for FR5: Weekly Summary Notification

- When the user logs in, they see a report showing total calories per day, days the calorie limit was exceeded, and total expenditure for the week.

Scenario for FR6: Food Entry Filtering

- The user filters food entries by specifying a date range and views entries from that period only.

Scenario for FR7: Admin Role Features

- An admin manages food entries (add, edit, delete) and views a report with statistics like entries added in the past 7 days, average calories per user, and users exceeding the spending limit.

Part I (Requirement specification: AGILE METHODOLOGIES (SCRUM) - SPECIFY USER STORIES)

Consider requirements specification *(pages 1,2 in this document)*

1. Role Product Owner: Create the **product backlog --** *Write all necessary user stories to create the calorie calculation application (Document 1)*
2. Role SCRUM Master: For each sprint, specify the **sprint backlog. (**Stories that depend on each other should not be inserted in the same sprint – except for the cases when these stories are so small that they can be implemented in the same sprint but in not overlapping times) *(Document 2)*

Suppose that the developer team is composed of 5 developers. Each sprint lasts three weeks.

Remember that:

- a user story is implemented by one developer
- should be implementable in one sprint (including also the time needed to test the story).
- during the same sprint one developer can implement more than one story – in cases when the stories are small.

Each story should have a point number. To specify this number, you should consider the complexity and the time needed to implement the story. One day is converted to 0.5 points.

(Read the uploaded document: "TaskAgile - Example Agile Requirements.pdf". This document can help you write user stories)

Part II (Implementing/Testing Software)

Consider requirements specification ***"Calories Calculation Application"***

- Implement the application using Java as a programming language and a relational database

  Apply what you have learned about software architecture, software design, and internal quality.

  The code should not only work but also it should be written in a way that is easily maintainable (readable, understandable…). Apply the refactoring patterns discussed in the lectures.

- Write Unit/Integration tests using the Junit testing framework in order to achieve not less than 50% of code coverage

  The tests should be maintainable (have a good internal quality)