Aegle

E+1

Sebastian Zawadzki - 201632011

sebastianzawadzki@gmx.de

Kay Gillmann - 201632012

kaygillmann@aol.de

Kevin Oh (오주환) – 201520936

kevinoh94@gmail.com

Lucy Liu – 201632020

lucy.liu@myy.haaga-helia.fi

Tam Tran – 201632027

h8265@student.jamk.fi

Elaboration Report

6 Mai 2016

Domain analysis and design

Table of Contents

[2 Vision 1](#_Toc450456000)

[2.1 Introduction 1](#_Toc450456001)

[2.2 Positioning 2](#_Toc450456002)

[2.2.1 Business Opportunity 2](#_Toc450456003)

[2.2.2 Problem Statement 2](#_Toc450456004)

[2.2.3 Product Position Statement 2](#_Toc450456005)

[2.2.4 Alternatives and Competition 3](#_Toc450456006)

[2.3 Stakeholder Description 3](#_Toc450456007)

[2.3.1 Stakeholder Summary 3](#_Toc450456008)

[2.3.2 User Summary 3](#_Toc450456009)

[2.3.3 Key High-level Goals and Problems of the Stakeholders 3](#_Toc450456010)

[2.3.4 User Goals 4](#_Toc450456011)

[2.3.5 User Environment 4](#_Toc450456012)

[2.4 Product Overview 5](#_Toc450456013)

[2.4.1 Product Perspective 5](#_Toc450456014)

[2.4.2 Summary of Benefits 5](#_Toc450456015)

[2.4.3 Assumptions and Dependencies 5](#_Toc450456016)

[2.4.4 Other Requirements and Constraints 6](#_Toc450456017)

[3 Glossary 6](#_Toc450456018)

[4 Requirements 7](#_Toc450456019)

[4.1 Functional Requirements 7](#_Toc450456020)

[4.1.1 Use Case Diagram 7](#_Toc450456021)

[4.1.2 Use Case List 8](#_Toc450456022)

[4.1.3 Use Case Text 9](#_Toc450456023)

[4.2 Non-functional Requirements 15](#_Toc450456024)

[4.2.1 Introduction 15](#_Toc450456025)

[4.2.2 Functionality 15](#_Toc450456026)

[4.2.3 Usability 16](#_Toc450456027)

[4.2.4 Reliability 16](#_Toc450456028)

[4.2.5 Purchased components 17](#_Toc450456029)

[4.2.6 Implementation Constraints 17](#_Toc450456030)

[5 Domain Model 18](#_Toc450456031)

[5.1 Domain Model Diagram 18](#_Toc450456032)

[5.1.1 Domain classes 18](#_Toc450456033)

[6 System Sequence Diagram 21](#_Toc450456034)

[6.1 UC1 – Do exercise program 21](#_Toc450456035)

[6.1.1 UC1 Main 1 – SSD 21](#_Toc450456036)

[6.1.2 UC1 Main 1– Operation Contracts 22](#_Toc450456037)

[6.1.3 UC1 Main 2 – SSD 23](#_Toc450456038)

[6.1.4 UC1 Main 2 – Operation Contracts 23](#_Toc450456039)

[6.2 UC2 – Do diet program 24](#_Toc450456040)

[6.2.1 UC2 – SSD 25](#_Toc450456041)

[6.2.2 UC2 – Operation Contracts 25](#_Toc450456042)

[6.3 UC3 – Interact with the social media 26](#_Toc450456043)

[6.3.1 UC3 – SSD 26](#_Toc450456044)

[6.3.2 UC3 – Operation Contracts 26](#_Toc450456045)

[6.4 UC4 – Sees contents 27](#_Toc450456046)

[6.4.1 UC4 – SSD 27](#_Toc450456047)

[6.4.2 UC4 – Operation Contracts 27](#_Toc450456048)

[6.5 UC5 – Manage users 28](#_Toc450456049)

[6.5.1 UC5 – SSD 28](#_Toc450456050)

[6.5.2 UC5 – Operation Contracts 28](#_Toc450456051)

[7 Design Model 29](#_Toc450456052)

[7.1 Do Exercise Program Realization 29](#_Toc450456053)

[7.1.1 Main1 - Operation 1 29](#_Toc450456054)

[7.1.2 Main1 - Operation 2 30](#_Toc450456055)

[7.1.3 Main1 - Operation 3 31](#_Toc450456056)

[7.1.4 Main1 - Operation 4 32](#_Toc450456057)

[7.1.5 Main1 - Operation 5 33](#_Toc450456058)

[7.1.6 Main1 - Operation 6 34](#_Toc450456059)

[7.1.7 Main2 - Operation 1 35](#_Toc450456060)

[7.1.8 Main2 - Operation 2 36](#_Toc450456061)

[7.1.9 Main2 - Operation 3 37](#_Toc450456062)

[7.1.10 Main2 - Operation 4 38](#_Toc450456063)

[7.1.11 Main2 - Operation 5 39](#_Toc450456064)

[7.2 Design Class Diagram 40](#_Toc450456065)

[8 References 41](#_Toc450456066)

[9 Content of Figures 41](#_Toc450456067)

[10 Content of Tables 42](#_Toc450456068)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No. | Date | Version | Changed chapters | Description of change | | Authors |
| 1 | 2016-03-24 | 1.0 | all | Inception Phase Draft 1 | | Team E+1 |
| 2 | 2016-05-02 | 1.1 | 4.1.1 Use Case Diagram | Deletion of UC1 and UC7, changed name of new UC1 and UC2 | | Team E+1 |
| 3 | 2016-05-02 | 1.1 | 4.1.2 Use Case List | Content of UC3 | | Team E+1 |
| 4 |  |  |  |  | | Team E+1 |
| No. | Date | | New chapters | | Authors | |
| 1 | 2016-05-02 | | 5 Domain Model | | Team E+1 | |
| 2 | 2016-05-02 | | 6 System Sequence Diagram | | Team E+1 | |

# Vision

## Introduction

We envision a next generation, social, all-in-one smart fitness companion application, Aegle, designed to have the flexibility to be useful for anyone interested in fitness, containing multiple user interface mechanisms supported by a variety of devices intuitively. The idea behind Aegle arose from the inconvenience and general incompleteness of existing fitness related applications. This system will provide a convenient way for users to start and maintain a regular exercise program, along with a proper diet.

Aegle is also integrated with its own social media platform. Users will be able to share their personal achievements and milestones, along with the ability to fit in with others in your own niche of the health community. The greatest advantage of the social media platform is the ability for users to share their workout programs and recipes. Other users can try them out and leave reviews. The most popular workouts and recipes will then be showcased and recommended to users. Doing this will ensure that Aegle will always have the best information available to our users.

## Positioning

### Business Opportunity

A recently published study found that only 2.7 % of U.S. adults live a healthy lifestyle. A significant part of living a healthy lifestyle comes from fitness and diet. Almost anyone could benefit from eating a better diet and being more active. Yet the number of healthy people is so low because of the barriers to getting fit, such as the time, money, and know-how required. Existing fitness related applications are numerous but are lacking in important features. Current applications fail to be essential. They are not satisfying to use. Aegle will be different because of how its community driven. Since users will be able to get information and advice directly from experts it will be more exciting and useful for users to use.

### Problem Statement

Fitness is a science and can be intimidating for new users because of the overwhelming amounts of information and conflicting regimes. Since everyone is different there is no correct way to get fit. Our system is designed to ease the burden of information by providing users with accurate information verified by other users.

### Product Position Statement

Aegle is designed to be the ultimate fitness application, specifically geared towards beginners, but useful for anyone. Aegle combines the exercise and diet aspects of fitness and provides users with relevant information and research so that anyone can just dive in and start becoming more fit.

The most unique part of Aegle comes from its community. A specially designed social media seamlessly integrated with the system will allow users to share their workout programs and favorite recipes.

Aegle will give users the ability to create highly detailed meal plans with nutrition and calorie content of individual meals to help them map out the road to a healthier lifestyle.

The goal behind Aegle is to give a complete beginner the ability to step right in and begin their journey to a healthier lifestyle the moment they download the application.

### Alternatives and Competition

There are numerous fitness and diet related applications already available on the market. What differentiates Aegle from similar apps is Aegle’s more intuitive and useful user interface, and the sharing system for workout programs and diet ideas. Aegle offers a much more detailed dietary plan, letting users map out each individual meal to reach their calorie goals. The biggest selling point of Aegle is the fact that it brings everything together so that anyone can begin to live a healthier lifestyle with just one application and minimum research.

## Stakeholder Description

### Stakeholder Summary

*System Administrator* - Designed the software and pushes out updates in order to gain a userbase. The application will be free to download but offer greater features in the form of in-app purchases in order to make a profit.

*Social Media Administrator -* Moderates social media and the sharing system. The driving force behind the success of the application is its community. Creating and establishing a large community will ensure that good, relevant information will always be available to users. Thus further growing the user base, and profit.

### User Summary

*General User* - Can share content that worked for them, leave reviews for others’ content. Can see and download popular content from the social media. Can join and participate in communities.

*Certified User* - Certified health experts are verified and given special privileges. Their posts and reviews hold greater weight in the community.

### Key High-level Goals and Problems of the Stakeholders

Table 2‑1 Key High-level Goals

|  |  |  |  |
| --- | --- | --- | --- |
| **High-Level Goal** | **Priority** | **Problems and Concerns** | **Current Solutions** |
| Fast and easy to get started exercise routines | high | - Exercise routines need to meet certain standards  - Everyone is different so a routine that works for one person may not be right for someone else | Leave it to the user to do the research and figure out a routine that will work for them |
| Healthy and nutritious dieting | high | - Recipes must be cheap and easy to make  - Recipes need to be delicious  - Recipes need to be flexible for varying calorie targets | There are plenty of services available for figuring out proper calorie and nutrition balance but not as easy to design a meal plan to match |
| Sharing of accurate and relevant information in the health world | medium | - Fitness is not a completely established science  -There is a lot of misinformation and conflicting information | User can join existing health communities to ask questions and gain knowledge |

### User Goals

*General User* - Wants to start and maintain an effective workout program along with a healthy diet. Don’t want to be burdened by complicated information and overly expensive or complicated foods.

*Certified User* - Having trained health professionals recognized improves the quality of the content available and allows the experts to help others while promoting themselves and therefore profiting from using the application.

### User Environment

* General user can log in, connect to the integrated social media with an existing social media account (such as google or Facebook)
* Users can see and download exercise routines from the integrated social media
* Users can find their recommended calorie intake based on their goal, height, weight, age, sex, and activity-level
* Users can see and download recipes to design a meal plan to match their recommended calorie intake
* Users can see and get advice from Certified Users who are verified health experts
* Users can share personal milestones and information through the integrated social media
* Social media administrators moderate the content that is uploaded and interactions between the community
* System administrators ensure safety of the user’s personal information and manages verification of Certified Users

## Product Overview

### Product Perspective

Other existing apps only offer the part of an exercising or a diet program. With Aegle the user is able to generate his own exercising program, combined with a balanced food plan to get the most efficacy. Also the user able to share his achievements at social media platforms or compare it with other users from the app, to check his goals. Aegle is an all-in-one app to structure the daily routine as much as possible.

### Summary of Benefits

Table 2‑2 Summary of Benefits

|  |  |
| --- | --- |
| **Features** | **Benefits** |
| Display exercises for the user | Helps user begin and maintain a regular exercise routine |
| Display food plan | Helps user begin and maintain a healthy diet |
| Get information about the substance in the food | Makes it easier to create a meal plan with the proper calorie and nutritious content |
| Share workout goals to the social media | Give users motivation and be a part of a community |

### Assumptions and Dependencies

* The social media has a large enough user base that people are actively participating in sharing and reviewing content
* Aegle needs a sufficiently sized database of foods that users can pick from to get nutrition and calorie information in order to create meal plans
* Certified health experts join the community along with generally fitness experienced people

### Other Requirements and Constraints

* The design of an interface (GUI) that shows enough information for the user but still clearly arranged
* The idea behind Aegle is the promise of a great wealth of useful information, but we are relying on having a user base to provide that information. So we will need to provide a large amount of information to begin with in order to drive the creation of a user base
* We are relying on establishing a user base large enough that health professionals would want to join and participate in order to promote themselves to a large community. So we need users to draw in professionals in order to draw in users

# Glossary

|  |  |
| --- | --- |
| **Term** | **Definition and Information** |
| Aegle | Name of the software system. Aegle is the name of the Greek Goddess of Radiance and Good Health |
| Social Media | Aegle will contain an integrated social media service designed to be used for sharing content such as exercise programs, meal plans, and recipes. |
| (General) User | Normal user of the application |
| Certified User | A verified health expert who holds greater weight when sharing content |
| Social Media Administrator | Administrator who manages user accounts. Moderates the community.  Verifies certified users. |
| System Administrator | Manages the software. |
| Exercise Program/Routine | A set exercise routine meant to be followed regularly. |
| Diet Program/Meal Plan | A meal plan meant to be followed regularly. Is highly specific mapping out nutrition and calorie content of each individual meal. |
| User Base | The total of the users who make up the community. The users are the ones who primarily create and share content. |
| Content | The information uploaded, downloaded, and reviewed by users such as exercise programs, diet programs, recipes, general advice. |
| Database | System that holds all content |

# Requirements

## Functional Requirements

### Use Case Diagram

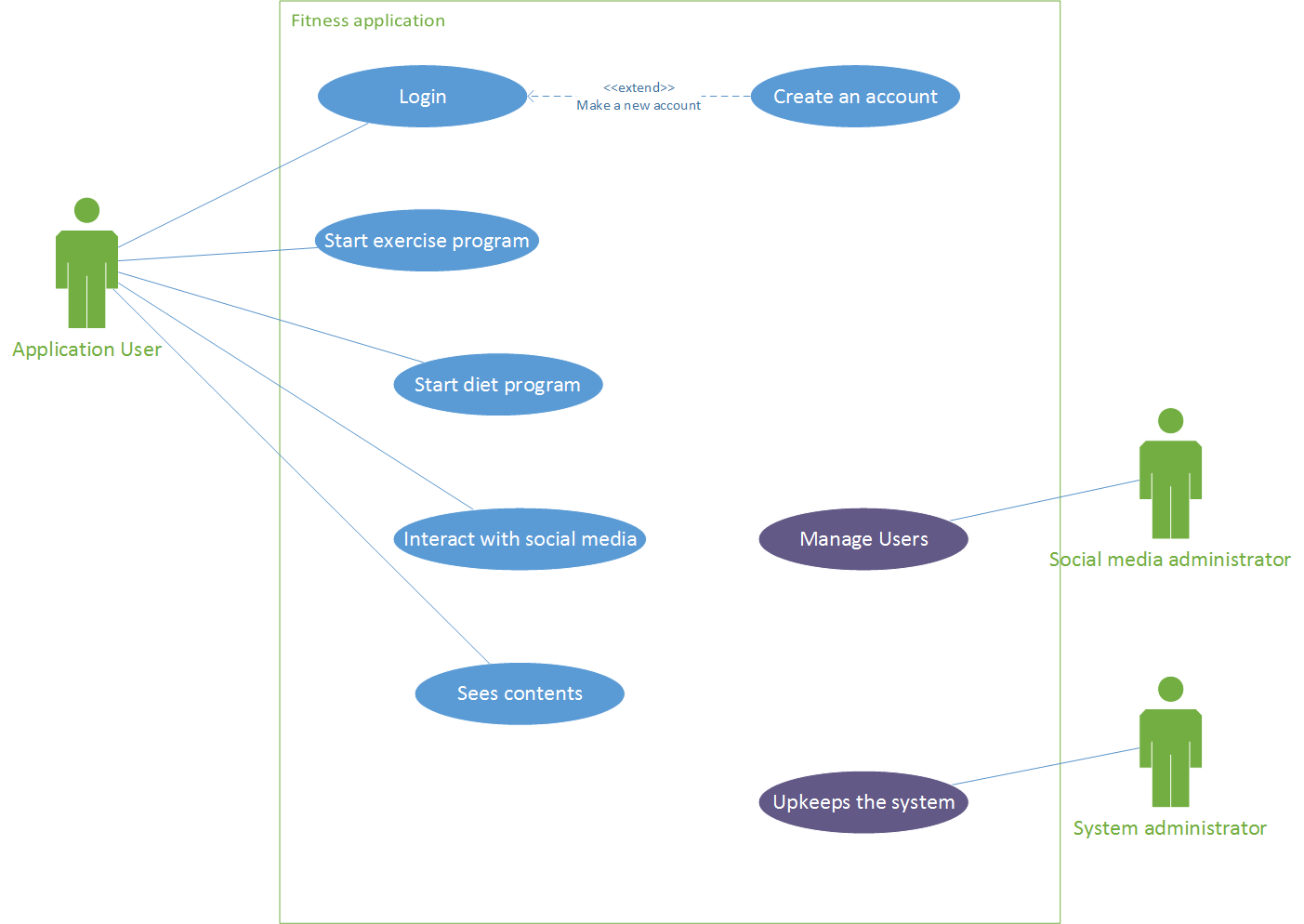
**

Figure 4‑1 Use case diagram

### Use Case List

*UC1.* Do exercise program (User)

* User chooses a kind of exercise program and the intensity of exercise
* User can choose to allow the system to use his information to suggest alternative exercise programs
* User starts exercise
* System measures health status during the exercise
* User finishes exercise and gives info like time spent
* System provides feedback on health and fitness status and saves it (if allowed)

*UC2.* Do diet program (User)

* User inputs information for age, weight, height, sex, activity level, goal weight, and time frame. (optional)
* System saves the information about the user (optional)
* System will provides a daily calorie amount in order to reach the goal weight in the time frame
* User creates or select an appropriate diet program
* User starts the diet program
* System creates a daily planner for the user, as specific as the user wants. Planning individual meals is recommended.

*UC3.* Interact with the social media (User)

* User select the ”Feed/Media” button
* User chooses from the list what content one would like to see
* User see the feed from chosen category
* User interact with the content from the feed
* User select the diet, exercise or both to share
* User push ”Share” to finish

*UC4.* Sees contents (User)

* User accesses database with all content. Separate sections for exercise programs and diet programs.
* User chooses type of content what he wants to see
* System will highlight popular content / up voted shared content
* User chooses specific file to see
* User sees the content of the file
* User can download chosen content
* User exits the database

*UC5.* Manage users (Social media administrator)

* Social media admin checks shared comments and material if everything is politically correct, there is no cybercrime, etc
* Social admin can warn, punish user or delete/block their account depending on what they did

### Use Case Text

**Use Case Text – 1**

**Use case name:** UC1. Do exercise program

**Scope:**

**Level:** User-goal

**Primary Actor:** User, Database

**Stakeholders and Interests:**

* User: wants to choose an exercise program to achieve his attempted goal without any problem.
* Social media application: wants to share or offer exercises for all users who use the application.
* Data base: wants to store/offer all the information from the social media application and the user without missing data and any problem.

**Preconditions:**

* The user has to create an account for the application
* The user has to choose if he want to do his exercise program with or without a diet
* If the user chose to do his exercise with a diet, a diet program must be configured
* The user has to download an exercise program before he starts a predetermined workout

**Postconditions:**

* Confirmation from user that the workout is finished.
* System has stored the information between the user and the Social Media platform frequently and at any time.

**Main Success Scenario 1 (own** **workout constellation):**

1. User chooses that he wants to do an exercise program
2. User chooses that he wants to create his own workout
3. System gives user information about the selectable exercises
4. User selects all exercise he wants to do in the workout
5. User chooses intensity and duration of workout
6. System asks the user if he wants to save his workout
7. User starts exercise
8. System measures health status during the exercise
9. User confirm that he finished a set of his exercise
10. User confirm that he finished his complete workout

**Main Success Scenario 2 (predetermined** **workout):**

1. User chooses that he wants to do an exercise program
2. User chooses that he wants to do a predetermined workout
3. User chooses a workout that he downloads before
4. User chooses intensity and duration of workout
5. System gives user information about his chosen workout
6. User starts exercise
7. System measures health status during the exercise
8. User confirm that he finished a set of his exercise
9. User confirm that he finished his complete workout

**Extensions (Alternative Flows):**

1. In case of system failure in any time
   1. User reboots the system and returns to its former state.
   2. The system is restored to its original state.
2. In case of wanting cancellation of using this system
   1. User terminates the service on the app.
   2. System deletes all data related to the user.
   3. Pop up a message to notify passenger that the system end
3. In case that the app has no connection to the database
   1. The user checks the internet connection of his device
      1. If the connection of the device is bad
         1. The user reconnects to his device
         2. The user reboots the application
      2. If the database cannot be reached
         1. The user reboots the application

**Main 1 (own workout constellation)**

1. If the system doesn’t offer the exercise that the user wants
2. User chooses similar exercise to his request
3. If the system doesn’t save the information
   1. System gives the user an error massage
   2. User checks the connection to his Wi-Fi
4. In case the user wants to change his workout.
5. The user brake off his workout
6. The app asks the user if he really wants to brake off the current workout
7. The database erases the information that was stored for the current workout
8. The user chooses a new workout
9. In case the system don’t save the confirmation
   1. System gives the user an error massage
   2. The user tries to save his data again
      1. In case that it is not able to confirm a finished set
         1. The user skips the confirmation
         2. The user starts with the next exercise
10. In case the system don’t save the confirmation
    1. System gives the user an error massage
    2. The user tries to confirm again
       1. In case that it is not able to confirm a finished set
          1. The user skips the confirmation
          2. The user starts with the next exercise

**Main 2 (predetermined workout)**

1. In case that the system cannot load information for a predetermined workout.
2. User restarts the app
3. User chooses his workout again
4. In case that the system cannot load information for a predetermined workout.
5. User downloads the workout again
6. User starts the workout
7. In case that the system doesn’t provide the information about the workout
8. System gives the user an error massage
9. User downloads the chosen workout again
10. In case the user wants to change his workout.
11. User brakes off his workout
12. System asks the user if he really wants to brake off the current workout
13. Database erases the information that was stored for the current workout
14. User chooses a new workout
15. In case the system don’t save the confirmation
    1. The user confirm his finished exercise again
       1. In case that it is not able to confirm a finished set
          1. The user skips the confirmation
          2. The user starts with the next exercise
16. In case the system don’t save the confirmation
    1. The user confirm his finished workout again
       1. In case that it is not able to confirm a finished set
          1. The user skips the confirmation
          2. The user starts with the next exercise

**Special Requirements:**

* Aegle hinges on there being an active user base to drive content
* In the early stages of Aegle we will provide some workouts ideas for a balanced exercise set
* User should be able to easily adjust own workouts in order to create own exercises

**Technology and Data variations list:**

The System is optimized for Tizen environment.

**Frequency of Occurrence:**

Always.

**Open issues:**

None.

**Use Case Text – 4**

**Use case name:** UC4. Do exercise program

**Use case name:** Interact with the social media (User)

**Scope:** ???

**Level:** User goal

**Primary Actor:** User, social media application, Database

**Stakeholders and Interests:**

* User: wants to browser and interact with ”Feed/Media”
* Social media application: wants to share and/or offer new/popular content
* Data base: wants to store/offer all the information from the social media application and the user

**Preconditions:**

* The user has to create an account for the app
* The user has to choose from the list what content one would like to see

**Postconditions:**

* The database has to store the information about user’s interaction frequently

**Main Success Scenario 1:**

1. User choose ”Feed/Media” button
2. User choose popular content
3. User sees popular content
4. User interact with content

**Main success Scenario 2:**

1. User choose ”Feed/Media” button
2. User choose new content
3. User sees new content
4. User interact with content

**Extensions (Alternative Flows):**

1. In case the app doesn’t work correct
   1. The user stops the app and reboot the system
   2. The user starts again and return to his current plan
2. In case the user doesn’t have connection to the database
   1. The user checks the internet connection of his device
3. If the connection of the device is bad
4. The user reconnects to his device
5. The user reboots the application
6. In case the content isn’t available that user would like to interact
7. The user refresh the ”Feed/Media”
8. The user tries to load the content again
9. In case the content isn’t available that user would like to interact
10. The user reboot the app
11. The user tries to load the content again
12. In case the content isn’t available that user would like to interact
13. The user refresh internet connection
14. The user tries to load the content again
15. User report dysfunctions/inappropriate content
16. User experience some bugs/dysfunction/inappropriate content
17. User reports about these problems

**Special Requirements:**

* Touch screen UI on the mobile device. Text size and colors can be adjustable for user preference (e.g. bad sight, color blindness, etc…)
* Language internationalization on the text displayed

**Frequency of Occurrence:**

* Up to user. User can decide when he wants to browser the ”Feed/Media”

**Open Issues:**

* Is the uploaded content appropriate?

## Non-functional Requirements

### Introduction

This section explains all of the requirements not expressed in the use cases

### Functionality

Security

* If the user starts a new training program and didn’t agree the security policy there will be a popup-window to ask the user about his security settings

Safety

* The user should keep his devices on which he uses the app updated in order to avoid the loss of his local stored personal data

### Usability

Connection error

* If the user isn’t connected to the internet there will be a sign that signalizes that some services are blocked until the user connect again to the internet

Simplicity

* The use of this app on the watch shall be intuitive. The buttons are clearly labeled and the user can easily navigate through the different features of the app. Users aged from 12 to 80 should be able to use the app without any major problems

Regional requirements

* The app is designated for the Korean market. Although the displayed text is provided in English and payments shall be done in USD to enable foreigners living in Korea to use the app too

### Reliability

Availability

* The app should work properly in 99.99% of used times. A restart of the app should be the main action which can fix any occured problems during using the app.

Health safety

* The app should provide exercise programs and diet recipes on high quality. The feedback feature and certified users shall provide that the chance of damaging the user’s health is minimized.

Changeability

* Changes to the app are done through updates from the system administrator.

### Purchased components

In-app purchases possible

* The user can unlock certified diet recipes and exercise programs with money.

### Implementation Constraints

* Need a significant amount of content in the beginning to attract early users.

# Domain Model

## Domain Model Diagram

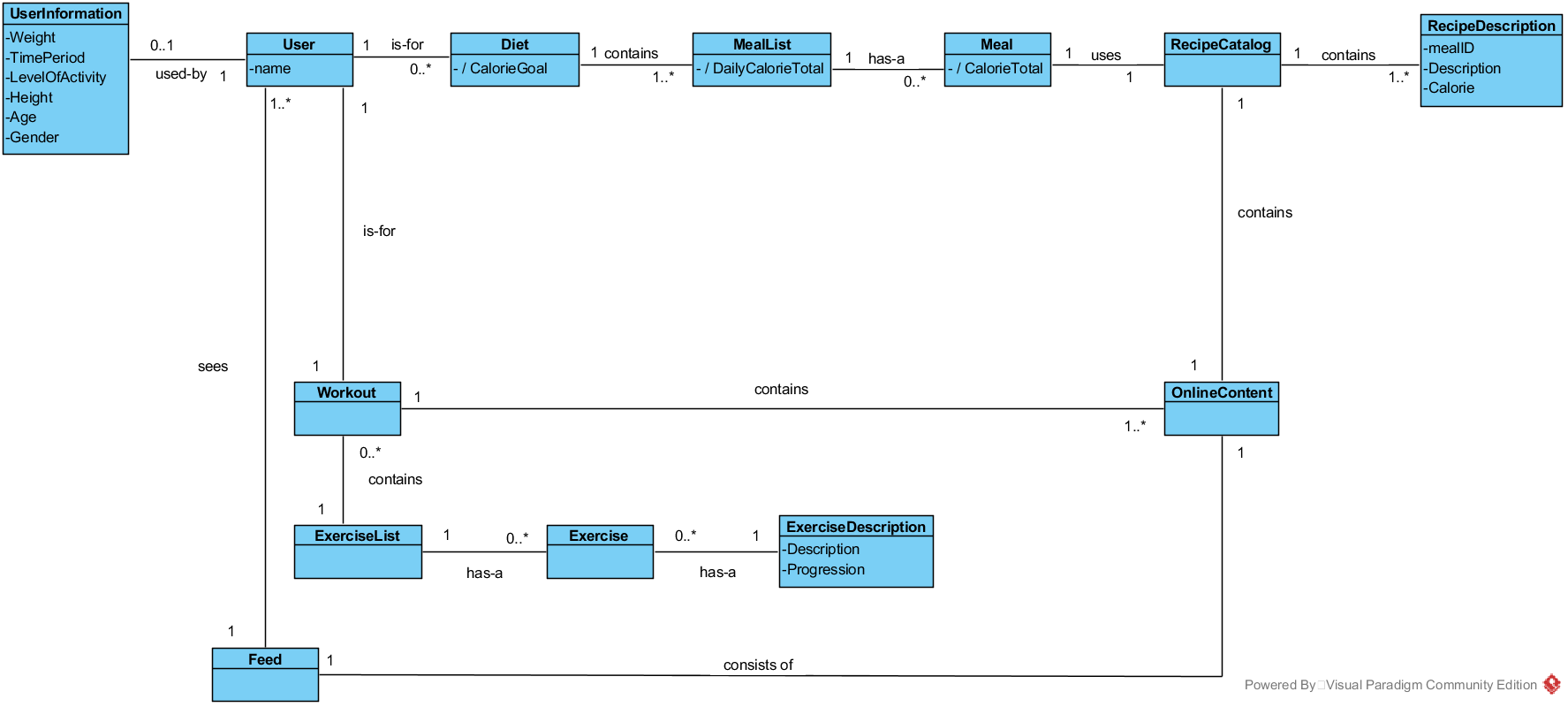


Figure 5‑1 Domain Class Diagram

Figure 5-1 shows the conceptual model of the howl system, using associations between the classes what represents the real world problem.

### Domain classes

Table 5‑1 description for domain class User

|  |  |
| --- | --- |
| **User** | |
| **Attributes** | name |
| **Association** | * One user makes one diet * One user makes one workout * One or more user uses one feed |

Table 5‑2 description for domain class Diet

|  |  |
| --- | --- |
| **Diet** | |
| **Attributes** | CalorieGoal |
| **Association** | * One diet is made by one user * One diet contains a meal list * One diet uses user information (optional) |

Table 5‑3 description for domain class UserInformation

|  |  |
| --- | --- |
| **UserInformation** | |
| **Attributes** | Weight  TimePeriod  LevelOfActivty  Height  Age  Gender |
| **Association** | * User information is contained by one diet |

Table 5‑4 description for domain class MealList

|  |  |
| --- | --- |
| **MealList** | |
| **Attributes** | DailyCalorieTotal |
| **Association** | * One or more meal lists are contained by one diet * One meal list contains one or more meals |

Table 5‑5 description for domain class Meal

|  |  |
| --- | --- |
| **Meal** | |
| **Attributes** | CalorieTotal |
| **Association** | * One or more meals are contained by one meal list * One meal uses a recipe catalog |

Table 5‑6 description for domain class RecipeCatalog

|  |  |
| --- | --- |
| **RecipeCatalog** | |
| **Attributes** | - |
| **Association** | One recipe catalog contains one or more recipe descriptions |

Table 5‑7 description for domain class RecipeDescription

|  |  |
| --- | --- |
| **RecipeDescription** | |
| **Attributes** | mealID  Description  Calories |
| **Association** | One or more recipe descriptions are contained by one recipe catalog |

Table 5‑8 description for domain class Workout

|  |  |
| --- | --- |
| **Workout** | |
| **Attributes** | - |
| **Association** | * One workout is created by one user * One workout contains one exercise list |

Table 5‑9 description for domain class Workout

|  |  |
| --- | --- |
| **ExerciseList** | |
| **Attributes** | - |
| **Association** | * One exercises list is contains one or more exercises * One workout contained one exercise list |

Table 5‑10 description for domain class Workout

|  |  |
| --- | --- |
| **Exercise** | |
| **Attributes** | - |
| **Association** | * One exercise contains one exercise description * One or more exercises are contained by one exercise list |

Table 5‑11 description for domain class ExerciseDescription

|  |  |
| --- | --- |
| **ExerciseDescription** | |
| **Attributes** | - |
| **Association** | * One exercise description are contained by one exercise |

Table 5‑12 description for domain class OnlineContent

|  |  |
| --- | --- |
| **OnlineContent** | |
| **Attributes** | - |
| **Association** | * One online content contains one workout * One online content contains one recipe catalog |

Table 5‑13 description for domain class Feed

|  |  |
| --- | --- |
| **Feed** | |
| **Attributes** | - |
| **Association** | * One feed is used by one or more users * One feed uses one online content |

# System Sequence Diagram

## UC1 – Do exercise program

### UC1 Main 1 – SSD

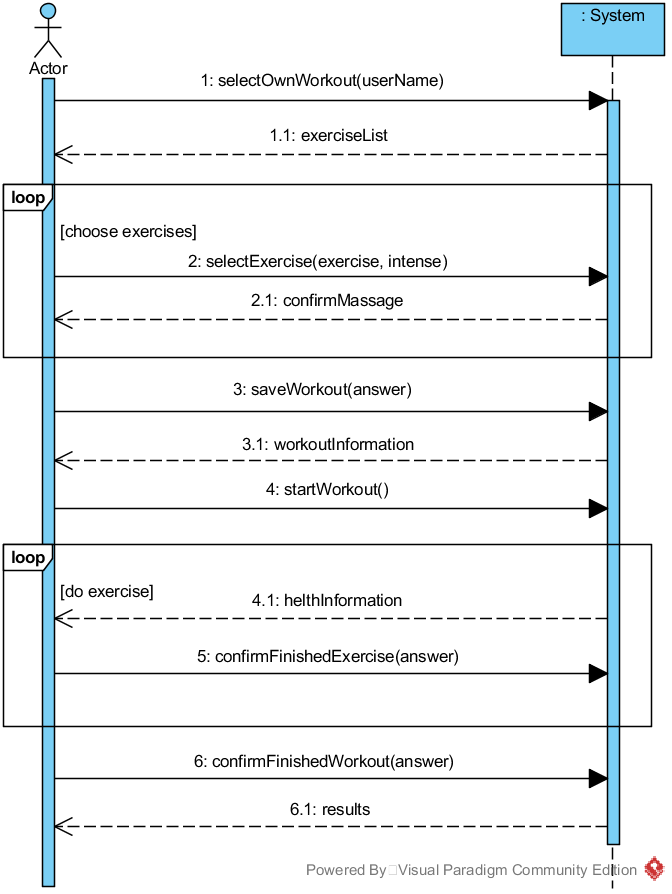


Figure 6‑1 UC1 SSD, Main1

### UC1 Main 1– Operation Contracts

Table 6‑1 Main 1 - selectOwnWorkout(username : String)

|  |  |
| --- | --- |
| **Operation:** | selectOwnWorkout(username : String) |
| **Cross references:** | Do exercise program |
| **Preconditions:** | * none |
| **Postconditions:** | * A OwnWorkout instance ow was created * ow was associated with a User * Attribute ow.name was initialized with userName |

Table 6‑2 Main 1 - selectExercise(exercise : Exercise, intense : Intense)

|  |  |
| --- | --- |
| **Operation:** | selectExercise(exercise : Exercise, intense : Intense) |
| **Cross references:** | Do exercise program |
| **Preconditions:** | * There is a own workout is underway |
| **Postconditions:** | * Attribute exercise.intense was initialized with intense * Attribute exercise was added to exerciseList * Workout was associated with ExerciseCatalog |

Table 6‑3 Main 1 - saveWorkout(answer : int)

|  |  |
| --- | --- |
| **Operation:** | saveWorkout(answer : int) |
| **Cross references:** | Do exercise program |
| **Preconditions:** | * There is a own workout is underway |
| **Postconditions:** | * Attribute userWorkout was initialized with current workout * User was associated with Workout |

Table 6‑4 Main 1 - startWorkout()

|  |  |
| --- | --- |
| **Operation:** | startWorkout() |
| **Cross references:** | Do exercise program |
| **Preconditions:** | * There is a own workout is underway |
| **Postconditions:** | * Attribute workout.status became true |

Table 6‑5 Main 1 - confirmFinishedExercise(answer : boolean)

|  |  |
| --- | --- |
| **Operation:** | confirmFinishedExercise(answer : boolean) |
| **Cross references:** | Do exercise program |
| **Preconditions:** | * There is a own workout is underway |
| **Postconditions:** | * Attribute workout.exercise became true |

Table 6‑6 Main 1 - confirmFinishedWorkout(answer : boolean)

|  |  |
| --- | --- |
| **Operation:** | confirmFinishedWorkout(answer : boolean) |
| **Cross references:** | Do exercise program |
| **Preconditions:** | * There is a own workout is underway |
| **Postconditions:** | * Attribute workout.status became false |

### UC1 Main 2 – SSD

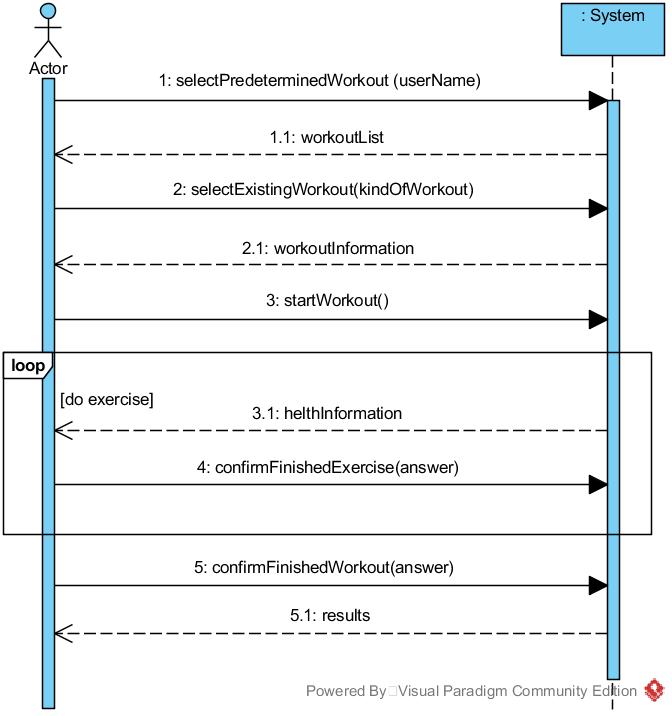


Figure 6‑2 UC1 SSD, Main2

### UC1 Main 2 – Operation Contracts

Table 6‑7 Main 2 - selectPrederterminedWorkout(username : String)

|  |  |
| --- | --- |
| **Operation:** | selectPrederterminedWorkout(username : String) |
| **Cross references:** | Do exercise program |
| **Preconditions:** | none |
| **Postconditions:** | * A PrederterminedWorkout instance pw was created * pw was associated with User * Attribute pw.name was initialized with userName |

Table 6‑8 Main 2 - selectExistingWorkout(exerciseList : List<Exercise>)

|  |  |
| --- | --- |
| **Operation:** | selectExistingWorkout(exerciseList : List<Exercise>) |
| **Cross references:** | Do exercise program |
| **Preconditions:** | * There is a predetermined workout underway |
| **Postconditions:** | * Attribute exerciseList was initialized with exerciseList * Workout was associated with OnlineContent |

Table 6‑9 Main 2 - startWorkout()

|  |  |
| --- | --- |
| **Operation:** | startWorkout() |
| **Cross references:** | Do exercise program |
| **Preconditions:** | * There is a predetermined workout underway |
| **Postconditions:** | * Attribute workout.status became true |

Table 6‑10 Main 2 - confirmFinishedExercise(answer : boolean)

|  |  |
| --- | --- |
| **Operation:** | confirmFinishedExercise(answer : boolean) |
| **Cross references:** | Do exercise program |
| **Preconditions:** | * There is a predetermined workout underway |
| **Postconditions:** | * Attribute workout.exercise became true |

Table 6‑11 Main 2 - confirmFinishedWorkout(answer : boolean)

|  |  |
| --- | --- |
| **Operation:** | confirmFinishedWorkout(answer : boolean) |
| **Cross references:** | Do exercise program |
| **Preconditions:** | * There is a predetermined workout underway |
| **Postconditions:** | * Attribute workout.status became false |

## UC2 – Do diet program

### UC2 – SSD

### UC2 – Operation Contracts

## UC3 – Interact with the social media

### UC3 – SSD

### UC3 – Operation Contracts

## UC4 – Sees contents

### UC4 – SSD

### UC4 – Operation Contracts

## UC5 – Manage users

### UC5 – SSD

### UC5 – Operation Contracts

# Design Model

## Do Exercise Program Realization

### Main1 - Operation 1

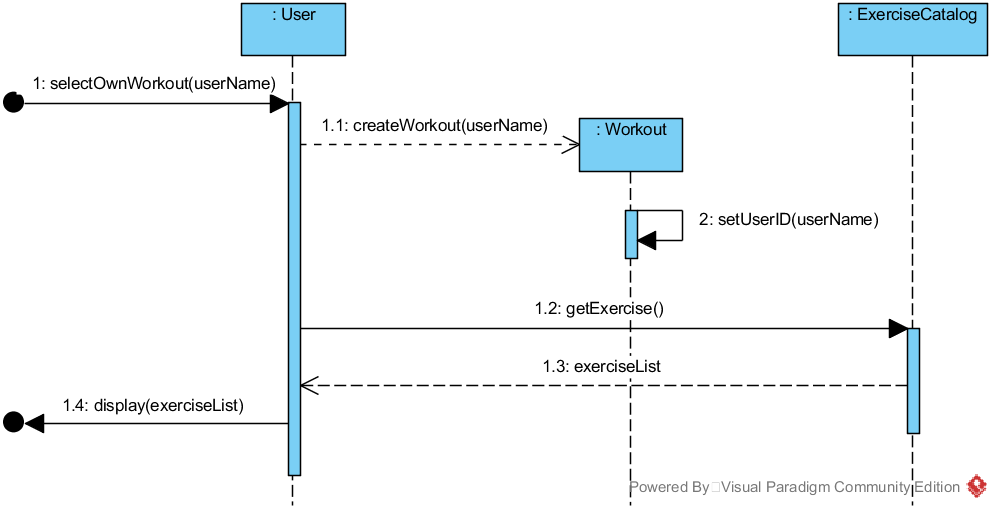


Figure 7‑1 Main1 Sequence Diagram for Operation 1

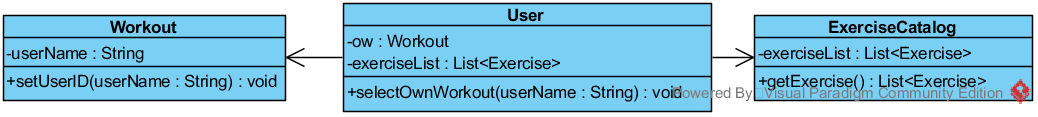


Figure 7‑2 Main1 Class Diagram Operation 1

Table 7‑1 GRASP Main1 Operation 1

|  |  |
| --- | --- |
| **Creator** | * User creates a Workout |
| **Information Expert** | * User knows username * ExerciseCatalog knows a list of exercises (exerciseList) |
| **Low Coupling** | * User does not need to know how to bring exerciseList |
| **High Cohesion** | * Workout has only one responsibility. Set name of user. |
| **Controller** | * User represents a handler of all system events, used by user. |

### Main1 - Operation 2

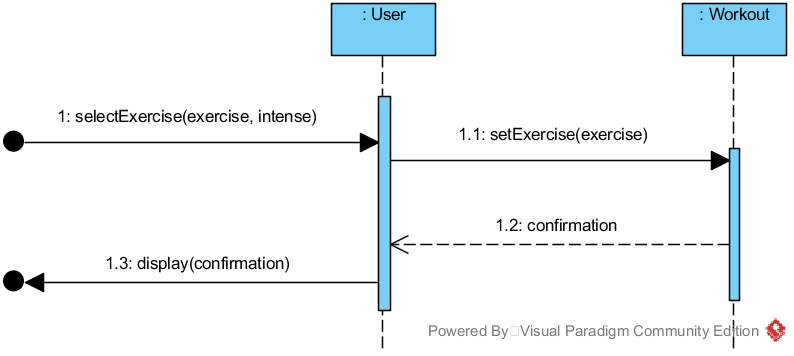


Figure 7‑3 Main1 Sequence Diagram for Operation 2

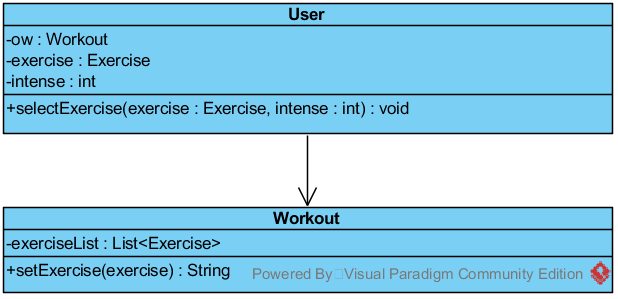


Figure 7‑4 Main1 Class Diagram Operation 2

Table 7‑2 GRASP Main1 Operation 2

|  |  |
| --- | --- |
| **Creator** | * None |
| **Information Expert** | * User knows exercise and intense * Workout knows confirmation |
| **Low Coupling** | * User doesn’t need to know how to set exercise in workout |
| **High Cohesion** | * User has only one responsibility. Set exercise. |
| **Controller** | * User represents a handler of all system events, used by user. |

### Main1 - Operation 3

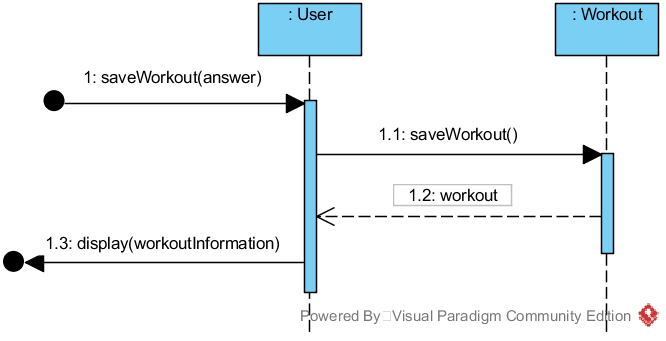


Figure 7‑5 Main1 Sequence Diagram for Operation 3

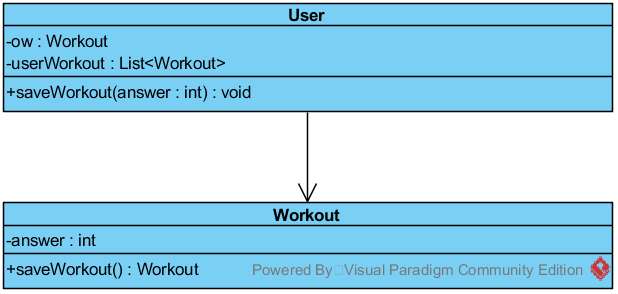


Figure 7‑6 Main1 Class Diagram Operation 3

Table 7‑3 GRASP Main1 Operation 3

|  |  |
| --- | --- |
| **Creator** | * None |
| **Information Expert** | * User knows answer * Workout knows workout |
| **Low Coupling** | * User doesn’t need to know how to save workout |
| **High Cohesion** | * User has only one responsibility. Save workout. |
| **Controller** | * User represents a handler of all system events, used by user. |

### Main1 - Operation 4

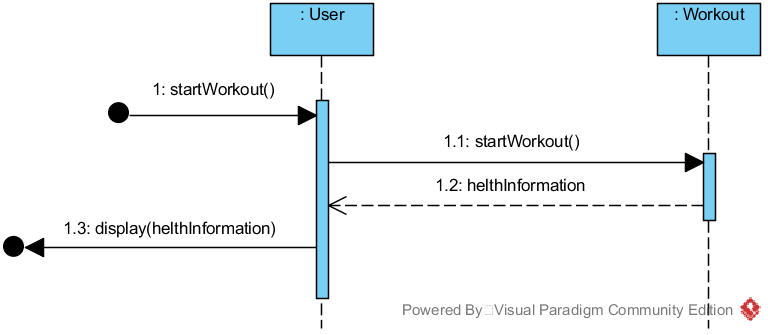


Figure 7‑7 Main1 Sequence Diagram for Operation 4

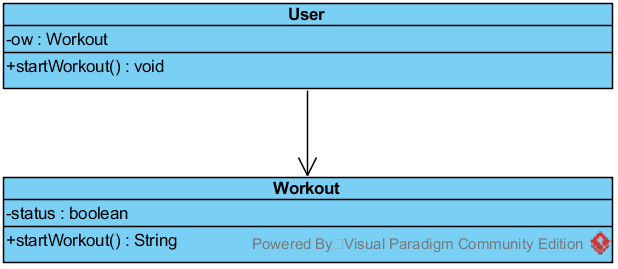


Figure 7‑8 Main1 Class Diagram Operation 4

Table 7‑4 GRASP Main1 Operation 4

|  |  |
| --- | --- |
| **Creator** | * None |
| **Information Expert** | * Workout knows healthInformation |
| **Low Coupling** | * User doesn’t need to know how to start workout |
| **High Cohesion** | * User has only one responsibility. Start workout. |
| **Controller** | * User represents a handler of all system events, used by user. |

### Main1 - Operation 5

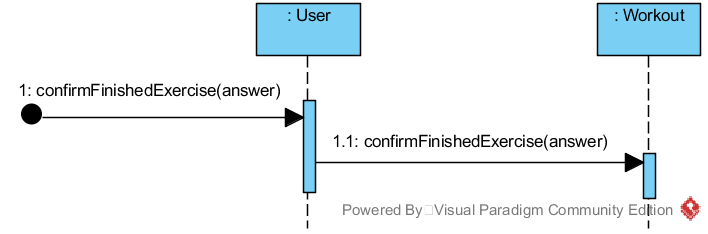


Figure 7‑9 Main1 Sequence Diagram for Operation 5

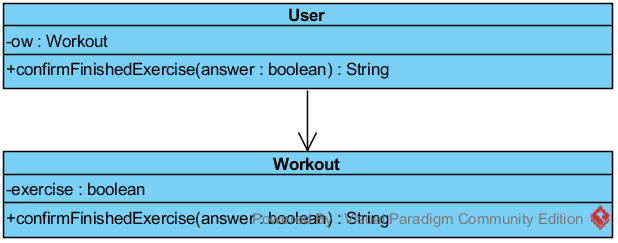


Figure 7‑10 Main1 Class Diagram Operation 5

Table 7‑5 GRASP Main1 Operation 5

|  |  |
| --- | --- |
| **Creator** | * None |
| **Information Expert** | * User knows answer |
| **Low Coupling** | * User doesn’t need to know how to confirm a finished exercise |
| **High Cohesion** | * User hast only one responsibility. Confirm a finished exercise. |
| **Controller** | * User represents a handler of all system events, used by user. |

### Main1 - Operation 6

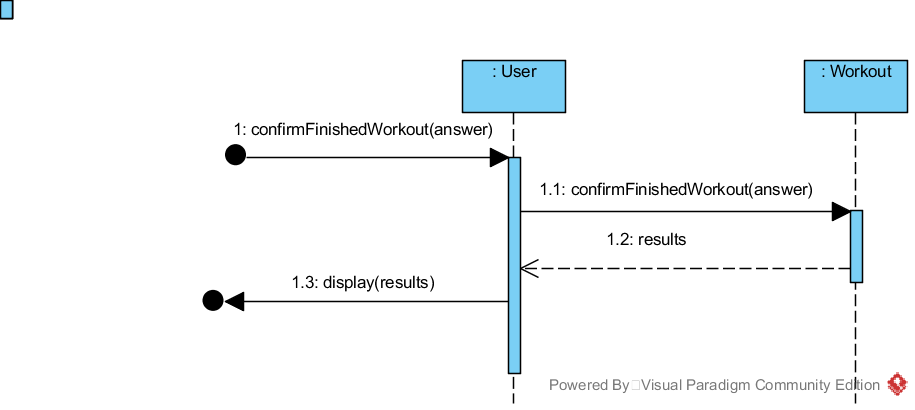


Figure 7‑11 Main1 Sequence Diagram for Operation 6

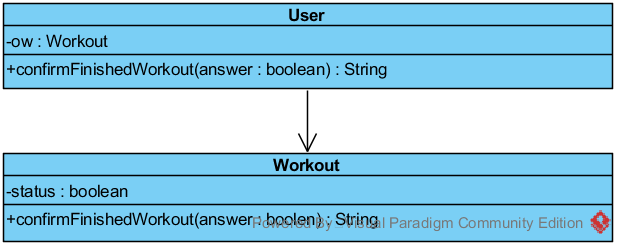


Figure 7‑12 Main1 Class Diagram Operation 6

Table 7‑6 GRASP Main1 Operation 6

|  |  |
| --- | --- |
| **Creator** | * None |
| **Information Expert** | * User knows answer * Workout knows result |
| **Low Coupling** | * User doesn’t need to know how to confirm a finished workout |
| **High Cohesion** | * User has only one responsibility. Confirm a finished workout. |
| **Controller** | * User represents a handler of all system events, used by user. |

### Main2 - Operation 1

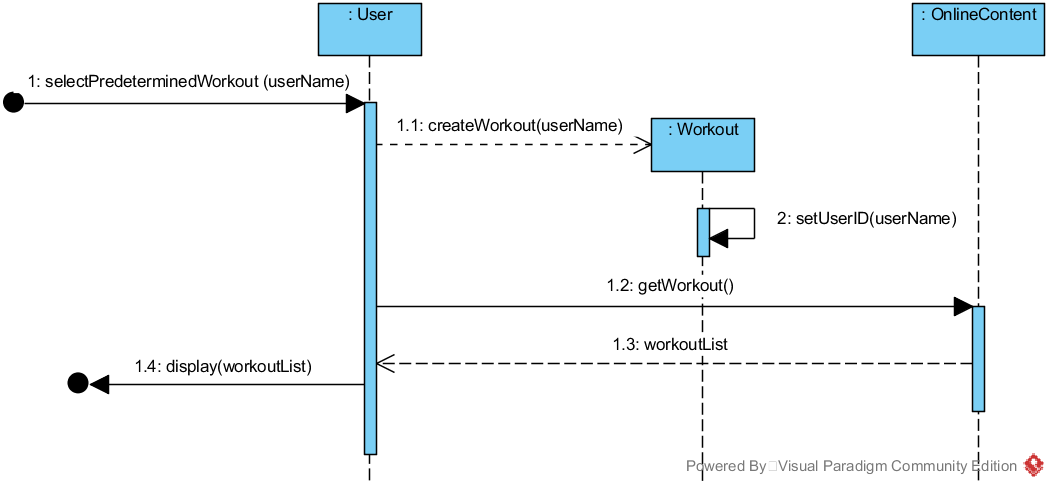


Figure 7‑13 Main2 Sequence Diagram for Operation 1

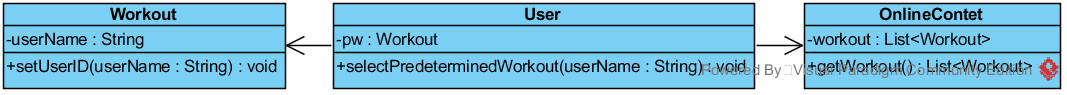


Figure 7‑14 Main2 Class Diagram Operation 1

Table 7‑7 GRASP Main2 Operation 1

|  |  |
| --- | --- |
| **Creator** | * User creates a Workout |
| **Information Expert** | * User knows user name * OnlineContent knows workoutList |
| **Low Coupling** | * User doesn’t need to know how to get a workout |
| **High Cohesion** | * Workout has only one responsibility. Set user name. |
| **Controller** | * User represents a handler of all system events, used by user. |

### Main2 - Operation 2

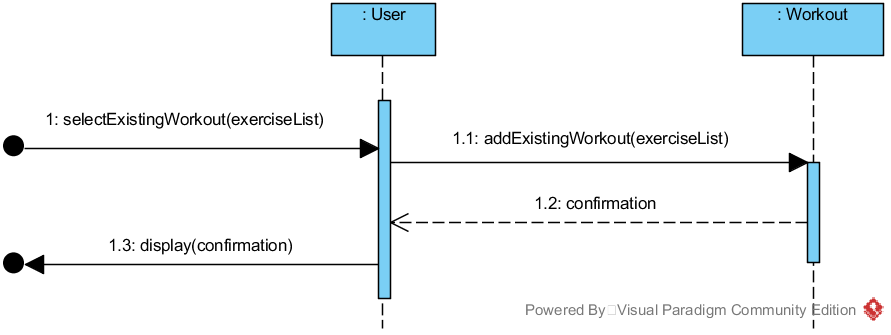


Figure 7‑15 Main2 Sequence Diagram for Operation 2

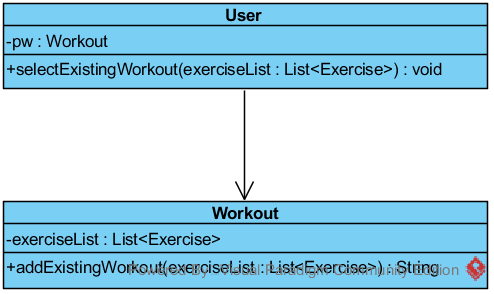


Figure 7‑16 Main2 Class Diagram Operation 2

Table 7‑8 GRASP Main2 Operation 2

|  |  |
| --- | --- |
| **Creator** | * None |
| **Information Expert** | * User knows exerciseList * Workout knows confirmation |
| **Low Coupling** | * User doesn’t need to know how to add an existing workout |
| **High Cohesion** | * User has only one responsibility. Add an existing workout |
| **Controller** | * User represents a handler of all system events, used by user. |

### Main2 - Operation 3

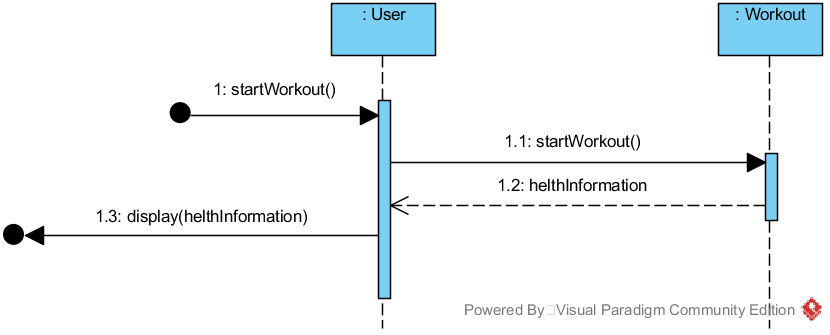


Figure 7‑17 Main2 Sequence Diagram for Operation 3

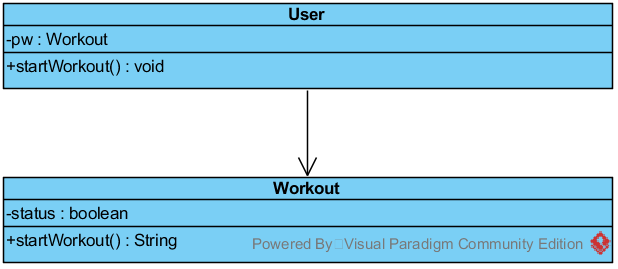


Figure 7‑18 Main2 Class Diagram Operation 3

Table 7‑9 GRASP Main2 Operation 3

|  |  |
| --- | --- |
| **Creator** | * None |
| **Information Expert** | * Workout knows helthInformation |
| **Low Coupling** | * User doesn’t need to know how to start a workout |
| **High Cohesion** | * User has only one responsibility. Start a workout. |
| **Controller** | * User represents a handler of all system events, used by user. |

### Main2 - Operation 4

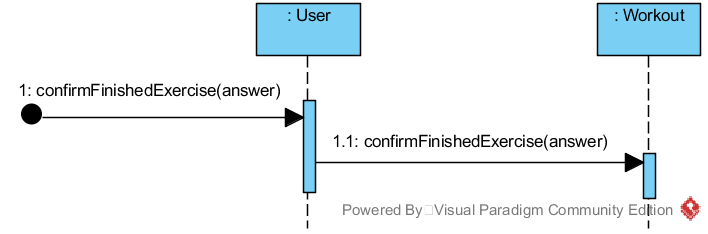


Figure 7‑19 Main2 Sequence Diagram for Operation 4

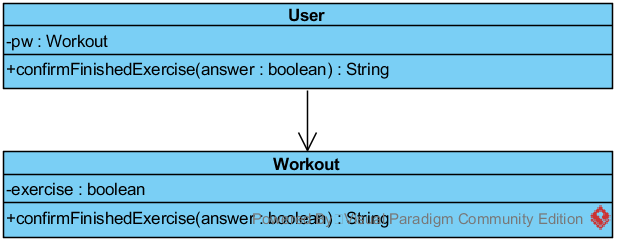


Figure 7‑20 Main2 Class Diagram Operation 4

Table 7‑10 GRASP Main2 Operation 4

|  |  |
| --- | --- |
| **Creator** | * None |
| **Information Expert** | * User knows answer |
| **Low Coupling** | * User doesn’t need to know how to confirm a finished exercise |
| **High Cohesion** | * User hast only one responsibility. Confirm a finished exercise. |
| **Controller** | * User represents a handler of all system events, used by user. |

### Main2 - Operation 5

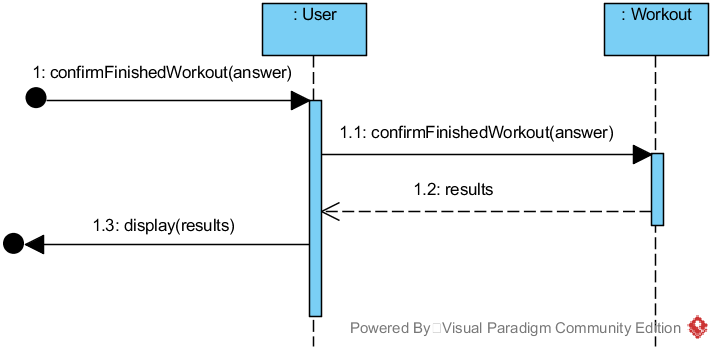


Figure 7‑21 Main2 Sequence Diagram for Operation 5

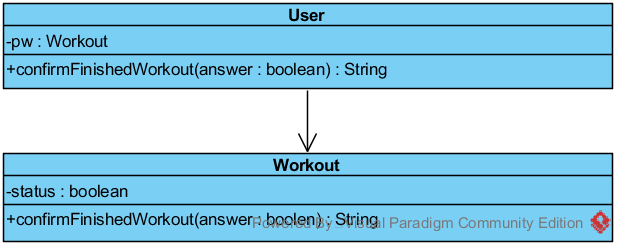


Figure 7‑22 Main2 Class Diagram Operation 5

Table 7‑11 GRASP Main2 Operation 5

|  |  |
| --- | --- |
| **Creator** | * None |
| **Information Expert** | * User knows answer * Workout knows result |
| **Low Coupling** | * User doesn’t need to know how to confirm a finished workout |
| **High Cohesion** | * User has only one responsibility. Confirm a finished workout. |
| **Controller** | * User represents a handler of all system events, used by user. |

## Design Class Diagram

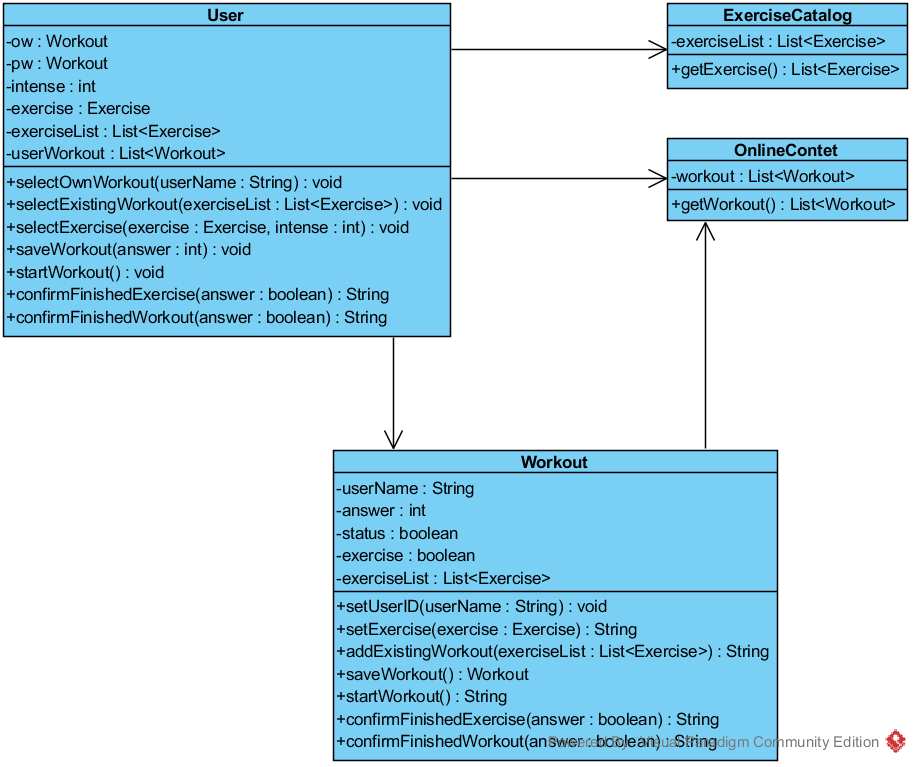


Figure 7‑23 Design Class Diagram

# References

* Larman, Craig. Applying UML and Patterns: An Introduction to Object-oriented Analysis and Design and Iterative Development. Upper Saddle River, NJ: Prentice Hall PTR, 2010. Print.
* Smit, Ellen. "News and Research Communications." *U.S. Adults Get failing Grade in Healthy Lifestyle Behavior*. Oregon State University, 21 Mar. 2016. Web. 23 Mar. 2016. <http://oregonstate.edu/ua/ncs/archives/2016/mar/us-adults-get-failing-grade-healthy-lifestyle-behavior>.

# Content of Figures

[Figure 4‑1 Use case diagram 7](#_Toc450455973)

[Figure 5‑1 Domain Class Diagram 18](#_Toc450455974)

[Figure 6‑1 UC1 SSD, Main1 21](#_Toc450455975)

[Figure 6‑2 UC1 SSD, Main2 23](#_Toc450455976)

[Figure 7‑1 Main1 Sequence Diagram for Operation 1 29](#_Toc450455977)

[Figure 7‑2 Main1 Class Diagram Operation 1 29](#_Toc450455978)

[Figure 7‑3 Main1 Sequence Diagram for Operation 2 30](#_Toc450455979)

[Figure 7‑4 Main1 Class Diagram Operation 2 30](#_Toc450455980)

[Figure 7‑5 Main1 Sequence Diagram for Operation 3 31](#_Toc450455981)

[Figure 7‑6 Main1 Class Diagram Operation 3 31](#_Toc450455982)

[Figure 7‑7 Main1 Sequence Diagram for Operation 4 32](#_Toc450455983)

[Figure 7‑8 Main1 Class Diagram Operation 4 32](#_Toc450455984)

[Figure 7‑9 Main1 Sequence Diagram for Operation 5 33](#_Toc450455985)

[Figure 7‑10 Main1 Class Diagram Operation 5 33](#_Toc450455986)

[Figure 7‑11 Main1 Sequence Diagram for Operation 6 34](#_Toc450455987)

[Figure 7‑12 Main1 Class Diagram Operation 6 34](#_Toc450455988)

[Figure 7‑13 Main2 Sequence Diagram for Operation 1 35](#_Toc450455989)

[Figure 7‑14 Main2 Class Diagram Operation 1 35](#_Toc450455990)

[Figure 7‑15 Main2 Sequence Diagram for Operation 2 36](#_Toc450455991)

[Figure 7‑16 Main2 Class Diagram Operation 2 36](#_Toc450455992)

[Figure 7‑17 Main2 Sequence Diagram for Operation 3 37](#_Toc450455993)

[Figure 7‑18 Main2 Class Diagram Operation 3 37](#_Toc450455994)

[Figure 7‑19 Main2 Sequence Diagram for Operation 4 38](#_Toc450455995)

[Figure 7‑20 Main2 Class Diagram Operation 4 38](#_Toc450455996)

[Figure 7‑21 Main2 Sequence Diagram for Operation 5 39](#_Toc450455997)

[Figure 7‑22 Main2 Class Diagram Operation 5 39](#_Toc450455998)

[Figure 7‑23 Design Class Diagram 40](#_Toc450455999)

# Content of Tables

[Table 2‑1 Key High-level Goals 4](#_Toc450455936)

[Table 2‑2 Summary of Benefits 5](#_Toc450455937)

[Table 5‑1 description for domain class User 18](#_Toc450455938)

[Table 5‑2 description for domain class Diet 18](#_Toc450455939)

[Table 5‑3 description for domain class UserInformation 19](#_Toc450455940)

[Table 5‑4 description for domain class MealList 19](#_Toc450455941)

[Table 5‑5 description for domain class Meal 19](#_Toc450455942)

[Table 5‑6 description for domain class RecipeCatalog 19](#_Toc450455943)

[Table 5‑7 description for domain class RecipeDescription 19](#_Toc450455944)

[Table 5‑8 description for domain class Workout 20](#_Toc450455945)

[Table 5‑9 description for domain class Workout 20](#_Toc450455946)

[Table 5‑10 description for domain class Workout 20](#_Toc450455947)

[Table 5‑11 description for domain class ExerciseDescription 20](#_Toc450455948)

[Table 5‑12 description for domain class OnlineContent 20](#_Toc450455949)

[Table 5‑13 description for domain class Feed 20](#_Toc450455950)

[Table 6‑1 Main 1 - selectOwnWorkout(username : String) 22](#_Toc450455951)

[Table 6‑2 Main 1 - selectExercise(exercise : Exercise, intense : Intense) 22](#_Toc450455952)

[Table 6‑3 Main 1 - saveWorkout(answer : int) 22](#_Toc450455953)

[Table 6‑4 Main 1 - startWorkout() 22](#_Toc450455954)

[Table 6‑5 Main 1 - confirmFinishedExercise(answer : boolean) 22](#_Toc450455955)

[Table 6‑6 Main 1 - confirmFinishedWorkout(answer : boolean) 23](#_Toc450455956)

[Table 6‑7 Main 2 - selectPrederterminedWorkout(username : String) 23](#_Toc450455957)

[Table 6‑8 Main 2 - selectExistingWorkout(exerciseList : List<Exercise>) 24](#_Toc450455958)

[Table 6‑9 Main 2 - startWorkout() 24](#_Toc450455959)

[Table 6‑10 Main 2 - confirmFinishedExercise(answer : boolean) 24](#_Toc450455960)

[Table 6‑11 Main 2 - confirmFinishedWorkout(answer : boolean) 24](#_Toc450455961)

[Table 7‑1 GRASP Main1 Operation 1 29](#_Toc450455962)

[Table 7‑2 GRASP Main1 Operation 2 30](#_Toc450455963)

[Table 7‑3 GRASP Main1 Operation 3 31](#_Toc450455964)

[Table 7‑4 GRASP Main1 Operation 4 32](#_Toc450455965)

[Table 7‑5 GRASP Main1 Operation 5 33](#_Toc450455966)

[Table 7‑6 GRASP Main1 Operation 6 34](#_Toc450455967)

[Table 7‑7 GRASP Main2 Operation 1 35](#_Toc450455968)

[Table 7‑8 GRASP Main2 Operation 2 36](#_Toc450455969)

[Table 7‑9 GRASP Main2 Operation 3 37](#_Toc450455970)

[Table 7‑10 GRASP Main2 Operation 4 38](#_Toc450455971)

[Table 7‑11 GRASP Main2 Operation 5 39](#_Toc450455972)

Use Case Text - 2 Fully Dressed

Use case name: Select a diet program

Level: User goal

Primary Actor: User, social media application, Database

Stakeholders and Interests

* User: wants to get a balanced program with exercise and/or diet program
* Social media application: wants to share or offer exercise and/or diet plans
* Data base: wants to store/offer all the information from the social media application and the user

Preconditions:

* The user has to create an account for the app
* The user has to choose if he want to do his diet with or without an exercise program
* The user has to choose if he wants to download his diet program form the social media platform or not

Postconditions:

* The database has to store the information frequently

Main Success Scenario 1:

* User inputs his information and receives a daily calorie target.
* User uses food database to create a meal plan that is nutritious and delicious
* User can share their meal plan to the social media to help others in a similar situation

Main Success Scenario 2:

* User inputs his information and receives a daily calorie target
* User finds and downloads a meal plan that meets his target
* User leaves a review for the meal plan helping others find the plan as well

Extensions (Alternative Flows):

\*a. In case the app doesn’t work correct

1. The user stops the app and reboot the system
2. The user starts again and return to his current plan

1. In case the user wants to change his diet plan

1. The user deletes the current plan
2. The database erases the information that are stored
3. The user creates a new plan

2. In case the user doesn’t have connection to the database

1. The user checks the internet connection of his device

1a. If the connection of the device is bad

1. The user reconnects to his device

2. The user reboots the application

3a. In case the diet plan that user wants to load isn’t in the database

1. The user make his own diet plan
2. The user load his own diet plan

3b. In case the diet plan that user wants to load isn’t in the database

1. The user downloads the plan from the social media platform
2. The user loads the downloaded diet plan

4. In case the user doesn’t complete his daily program

1. The database stores the information
2. The application sends the information to the user

Special Requirements:

* Aegle hinges on there being an active user base to drive content
* In the early stages of Aegle we will provide some meal plans and recipe ideas for various calorie targets
* User should be able to easily adjust meal plans in order to create variation

Frequency of Occurrence:

* Very frequent. The diet program is the biggest part of Aegle. Users are expected to frequently check their meal plans, make changes, and to try new meal plans recommended by others