HealthNCare App / 1

Project Design Document

#4 Team4AndCo.LLC.INC

Izzy Pedrizco-Carranza <ip7431@rit.edu>

Derek Kasmark <djk2529@rit.edu>

Colin Chomas <cwc6640@rit.edu>

Adhel Siddique <as8758@rit.edu>

Google Doc link to document with version history:

https://docs.google.com/document/d/1ixyqrHVhnMYQA3XHvu1HFkSmE65J_aWUPPTi8YTze80/edit?usp=sharing

1 Project Summary

An application we are developing is a health care app for users who want to log in their daily intake of foods and store food recipes. Additionally, this allows users to monitor and use data from the app to improve their daily lives. The application provides a list of basic foods which users can combine into recipes as well as adding new basic ingredients or recipes to the system. The information they must provide is the name of the food, calories, grams of fat, grams of proteins, and many more.

Moreover, the application has a feature of storing log entries for daily intake of consumption daily. The purpose of this is users can look at what food they have eaten and how much they need to adjust to their daily intake to improve their health.

Overall, users will be able to use the health food app that can improve their daily lives as well as carefully analyze what food they can add and make their daily intake. They can log their daily intake to get specific intake they need such as proteins, carbs, and many more. In addition, they can edit or delete some of their log entries based on their preferences.

2 Design Overview

The design of our healthcare application evolved from initial sketches to its final implementation with a focus on modularity, scalability, and maintainability. We structured the system into four main subsystems:

1. **User Management** – Handles authentication and profile settings.

2. **Food Database** – Stores predefined food items and user-added entries.

3. **Recipe Management** – Enables users to create and save meal combinations.

4. **Daily Log System** – Records user food consumption for health tracking.

Key Design Principles Applied

• Separation of Concerns: Each subsystem operates independently, reducing unnecessary dependencies.

• High Cohesion & Low Coupling: Components interact via well-defined interfaces, making the system easier to modify and extend.

• Dependency Inversion: Abstractions (interfaces) allow for future feature expansions without affecting existing implementations.

Design Decisions & Adjustments

• Initially, a monolithic database structure was considered but later revised to a modular approach for improved efficiency.

• The original manual food entry system was enhanced with auto-suggestions and predefined food categories for better usability.

Assumptions & Future Considerations

• Users are expected to input accurate nutritional data when adding new food items.

• The system assumes internet connectivity for cloud-based storage.

• Future versions may incorporate machine learning for personalized health recommendations.

3 Subsystem Structure

This section provides a graphical model of the subsystems that comprise the application as a whole.

Subsystems are groups of closely related classes. For example, an email program might have subsystems for contact list management, message composition, and message delivery, among many others. In large applications, a subsystem might even have sub-subsystems, though this should not be necessary in this course.

A design goal is to have much lower coupling among classes of *different* subsystems than among the classes *within* a subsystem. Put another way, each subsystem should be highly cohesive in purpose, and the subsystems as a group should exhibit separation of concerns, just as the set of classes *within* a given subsystem should each be highly cohesive, while the set as a whole exhibits separation of concerns.

Draw the subsystems as simple boxes with lines (possibly terminated in arrows) showing relationships between them. Include in each subsystem box a brief narrative of the subsystem's purpose and/or responsibilities. Label important relationship lines with an indication of what the relationship represents.

Replace all the blue text in this section with a diagram conforming to the description above and any complimentary textual description if appropriate. We suggest you ONLY remove blue sections instructions ONLY AFTER you have confirmed with all contributors that the instructions for a particular section have been fully covered prior to submission.

NOTE: the final Rationale section is meant to complement (not Duplicate) the information contained in other parts of the document.

4 Subsystems

This section contains one subsection for each of the subsystems in the high-level diagram above, describing the classes in the subsystem and their relationship. In what follows, ***class*** is a generic term for *concrete classes*, *abstract classes*, and *interfaces*).

Each subsystem has a CRC table naming each class in the subsystem, the class's responsibilities, and the class's collaborators (classes in this or other subsystems).

The table is followed by a UML class diagram showing inheritance (generalization) relationships between classes and interfaces and associations between objects in these classes.

In the actual document, this entire blue section should simply be deleted.

4.1 Subsystem name

Fill in a CRC table like those below for each class or interface in the subsystem; feel free to copy, paste, and edit our examples.

The example tables, based on a hypothetical library media database system, illustrates all the possible collaborations between classes and objects that you must record for your design. ***Remove this blue paragraph from your document.***

Class MediaCollection	
Responsibilities	Support access to the media in the library collection. Add, find, delete an existing media in the collection. Provide virtual collection for the entire library consortium collections.
Collaborators (uses)	Media - the basic type for all different media in the collection. network. Consortium - consortium communication (network subsystem).

Class Media (interface)	
Responsibilities	Provide a generic interface to all media types in the library. Includes the media name, a unique media id. Can retrieve a formatted media description.

Class Book	
Responsibilities	Represents a book in the collection.

	Provides access to author, title, genre, and ISBN for the book. Provides access to other attributes of a book (e.g., number of pages).
Collaborators (inheritance)	Media
Class DVD	
Responsibilities	Represents a DVD in the collection. Provides access to title, genre for the DVD. Provides access to other attributes of a DVD (time, cast, etc.).
Collaborators (inheritance)	Media

Each subsystem will also include an ***in-line*** UML class diagram of the system. In the simplest case the interfaces and classes will simply have class boxes with the appropriate name. In the final document, and preferably in intermediate documents, classes will also include the public methods provided by the class.

If a class in this subsystem collaborates with a class in a different subsystem, simply include link to a box with the ***other subsystem's name***.

4.2 Subsystem name

As described above for the second, third, etc. subsystems.

5 Sequence Diagrams

5.1 Description of labeled Sequence diagram #1 and (what feature / operation / scenario the diagram shows).

Include a sequence diagram that corresponds to the description above. Feel free to use a stick figure "pretend" object for whatever outside agent triggers the operation. In addition, it is acceptable to rotate your diagram to landscape orientation that makes the diagram easier to read.

Repeat as many times as necessary:

5.2 Description of labeled Sequence diagram #N (what feature / operation / scenario the diagram shows).

Same as above. Provide diagrams for each of the basic operations in the problem description and/or those operations specified from discussion with your instructor/stakeholder.

6 Pattern Usage

There will be a subsection for each pattern you use in your design (***including*** those that may be required in the project description or by your instructor).

6.1 Pattern #1 Name

Create a table like the one below listing the generic "roles" from the pattern description along with the specific class(es) of objects in your design that are associated with each role. For instance, if we are using the Observer pattern in an Alarm system where one or more alarm reporting objects observe one or more sensor objects, we might see a table like the following:

Observer Pattern	
Observer(s)	SoundAlarmReporter ADPAlarmReporter AlarmReporter911
Observable(s)	TemperatureSensor MotionSensor DoorOpenSensor

6.2 Pattern #2 Name

Just repeat the information above for the second and following patterns.

7 RATIONALE

Be sure to incorporate all major DESIGN and ARCHITECTURAL decisions and reasons for pursuing them. It helps to add entries with a time stamp (e.g., 9/27/2023 –We decided to..) >