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

Elaboration1 Report

9 Mai 2016

Domain Analysis & Design

Table of Contents

[2 Vision 1](#_Toc450525048)

[2.1 Introduction 1](#_Toc450525049)

[2.2 Positioning 2](#_Toc450525050)

[2.2.1 Business Opportunity 2](#_Toc450525051)

[2.2.2 Problem Statement 2](#_Toc450525052)

[2.2.3 Product Position Statement 2](#_Toc450525053)

[2.2.4 Alternatives and Competition 3](#_Toc450525054)

[2.3 Stakeholder Description 3](#_Toc450525055)

[2.3.1 Stakeholder Summary 3](#_Toc450525056)

[2.3.2 User Summary 3](#_Toc450525057)

[2.3.3 Key High-level Goals and Problems of the Stakeholders 3](#_Toc450525058)

[2.3.4 User Goals 4](#_Toc450525059)

[2.3.5 User Environment 4](#_Toc450525060)

[2.4 Product Overview 5](#_Toc450525061)

[2.4.1 Product Perspective 5](#_Toc450525062)

[2.4.2 Summary of Benefits 5](#_Toc450525063)

[2.4.3 Assumptions and Dependencies 5](#_Toc450525064)

[2.4.4 Other Requirements and Constraints 6](#_Toc450525065)

[3 Requirements 7](#_Toc450525066)

[3.1 Functional Requirements 7](#_Toc450525067)

[3.1.1 Use Case Diagram 7](#_Toc450525068)

[3.1.2 Use Case List 8](#_Toc450525069)

[3.1.3 Use Case Text 9](#_Toc450525070)

[3.2 Non-functional Requirements 21](#_Toc450525071)

[3.2.1 Introduction 21](#_Toc450525072)

[3.2.2 Functionality 21](#_Toc450525073)

[3.2.3 Usability 22](#_Toc450525074)

[3.2.4 Reliability 22](#_Toc450525075)

[3.2.5 Purchased components 23](#_Toc450525076)

[3.2.6 Implementation Constraints 23](#_Toc450525077)

[4 Domain Model 24](#_Toc450525078)

[4.1 Domain Model Diagram 24](#_Toc450525079)

[4.1.1 Domain classes 24](#_Toc450525080)

[5 System Sequence Diagram 27](#_Toc450525081)

[5.1 UC1 – Do exercise program 27](#_Toc450525082)

[5.1.1 UC1 Main 1 – SSD 27](#_Toc450525083)

[5.1.2 UC1 Main 1– Operation Contracts 28](#_Toc450525084)

[5.1.3 UC1 Main 2 – SSD 29](#_Toc450525085)

[5.1.4 UC1 Main 2 – Operation Contracts 30](#_Toc450525086)

[5.2 UC2 – Do diet program 31](#_Toc450525087)

[5.2.1 UC2 – SSD 31](#_Toc450525088)

[5.2.2 UC2 – Operation Contracts 31](#_Toc450525089)

[5.3 UC3 – Interact with the social media 33](#_Toc450525090)

[5.3.1 UC3 – SSD 33](#_Toc450525091)

[5.3.2 UC3 – Operation Contracts 33](#_Toc450525092)

[5.4 UC4 – Sees contents 35](#_Toc450525093)

[5.4.1 UC4 – SSD 35](#_Toc450525094)

[5.4.2 UC4 – Operation Contracts 35](#_Toc450525095)

[6 Design Model 37](#_Toc450525096)

[6.1 Use Case 1 - Do Exercise Program Realization 37](#_Toc450525097)

[6.1.1 UC1 Main1 - Operation 1 37](#_Toc450525098)

[6.1.2 UC1 Main1 - Operation 2 38](#_Toc450525099)

[6.1.3 UC1 Main1 - Operation 3 39](#_Toc450525100)

[6.1.4 UC1 Main1 - Operation 4 40](#_Toc450525101)

[6.1.5 UC1 Main1 - Operation 5 41](#_Toc450525102)

[6.1.6 UC1 Main1 - Operation 6 42](#_Toc450525103)

[6.1.7 UC1 Main2 - Operation 1 43](#_Toc450525104)

[6.1.8 UC1 Main2 - Operation 2 44](#_Toc450525105)

[6.1.9 UC1 Main2 - Operation 3 45](#_Toc450525106)

[6.1.10 UC1 Main2 - Operation 4 46](#_Toc450525107)

[6.1.11 UC1 Main2 - Operation 5 47](#_Toc450525108)

[6.2 Use Case 2 - Do Diet Program Realization 48](#_Toc450525109)

[6.2.1 UC2 Operation 1 48](#_Toc450525110)

[6.2.2 UC2 Operation 2 49](#_Toc450525111)

[6.3 Use Case 3 - Interact with the Social Media Realization 50](#_Toc450525112)

[6.3.1 UC3 Operation 1 50](#_Toc450525113)

[6.3.2 UC3 Operation 2 51](#_Toc450525114)

[6.3.3 UC3 Operation 3 52](#_Toc450525115)

[6.4 Use Case 4 - Interact With Content Realization 53](#_Toc450525116)

[6.4.1 UC4 Operation 1 53](#_Toc450525117)

[6.4.2 UC4 Operation 2 54](#_Toc450525118)

[6.4.3 UC4 Operation 3 55](#_Toc450525119)

[6.4.4 UC4 Operation 4 56](#_Toc450525120)

[6.4.5 UC4 Operation 5 57](#_Toc450525121)

[7 References 58](#_Toc450525122)

[8 Glossary 58](#_Toc450525123)

[9 Content of Figures 59](#_Toc450525124)

[10 Content of Tables 60](#_Toc450525125)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No. | Date | Changed chapters | Description of change | | Authors |
| 1 | 2016-03-24 | all | Inception Phase Draft 1 | | Team E+1 |
| 2 | 2016-05-02 | 2 and 3 | 2 became 3 | | Team E+1 |
| 3 | 2016-05-02 | 3.1.1 Use Case Diagram | Deletion of Use Case 1 and 7, changed name of new Use Case 1, 2 and 4 | | Team E+1 |
| 4 | 2016-05-02 | 3.1.2 Use Case List | Content of Use Case 3 | | Team E+1 |
| 5 | 2016-05-02 | 3.1.3 Use Case Text | Content of Use Case 3, added Use Case 1, 2, 4 and 5 | | Team E+1 |
| 6 | 2016-05-08 | 8 Glossary | Updated/added some descriptions | | Team E+1 |
| No. | Date | New chapters | | Authors | |
| 1 | 2016-03-24 | 1 Table of Content, 2 Vision and 3 Requirements | | Team E+1 | |
| 2 | 2016-05-02 | 4 Domain Model | | Team E+1 | |
| 3 | 2016-05-02 | 5 System Sequence Diagram | | Team E+1 | |
| 4 | 2016-05-06 | 6 Design Model | | 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

# Requirements

## Functional Requirements

### Use Case Diagram



Figure 3‑1 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 body information for age, weight, height, sex, activity level, goal weight, and time frame. (optional)
* System saves the information about the user (optional)
* System provides a daily calorie amount in order to reach the goal weight in the time frame
* User creates or selects 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.* Interact with content (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:** Do exercise program (User)

**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 wants 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 confirms that he finished a set of his exercise
10. User confirms 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 message
   2. User checks the connection to his Wi-Fi
4. In case the user wants to change his workout.
5. The user breaks off his workout
6. The app asks the user if he really wants to break 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 message
   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 message
    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 message
9. User downloads the chosen workout again
10. In case the user wants to change his workout.
11. User breaks off his workout
12. System asks the user if he really wants to break 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 confirms 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 - 2**

**Use case name:** Select a diet program (User)

**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 wants to do his diet with or without an exercise program.

**Postconditions:**

* The database has to store the information frequently.
* The meal plans displays the total nutritional information for the day.
* The system keeps a record of all list of meals and user’s stats.

**Main Success Scenario:**

1. User starts a diet.
2. User inputs his information and receives a daily calorie target.
3. User starts a meal list for the day.
4. User can select individual meals to add food items to the list from a database.
5. User can share their meal plan to the social media to help others in a similar situation.

**Extensions (Alternative Flows):**

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

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 is stored.
3. The user creates a new plan.

4a. In case the food the user wants to load isn’t in the database.

1. The user adds his own recipe
2. The user loads his own recipe

4b. In case the recipe 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 recipe.

4c. In case the user wants to use a complete meal plan from the database.

1. Rather than adding individual food items the user can use a complete meal plan from the database.
2. The user can find and download whole meal plans from the social media platform.

5.. In case the user wants to leave reviews .

1. The user writes a review of the meal plan..
2. The review is saved in online content.

**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 frequently check their meal plans, make changes, and to try new meal plans recommended by others.

**Open Issues:**

None

**Use Case Text – 3**

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

**Level:** User goal

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

**Stakeholders and Interests:**

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

**Preconditions:**

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

**Postconditions:**

* The database had stored the information about user’s interactions

**Main Success Scnario:**

1. User chooses ”Feed/Media” button
2. User chooses content
3. User sees content
4. User interacts with content

**Extensions (Alternative Flows):**

1. In case the app doesn’t work correctly
2. The user stops the app and reboots the system
3. The user starts the app again and tries to see the content
4. In case the user can’t access the database
5. The user stops the app and reboot the system
6. The user checks his connection
7. The user restarts his router
8. The user reconnects to the internet
9. The user starts the app again and tries to see the content
10. In case the system doesn’t react
11. The user stops the app and reboots the system
12. The user starts again and return to his current plan
13. In case the content isn’t available
14. The user refreshes the “Feed/Media”
15. The user tries to load the content again
16. In case the user doesn’t see the content
17. The user reboots app
18. he user tries to load the content again
19. User cannot interact with content
20. The user refresh the content
21. The user tries again

**Special Requirements**:

* Touch screen UI on the mobile device. Text size and colors can be adjustable foruser 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?

**Use Case Text – 4**

**Use case name:** InteractWithContent (User)

**Level:** User goal

**Primary Actor:** User, Database

**Stakeholders and Interests:**

* User: wants to compare and get a matching exercise program and diet program according to his personal goals
* Data base: wants to store all the uploaded information by the user and recommend the most popular programs to the user

**Preconditions:**

* The user has created an account for the app.
* The user has to be connected with the internet.
* The user chose the content he wants to see.
* The user has to have free memory space on his device for the downloaded content.

**Postconditions:**

* None

**Main Success Scenario:**

1. User accesses the database.
2. User selects between exercise and diet program.
3. Database highlights popular content.
4. User chooses a specific file to see.
5. User downloads the chosen content.
6. User sees the content of the file.
7. User decides to search for another file. Repeat from step 2 until user decides not to see anymore files.
8. User exits the database.

**Extensions (Alternative Flows):**

1. In case the app doesn’t work correctly
   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 stops the app and reboots the system.
   2. The user checks his connection.
   3. The user restarts his router.
   4. The user reconnects to the internet.
   5. The user starts the app again and try to see the content.
3. In case the user doesn’t have connection to the database
4. The user checks the internet connection of his device.
5. If the connection of the device is bad.
6. The user stops the app and reboot the system.
7. The user reconnects to the internet.
8. The user starts the app again and tries to see the content.
9. If the connection didn’t get better.
10. The user stops the app and reboot the system.
11. The user restarts his router.
12. The user reconnects to the internet.
13. The user starts the app again and tries to see the content.
14. In case the user can’t download the file.
15. The user checks the free memory space on his device.
16. If the memory on the device is full.
17. The user deletes data to get enough free memory space for the file.
18. In case the user can’t see the content of the file.
19. The user checks if he has an appropriate app to open the type of the file.
20. If the user doesn’t have an appropriate app to open the type of the file.
21. The user installs an appropriate app.
22. If the user still can’t see the content of the file.
23. The user checks the size of the file.
24. The user reports the issue.
25. The system/admin checks the reported file.

**Special Requirements:**

* User should be able to easily navigate through the folders in the database.

**Frequency of Occurrence:**

* Frequent. The database enables the user to compare their programs. Users are expected to frequently compare their exercise and diet program and update it often in order to have the newest and best available information when starting an exercise or diet program.

**Open Issues:**

* None

**Use Case Text – 5**

**Use Case Name:** Manage Users

**Level:** Summary

**Primary Actor:** Social Media Administrator

**Stakeholders and Interests:**

* Social Media Administrator: Wants the content in social media to be politically correct. Ensures that there’s no cybercrime happening.
* User: Wants to have safe experience in using application’s social media.

**Preconditions**:

* There are users

**Main Success Scenario:**

1. Admin notices a user who has bad behavior in the comment section
2. Admin deletes the comment and sends warning to the user
3. User stops his/her bad behavior

**Extensions (Alternative Flows):**

1a. Admin notices user who’s comment or uploaded content are against the rules

1. Admin deletes the comment or content and bans the user for certain time from the social media
2. User can’t access Aegle’s social media for a certain time

1b. Admin notices user who’s comment or uploaded content are hazardous and clearly against the law.

1. Admin gives user permanent ban and reports police for further actions.
2. User receives permanent ban and can’t access Aeagle’s social media anymore.

2. Because of continuous harassment of the rules, admin decides to give the user temporary ban from the social media

3. User won’t stop his/her bad behavior after warnings and continues uploading or giving inappropriate contents and comments.

1. Admin restricts user’s access rights (e.g. right to comment or upload)
2. If even a temporary ban won’t stop the user’s actions, admin can give him/her apermanent ban

**Special Requirements:**

None

**Technology and Data Variations List:**

None

**Frequency of Occurrence:**

Whenever there’s a user who is abusing their access rights or other users.

**Open Issues:**

None

## 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 signifies 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 problems during the use of the app.

Health safety

* The app should provide high quality exercise programs and diet recipes. 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

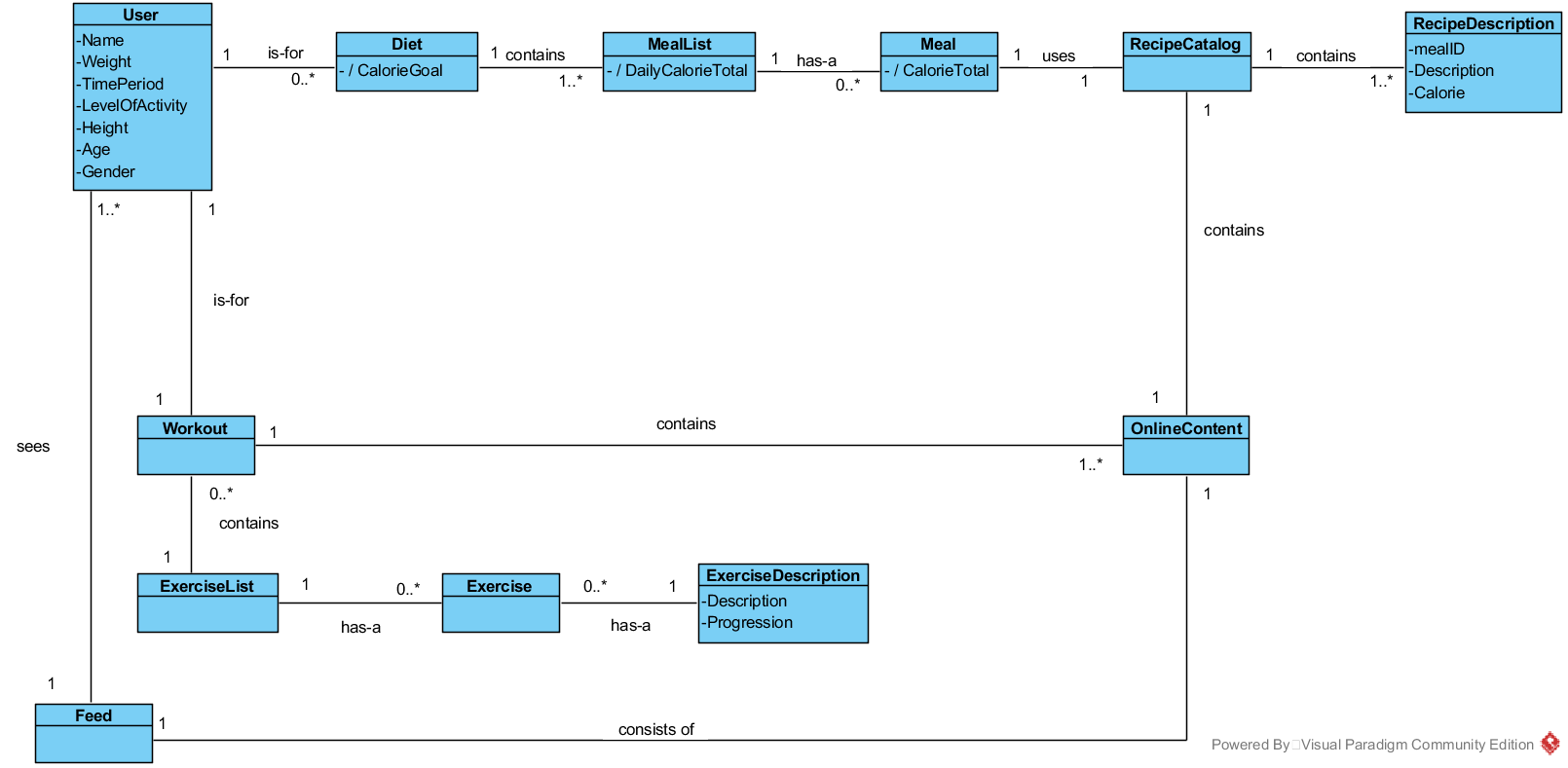


Figure 4‑1 Domain Class Diagram

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 4‑1 description for domain class User

|  |  |
| --- | --- |
| **User** | |
| **Attributes** | Name  Weight  TimePeriod  LevelOfActivty  Height  Age  Gender |
| **Association** | * One user makes one diet * One user makes one workout * One or more user uses one feed |

Table 4‑2 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 4‑3 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 4‑4 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 4‑5 description for domain class RecipeCatalog

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

Table 4‑6 description for domain class RecipeDescription

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

Table 4‑7 description for domain class Workout

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

Table 4‑8 description for domain class Workout

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

Table 4‑9 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 4‑10 description for domain class ExerciseDescription

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

Table 4‑11 description for domain class OnlineContent

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

Table 4‑12 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

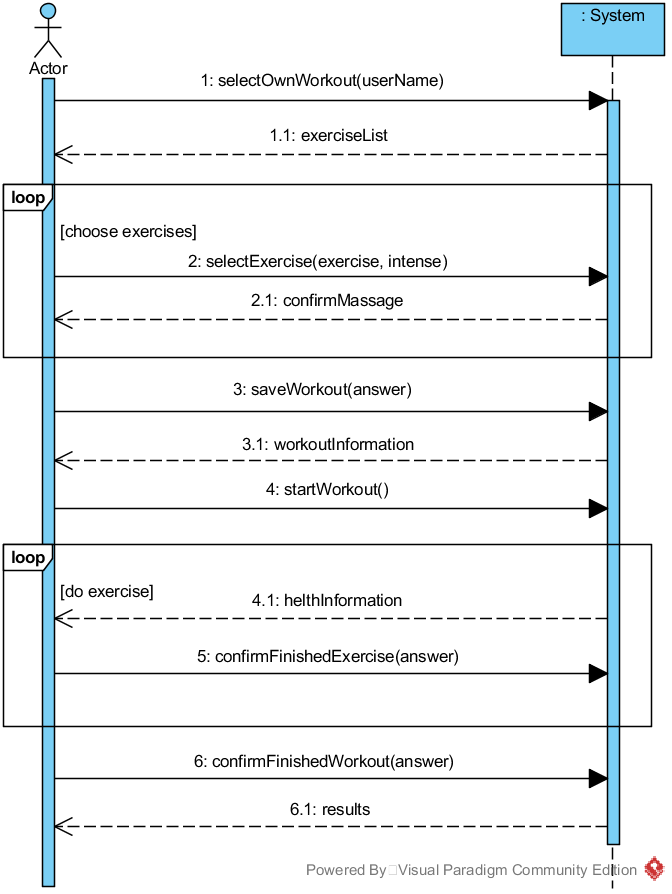


Figure 5‑1 Use Case 1 SSD, Main1

### UC1 Main 1– Operation Contracts

Table 5‑1 UC1 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 5‑2 UC1 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 5‑3 UC1 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 5‑4 UC1 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 5‑5 UC1 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 5‑6 UC1 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

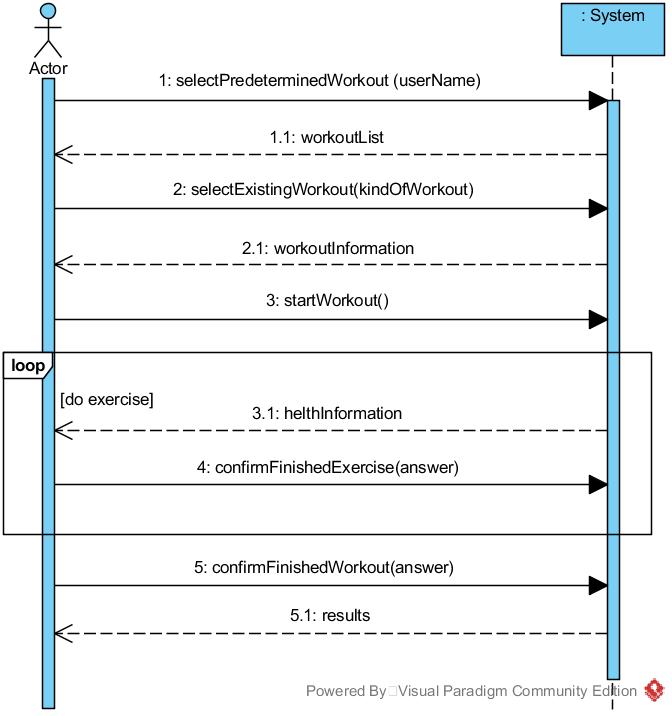


Figure 5‑2 Use Case 1 SSD, Main2

### UC1 Main 2 – Operation Contracts

Table 5‑7 UC1 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 5‑8 UC1 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 5‑9 UC1 Main 2 - startWorkout()

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

Table 5‑10 UC1 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 5‑11 UC1 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

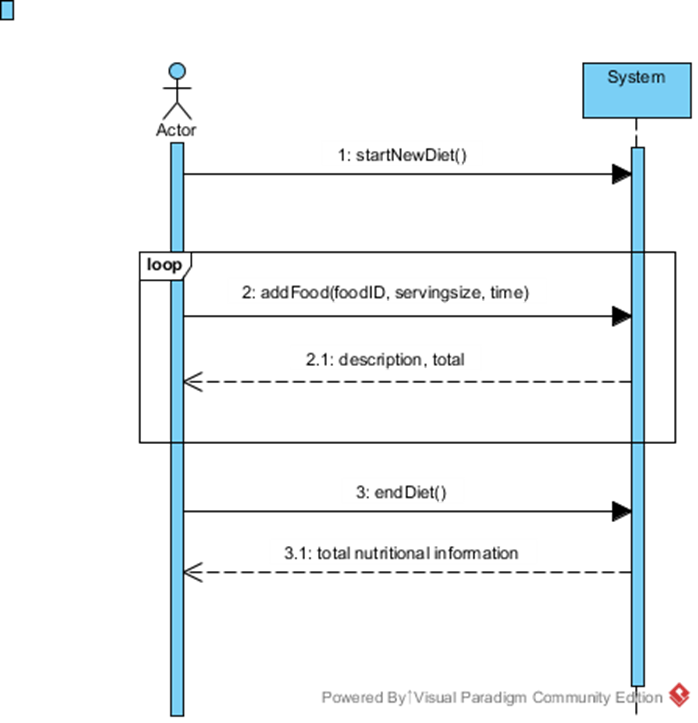


Figure 5‑3 Use Case 2 SSD

User begins a Diet for the day. The user can add a food item specifying the serving size and time of day it is eaten. Once user is finished adding food items changes are saved and the total values are displayed.

### UC2 – Operation Contracts

Table 5‑12 UC2 - startNewDiet()

|  |  |
| --- | --- |
| **Operation:** | startNewDiet() |
| **Cross references:** | Do Diet Program |
| **Preconditions:** | * None |
| **Postconditions:** | * Do Diet Program * d was associated to user * Attributes of MealList were initialized |

Table 5‑13 UC2 - addFood(food, servingsize, Time)

|  |  |
| --- | --- |
| **Operation:** | addFood(food, servingsize, Time) |
| **Cross references:** | Do Diet Program |
| **Preconditions:** | * MealList has been initialized |
| **Postconditions:** | * A Meal instance m was created * m was associated with current MealList. * An instance of class Time was created (for which meal i.e. lunch). * Attributes of Meal were initialized |

## UC3 – Interact with the social media

### UC3 – SSD

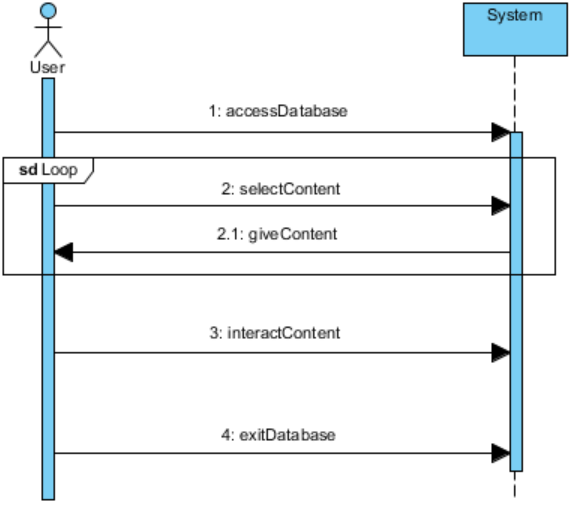


Figure 5‑4 Use Case 3 SSD

User logs in to the system and get access to Social Media database. Then User selects what content he wants to see. Information about that is sent to database and database gives back the content what the User choose. User interacts with content and these interaction will be stored into database for others to see. User logs out after he is done.

### UC3 – Operation Contracts

Table 5‑14 UC3 - accessDatabase ()

|  |  |
| --- | --- |
| **Operation:** | accessDatabase () |
| **Cross references:** | Interact with social media |
| **Preconditions:** | * User has already an account or user creates an account |
| **Postconditions:** | * User has access to Database |

When User logs in and wants to interact with “Feed/Media”, User will have input his account or create a new one, if he doesn’t have already.

Table 5‑15 UC3 - selectContent()

|  |  |
| --- | --- |
| **Operation:** | selectContent() |
| **Cross references:** | Interact with social media |
| **Preconditions:** | * User has already log in and have a access to database |
| **Postconditions:** | * Information about User’s requested content has been sent |

When User logs in and wants to interact with “Feed/Media”, User will have to select what kind of content one would like to see. After User has selected the type of content he or she wants to see, information about the choice will be sent to System.

Table 5‑16 UC3 - giveContent()

|  |  |
| --- | --- |
| **Operation:** | giveContent() |
| **Cross references:** | Interact with social media |
| **Preconditions:** | * System has information about what content User wants to receive |
| **Postconditions:** | * User’s requested content has been delivered to User |

System receives an information about User’s request to see chosen content and will deliver requested content to User.

Table 5‑17 UC3 - interactContent()

|  |  |
| --- | --- |
| **Operation:** | interactContent() |
| **Cross references:** | Interact with social media |
| **Preconditions:** | * User has received requested content |
| **Postconditions:** | * System stored information about the User’s interaction with the contents * Comment, rate or shared content has been submitted |

When User receives the content from the System User can start to interact with the content. Interactions will be stored into database. For example if User wrote a comment on video, it will be stored to database to others to see.

## UC4 – Sees contents

### UC4 – SSD

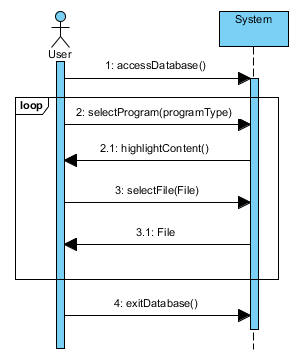


Figure 5‑5 Use Case 4 SSD

### UC4 – Operation Contracts

Table 5‑18 UC4 - accessDatabase()

|  |  |
| --- | --- |
| **Operation:** | accessDatabase() |
| **Cross references:** | InteractWithContent |
| **Preconditions:** | * None |
| **Postconditions:** | * user was associated with the Feed * Feed was associated with the OnlineContent |

Table 5‑19 UC4 - selectProgram(programType)

|  |  |
| --- | --- |
| **Operation:** | selectProgram(programType) |
| **Cross references:** | InteractWithContent |
| **Preconditions:** | * None |
| **Postconditions:** | * OnlineContent.Catalog[] (variable) was initialized * depending on programType either an ExerciseCatalog or a RecipeCatalog was saved into with the OnlineContent.Catalog[] |

Table 5‑20 UC4 - highlightContent()

|  |  |
| --- | --- |
| **Operation:** | highlightContent() |
| **Cross references:** | InteractWithContent |
| **Preconditions:** | * There are up-voted files in the onlineContent |
| **Postconditions:** | * onlineContent.Catalog.highlight (variable) became true for up voted programs |

Table 5‑21 UC4 - selectFile(File)

|  |  |
| --- | --- |
| **Operation:** | selectFile(File) |
| **Cross references:** | InteractWithContent |
| **Preconditions:** | * There are files to select |
| **Postconditions:** | * user.programm (variable) was initialized * file was associated with the user |

Table 5‑22 UC4 - exitDatabase()

|  |  |
| --- | --- |
| **Operation:** | exitDatabase() |
| **Cross references:** | InteractWithContent |
| **Preconditions:** | * None |
| **Postconditions:** | * feed association with the database was deleted * user association with the feed was deleted |

# Design Model

## Use Case 1 - Do Exercise Program Realization

### UC1 Main1 - Operation 1

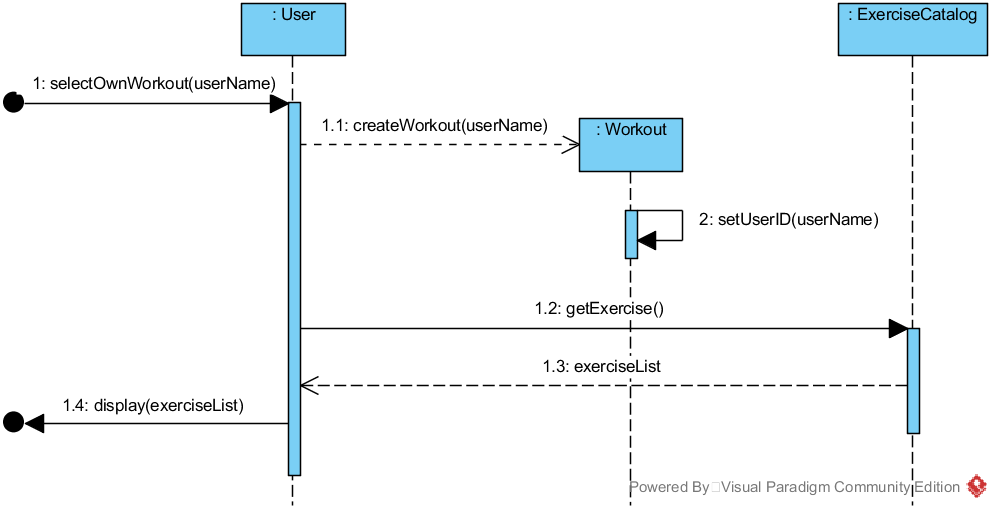


Figure 6‑1 UC1 Main1 Sequence Diagram for Operation 1

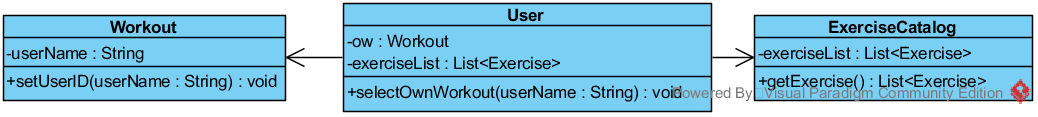


Figure 6‑2 UC1 Main1 Class Diagram Operation 1

Table 6‑1 UC1 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. |

### UC1 Main1 - Operation 2

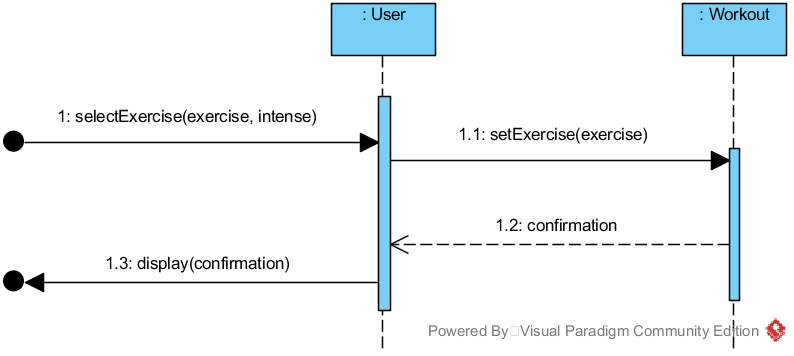


Figure 6‑3 UC1 Main1 Sequence Diagram for Operation 2

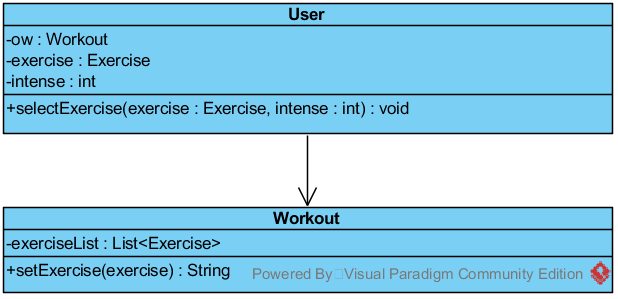


Figure 6‑4 UC1 Main1 Class Diagram Operation 2

Table 6‑2 UC1 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. |

### UC1 Main1 - Operation 3

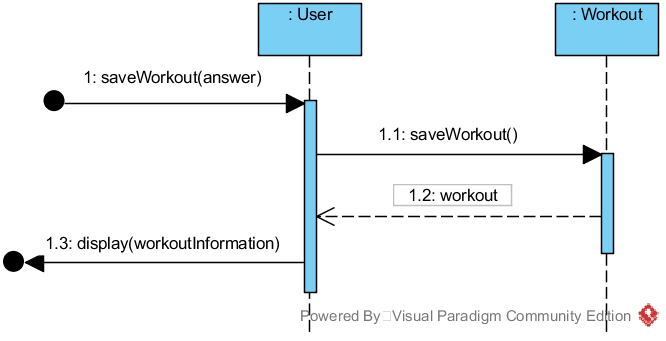


Figure 6‑5 UC1 Main1 Sequence Diagram for Operation 3

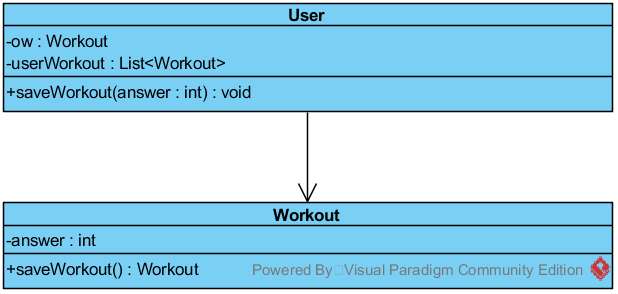


Figure 6‑6 UC1 Main1 Class Diagram Operation 3

Table 6‑3 UC1 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. |

### UC1 Main1 - Operation 4

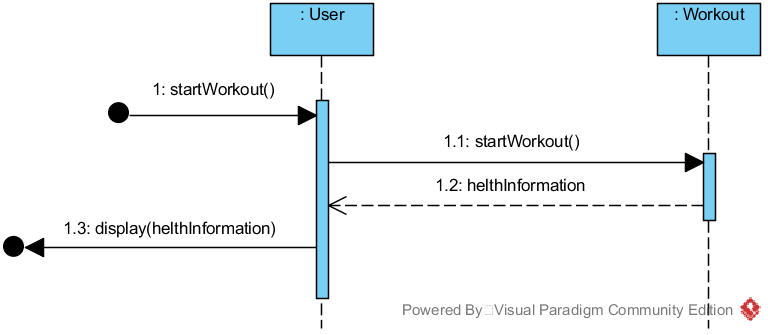


Figure 6‑7 UC1 Main1 Sequence Diagram for Operation 4

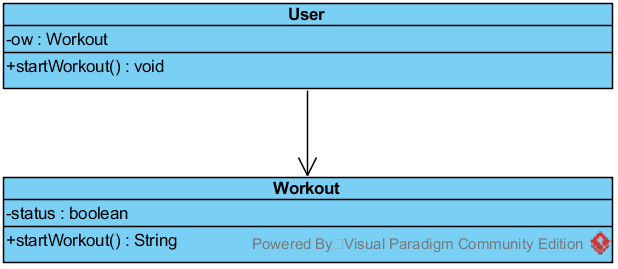


Figure 6‑8 UC1 Main1 Class Diagram Operation 4

Table 6‑4 UC1 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. |

### UC1 Main1 - Operation 5

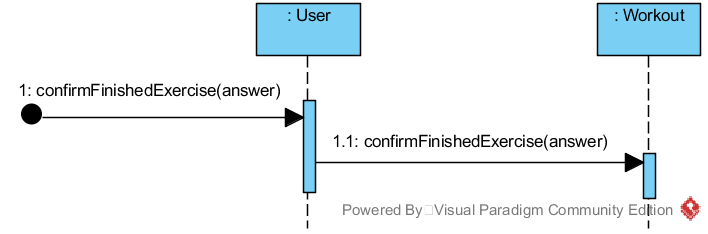


Figure 6‑9 UC1 Main1 Sequence Diagram for Operation 5

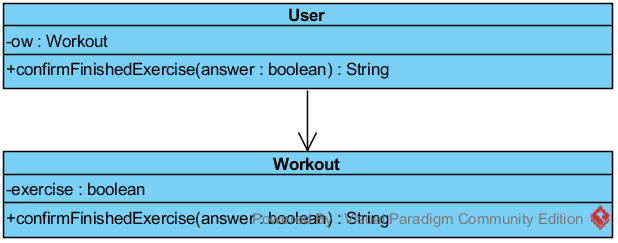


Figure 6‑10 UC1 Main1 Class Diagram Operation 5

Table 6‑5 UC1 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. |

### UC1 Main1 - Operation 6

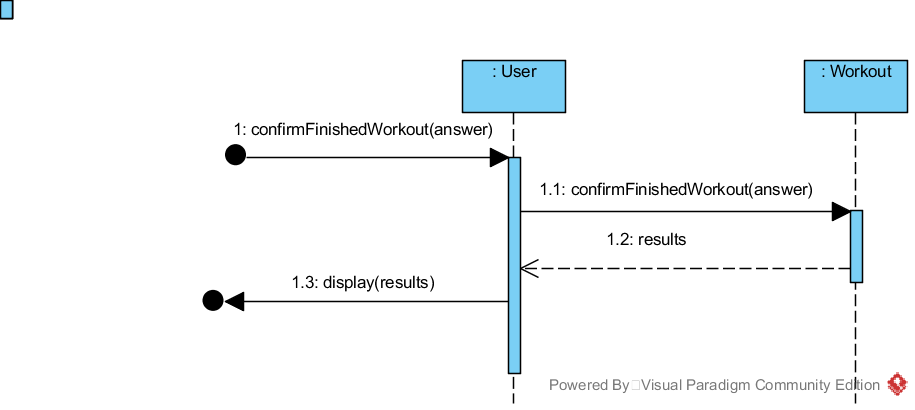


Figure 6‑11 UC1 Main1 Sequence Diagram for Operation 6

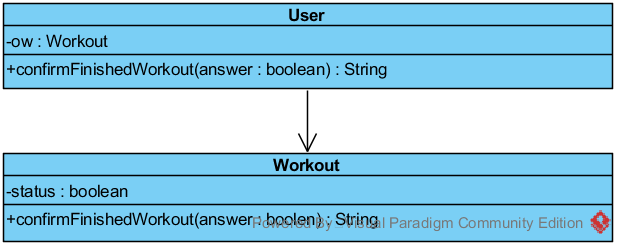


Figure 6‑12 UC1 Main1 Class Diagram Operation 6

Table 6‑6 UC1 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. |

### UC1 Main2 - Operation 1

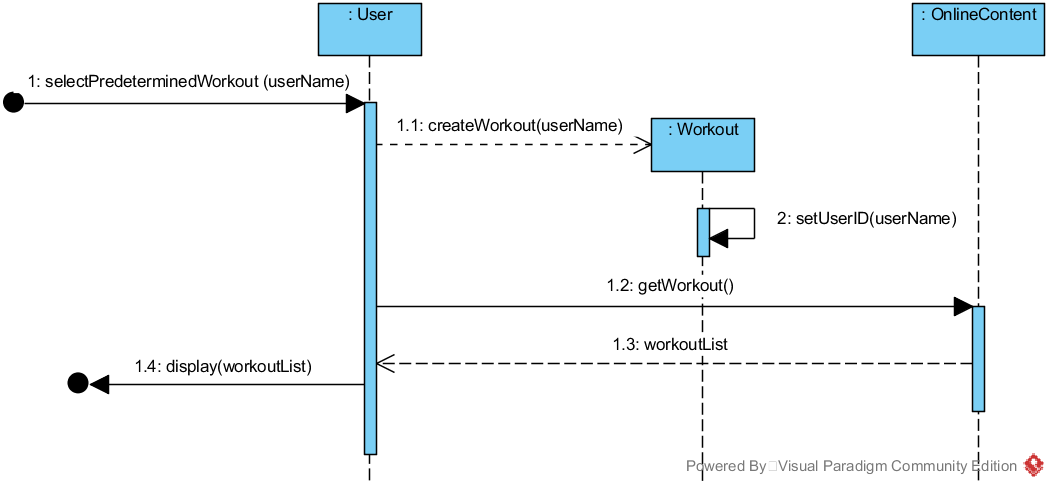


Figure 6‑13 UC1 Main2 Sequence Diagram for Operation 1

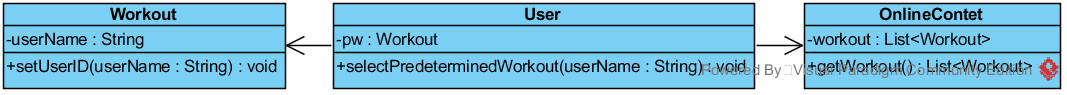


Figure 6‑14 UC1 Main2 Class Diagram Operation 1

Table 6‑7 UC1 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. |

### UC1 Main2 - Operation 2

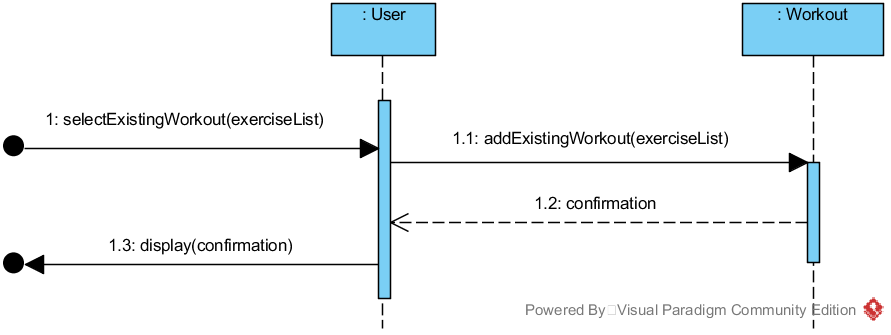


Figure 6‑15 UC1 Main2 Sequence Diagram for Operation 2

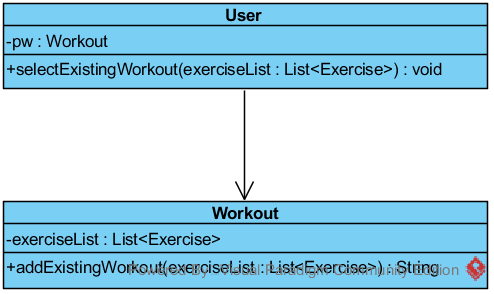


Figure 6‑16 UC1 Main2 Class Diagram Operation 2

Table 6‑8 UC1 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. |

### UC1 Main2 - Operation 3

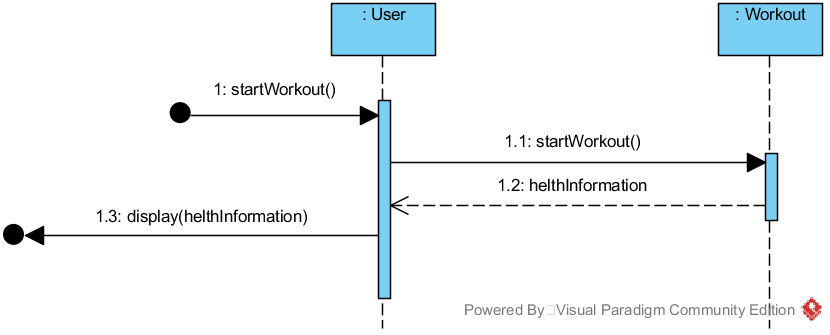


Figure 6‑17 UC1 Main2 Sequence Diagram for Operation 3

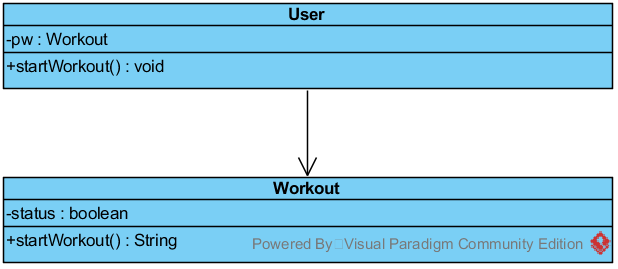


Figure 6‑18 UC1 Main2 Class Diagram Operation 3

Table 6‑9 UC1 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. |

### UC1 Main2 - Operation 4

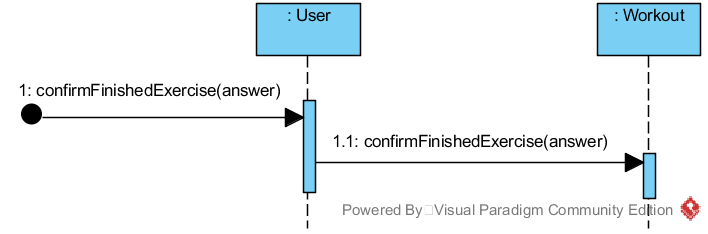


Figure 6‑19 UC1 Main2 Sequence Diagram for Operation 4

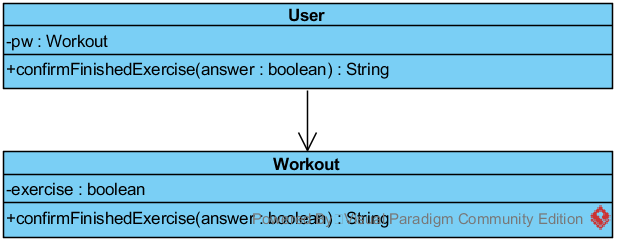


Figure 6‑20 UC1 Main2 Class Diagram Operation 4

Table 6‑10 UC1 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. |

### UC1 Main2 - Operation 5

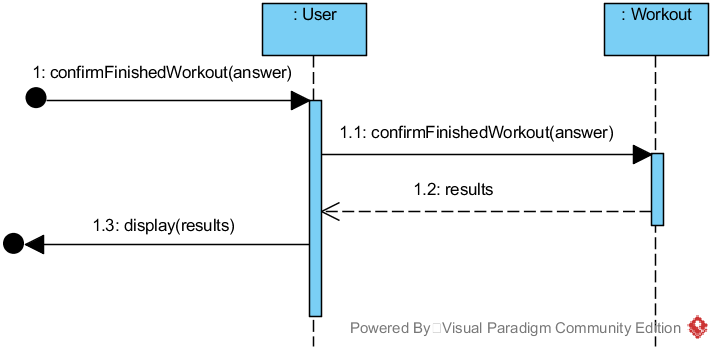


Figure 6‑21 UC1 Main2 Sequence Diagram for Operation 5

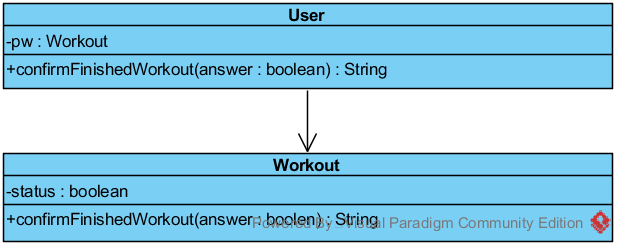


Figure 6‑22 UC1 Main2 Class Diagram Operation 5

Table 6‑11 UC1 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. |

## Use Case 2 - Do Diet Program Realization

### UC2 Operation 1



Figure 6‑23 UC2 Sequence Diagram for Operation 1



Figure 6‑24 UC2 Class Diagram Operation 1

Table 6‑12 UC2 GRASP Operation 1

|  |  |
| --- | --- |
| **Creator** | * User |
| **Information Expert** | * Diet knows user information from user |
| **Low Coupling** | * User does not need to know how to calculate daily calorie goal |
| **High Cohesion** | * User has only one responsibility to startDiet(). |
| **Controller** | * Controlled by user |

### UC2 Operation 2

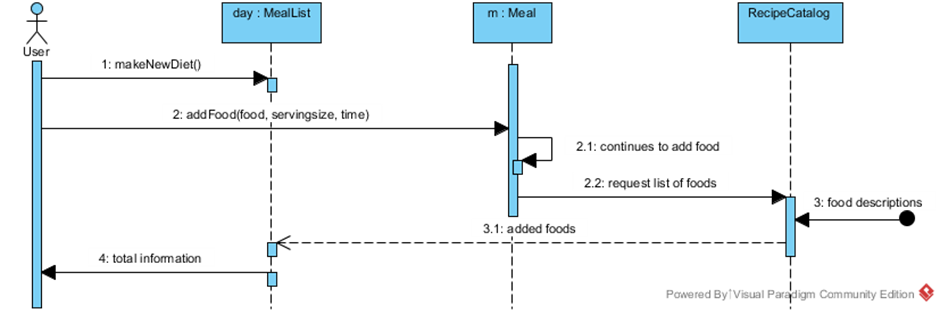


Figure 6‑25 UC2 Sequence Diagram for Operation 2

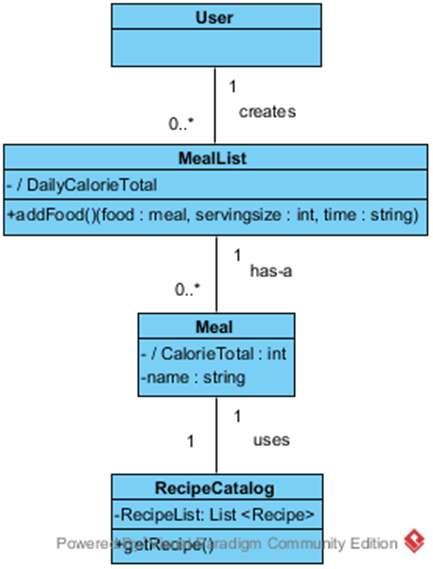


Figure 6‑26 UC2 Class Diagram Operation 2

Table 6‑13 UC2 GRASP Operation 2

|  |  |
| --- | --- |
| **Creator** | * User creates a Diet for the day. |
| **Information Expert** | * MealList knows nutrition information of Meals from RecipeCatalog. |
| **Low Coupling** | * User does not need to know how to get list of foods. |
| **High Cohesion** | * MealList has only one responsibility to display information. Organized by time of day. |
| **Controller** | * Controlled by user |

## Use Case 3 - Interact with the Social Media Realization

### UC3 Operation 1

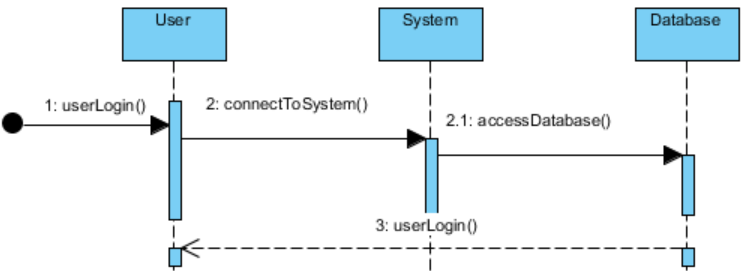


Figure 6‑27 UC3 Sequence Diagram for Operation 1

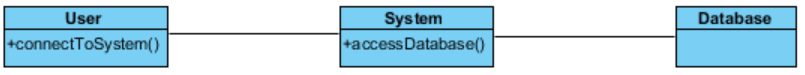


Figure 6‑28 UC3 Class Diagram Operation 1

Table 6‑14 UC3 GRASP Operation 1

|  |  |
| --- | --- |
| **Creator** | * User |
| **Information Expert** | * Database knows user’s account information |
| **Low Coupling** | * None |
| **High Cohesion** | * User has single responsibility to log in |
| **Controller** | * None |

### UC3 Operation 2

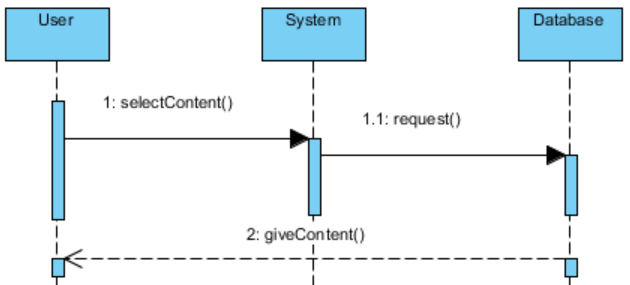


Figure 6‑29 UC3 Sequence Diagram for Operation 2

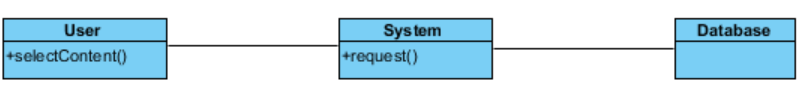


Figure 6‑30 UC3 Class Diagram Operation 2

Table 6‑15 UC3 GRASP Operation 2

|  |  |
| --- | --- |
| **Creator** | * User |
| **Information Expert** | * Database knows content |
| **Low Coupling** | * User does not need to know how to get content. |
| **High Cohesion** | * User has single responsibility to select content |
| **Controller** | * User |

### UC3 Operation 3

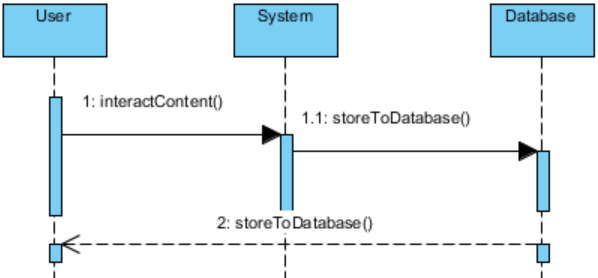


Figure 6‑31 UC3 Sequence Diagram for Operation 3

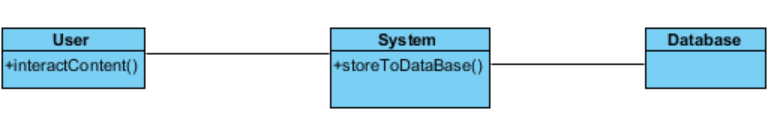


Figure 6‑32 UC3 Class Diagram Operation 3

Table 6‑16 UC3 GRASP Operation 2

|  |  |
| --- | --- |
| **Creator** | * User |
| **Information Expert** | * None |
| **Low Coupling** | * User does not need to know how to store information to database |
| **High Cohesion** | * User has single responsibility to interact with content |
| **Controller** | * User |

## Use Case 4 - Interact With Content Realization

### UC4 Operation 1

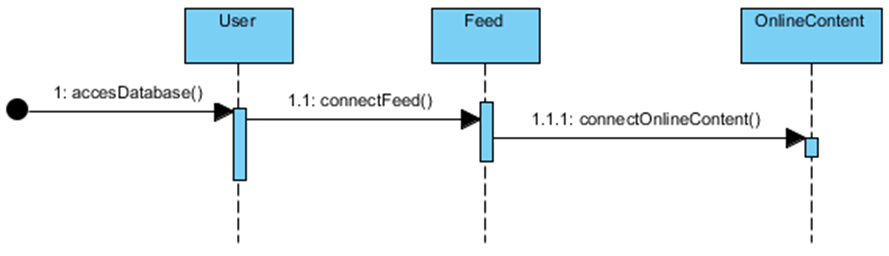


Figure 6‑33 UC4 Sequence Diagram for Operation 1



Figure 6‑34 UC4 Class Diagram Operation 1

Table 6‑17 UC4 GRASP Operation 1

|  |  |
| --- | --- |
| **Creator** | * None |
| **Information Expert** | * None |
| **Low Coupling** | * Responsibilities are delegated so that the dependencies remain low |
| **High Cohesion** | * User has only one responsibility to connectFeed() |
| **Controller** | * None |

### UC4 Operation 2

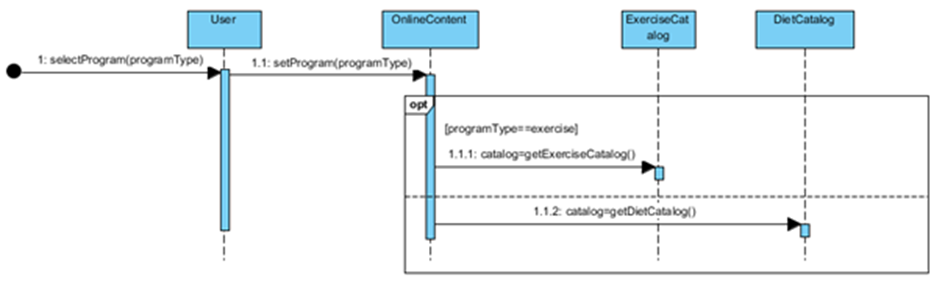


Figure 6‑35 UC4 Sequence Diagram for Operation 2



Figure 6‑36 UC4 Class Diagram Operation 2

Table 6‑18 UC4 GRASP Operation 2

|  |  |
| --- | --- |
| **Creator** | * None |
| **Information Expert** | * OnlineContent knows about the programtypes |
| **Low Coupling** | * Responsibilities are delegated so that the dependencies remain low |
| **High Cohesion** | * User has only one responsibility to selectProgram() |
| **Controller** | * None |

### UC4 Operation 3

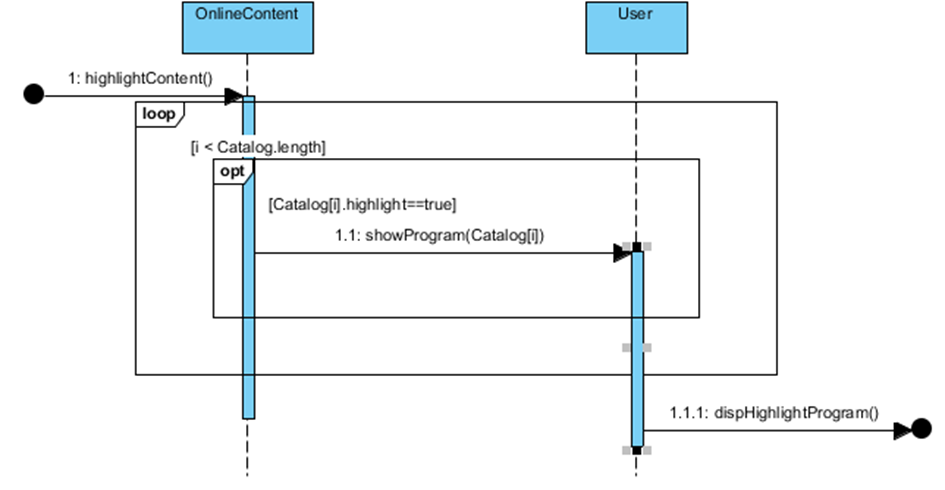


Figure 6‑37 UC4 Sequence Diagram for Operation 3

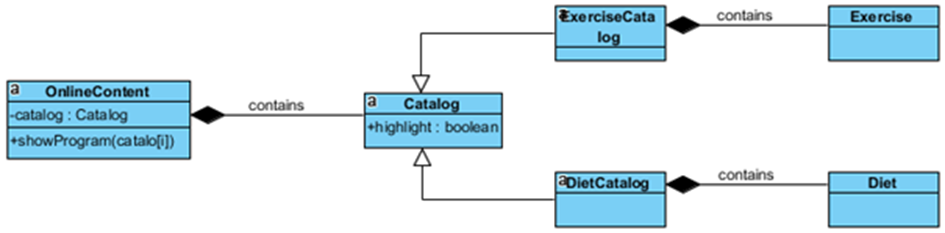


Figure 6‑38 UC4 Class Diagram Operation 3

Table 6‑19 UC4 GRASP Operation 3

|  |  |
| --- | --- |
| **Creator** | * None |
| **Information Expert** | * Catalog knows about the highlighted programs |
| **Low Coupling** | * Responsibilities are delegated so that the dependencies remain low |
| **High Cohesion** | * None |
| **Controller** | * None |

### UC4 Operation 4

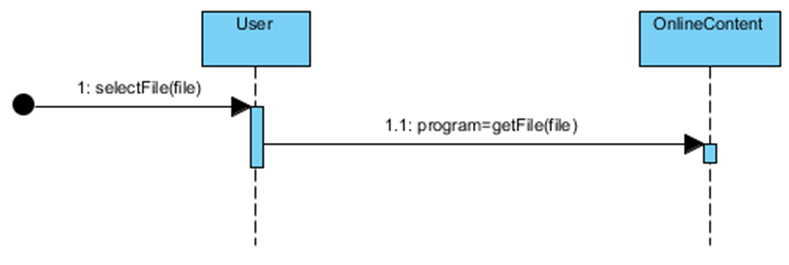


Figure 6‑39 UC4 Sequence Diagram for Operation 4

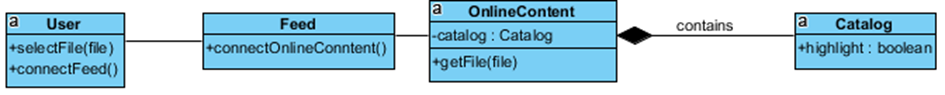


Figure 6‑40 UC4 Class Diagram Operation 4

Table 6‑20 UC4 GRASP Operation 4

|  |  |
| --- | --- |
| **Creator** | * None |
| **Information Expert** | * Catalog knows about the file |
| **Low Coupling** | * None |
| **High Cohesion** | * None |
| **Controller** | * None |

### UC4 Operation 5

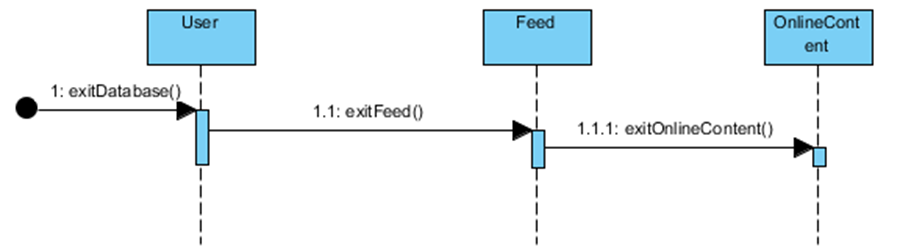


Figure 6‑41 UC4 Sequence Diagram for Operation 5

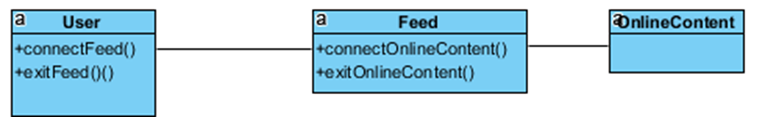


Figure 6‑42 UC4 Class Diagram Operation 5

Table 6‑21 UC4 GRASP Operation 5

|  |  |
| --- | --- |
| **Creator** | * None |
| **Information Expert** | * None |
| **Low Coupling** | * Responsibilities are delegated so that the dependencies remain low |
| **High Cohesion** | * None |
| **Controller** | * None |

# 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>.

# 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 |
| Recipe | Either an individual food item such as banana or a more complicated meal such as a sandwich. |
| Meal List | A menu that displays all the food items and recipes for the day organized by each meal. |
| UC | Short form of Use Case. |
| Workout | Includes a list of exercises for the user. |
| Exercise List | A collection of exercises. |
| Exercise | A description how to move the body to get fit. |
| GRASP | Consist of guidelines for assigning responsibility to classes and objects |

# Content of Figures

[Figure 3‑1 Use case diagram 7](#_Toc450524999)

[Figure 4‑1 Domain Class Diagram 24](#_Toc450525000)

[Figure 5‑1 Use Case 1 SSD, Main1 27](#_Toc450525001)

[Figure 5‑2 Use Case 1 SSD, Main2 29](#_Toc450525002)

[Figure 5‑3 Use Case 2 SSD 31](#_Toc450525003)

[Figure 5‑4 Use Case 3 SSD 33](#_Toc450525004)

[Figure 5‑5 Use Case 4 SSD 35](#_Toc450525005)

[Figure 6‑1 UC1 Main1 Sequence Diagram for Operation 1 37](#_Toc450525006)

[Figure 6‑2 UC1 Main1 Class Diagram Operation 1 37](#_Toc450525007)

[Figure 6‑3 UC1 Main1 Sequence Diagram for Operation 2 38](#_Toc450525008)

[Figure 6‑4 UC1 Main1 Class Diagram Operation 2 38](#_Toc450525009)

[Figure 6‑5 UC1 Main1 Sequence Diagram for Operation 3 39](#_Toc450525010)

[Figure 6‑6 UC1 Main1 Class Diagram Operation 3 39](#_Toc450525011)

[Figure 6‑7 UC1 Main1 Sequence Diagram for Operation 4 40](#_Toc450525012)

[Figure 6‑8 UC1 Main1 Class Diagram Operation 4 40](#_Toc450525013)

[Figure 6‑9 UC1 Main1 Sequence Diagram for Operation 5 41](#_Toc450525014)

[Figure 6‑10 UC1 Main1 Class Diagram Operation 5 41](#_Toc450525015)

[Figure 6‑11 UC1 Main1 Sequence Diagram for Operation 6 42](#_Toc450525016)

[Figure 6‑12 UC1 Main1 Class Diagram Operation 6 42](#_Toc450525017)

[Figure 6‑13 UC1 Main2 Sequence Diagram for Operation 1 43](#_Toc450525018)

[Figure 6‑14 UC1 Main2 Class Diagram Operation 1 43](#_Toc450525019)

[Figure 6‑15 UC1 Main2 Sequence Diagram for Operation 2 44](#_Toc450525020)

[Figure 6‑16 UC1 Main2 Class Diagram Operation 2 44](#_Toc450525021)

[Figure 6‑17 UC1 Main2 Sequence Diagram for Operation 3 45](#_Toc450525022)

[Figure 6‑18 UC1 Main2 Class Diagram Operation 3 45](#_Toc450525023)

[Figure 6‑19 UC1 Main2 Sequence Diagram for Operation 4 46](#_Toc450525024)

[Figure 6‑20 UC1 Main2 Class Diagram Operation 4 46](#_Toc450525025)

[Figure 6‑21 UC1 Main2 Sequence Diagram for Operation 5 47](#_Toc450525026)

[Figure 6‑22 UC1 Main2 Class Diagram Operation 5 47](#_Toc450525027)

[Figure 6‑23 UC2 Sequence Diagram for Operation 1 48](#_Toc450525028)

[Figure 6‑24 UC2 Class Diagram Operation 1 48](#_Toc450525029)

[Figure 6‑25 UC2 Sequence Diagram for Operation 2 49](#_Toc450525030)

[Figure 6‑26 UC2 Class Diagram Operation 2 49](#_Toc450525031)

[Figure 6‑27 UC3 Sequence Diagram for Operation 1 50](#_Toc450525032)

[Figure 6‑28 UC3 Class Diagram Operation 1 50](#_Toc450525033)

[Figure 6‑29 UC3 Sequence Diagram for Operation 2 51](#_Toc450525034)

[Figure 6‑30 UC3 Class Diagram Operation 2 51](#_Toc450525035)

[Figure 6‑31 UC3 Sequence Diagram for Operation 3 52](#_Toc450525036)

[Figure 6‑32 UC3 Class Diagram Operation 3 52](#_Toc450525037)

[Figure 6‑33 UC4 Sequence Diagram for Operation 1 53](#_Toc450525038)

[Figure 6‑34 UC4 Class Diagram Operation 1 53](#_Toc450525039)

[Figure 6‑35 UC4 Sequence Diagram for Operation 2 54](#_Toc450525040)

[Figure 6‑36 UC4 Class Diagram Operation 2 54](#_Toc450525041)

[Figure 6‑37 UC4 Sequence Diagram for Operation 3 55](#_Toc450525042)

[Figure 6‑38 UC4 Class Diagram Operation 3 55](#_Toc450525043)

[Figure 6‑39 UC4 Sequence Diagram for Operation 4 56](#_Toc450525044)

[Figure 6‑40 UC4 Class Diagram Operation 4 56](#_Toc450525045)

[Figure 6‑41 UC4 Sequence Diagram for Operation 5 57](#_Toc450525046)

[Figure 6‑42 UC4 Class Diagram Operation 5 57](#_Toc450525047)

# Content of Tables

[Table 2‑1 Key High-level Goals 4](#_Toc450524942)

[Table 2‑2 Summary of Benefits 5](#_Toc450524943)

[Table 4‑1 description for domain class User 24](#_Toc450524944)

[Table 4‑2 description for domain class Diet 25](#_Toc450524945)

[Table 4‑3 description for domain class MealList 25](#_Toc450524946)

[Table 4‑4 description for domain class Meal 25](#_Toc450524947)

[Table 4‑5 description for domain class RecipeCatalog 25](#_Toc450524948)

[Table 4‑6 description for domain class RecipeDescription 25](#_Toc450524949)

[Table 4‑7 description for domain class Workout 25](#_Toc450524950)

[Table 4‑8 description for domain class Workout 26](#_Toc450524951)

[Table 4‑9 description for domain class Workout 26](#_Toc450524952)

[Table 4‑10 description for domain class ExerciseDescription 26](#_Toc450524953)

[Table 4‑11 description for domain class OnlineContent 26](#_Toc450524954)

[Table 4‑12 description for domain class Feed 26](#_Toc450524955)

[Table 5‑1 UC1 Main 1 - selectOwnWorkout(username : String) 28](#_Toc450524956)

[Table 5‑2 UC1 Main 1 - selectExercise(exercise : Exercise, intense : Intense) 28](#_Toc450524957)

[Table 5‑3 UC1 Main 1 - saveWorkout(answer : int) 28](#_Toc450524958)

[Table 5‑4 UC1 Main 1 - startWorkout() 28](#_Toc450524959)

[Table 5‑5 UC1 Main 1 - confirmFinishedExercise(answer : boolean) 28](#_Toc450524960)

[Table 5‑6 UC1 Main 1 - confirmFinishedWorkout(answer : boolean) 29](#_Toc450524961)

[Table 5‑7 UC1 Main 2 - selectPrederterminedWorkout(username : String) 30](#_Toc450524962)

[Table 5‑8 UC1 Main 2 - selectExistingWorkout(exerciseList : List<Exercise>) 30](#_Toc450524963)

[Table 5‑9 UC1 Main 2 - startWorkout() 30](#_Toc450524964)

[Table 5‑10 UC1 Main 2 - confirmFinishedExercise(answer : boolean) 30](#_Toc450524965)

[Table 5‑11 UC1 Main 2 - confirmFinishedWorkout(answer : boolean) 30](#_Toc450524966)

[Table 5‑12 UC2 - startNewDiet() 31](#_Toc450524967)

[Table 5‑13 UC2 - addFood(food, servingsize, Time) 32](#_Toc450524968)

[Table 5‑14 UC3 - accessDatabase () 33](#_Toc450524969)

[Table 5‑15 UC3 - selectContent() 34](#_Toc450524970)

[Table 5‑16 UC3 - giveContent() 34](#_Toc450524971)

[Table 5‑17 UC3 - interactContent() 34](#_Toc450524972)

[Table 5‑18 UC4 - accessDatabase() 35](#_Toc450524973)

[Table 5‑19 UC4 - selectProgram(programType) 35](#_Toc450524974)

[Table 5‑20 UC4 - highlightContent() 36](#_Toc450524975)

[Table 5‑21 UC4 - selectFile(File) 36](#_Toc450524976)

[Table 5‑22 UC4 - exitDatabase() 36](#_Toc450524977)

[Table 6‑1 UC1 GRASP Main1 Operation 1 37](#_Toc450524978)

[Table 6‑2 UC1 GRASP Main1 Operation 2 38](#_Toc450524979)

[Table 6‑3 UC1 GRASP Main1 Operation 3 39](#_Toc450524980)

[Table 6‑4 UC1 GRASP Main1 Operation 4 40](#_Toc450524981)

[Table 6‑5 UC1 GRASP Main1 Operation 5 41](#_Toc450524982)

[Table 6‑6 UC1 GRASP Main1 Operation 6 42](#_Toc450524983)

[Table 6‑7 UC1 GRASP Main2 Operation 1 43](#_Toc450524984)

[Table 6‑8 UC1 GRASP Main2 Operation 2 44](#_Toc450524985)

[Table 6‑9 UC1 GRASP Main2 Operation 3 45](#_Toc450524986)

[Table 6‑10 UC1 GRASP Main2 Operation 4 46](#_Toc450524987)

[Table 6‑11 UC1 GRASP Main2 Operation 5 47](#_Toc450524988)

[Table 6‑12 UC2 GRASP Operation 1 48](#_Toc450524989)

[Table 6‑13 UC2 GRASP Operation 2 49](#_Toc450524990)

[Table 6‑14 UC3 GRASP Operation 1 50](#_Toc450524991)

[Table 6‑15 UC3 GRASP Operation 2 51](#_Toc450524992)

[Table 6‑16 UC3 GRASP Operation 2 52](#_Toc450524993)

[Table 6‑17 UC4 GRASP Operation 1 53](#_Toc450524994)

[Table 6‑18 UC4 GRASP Operation 2 54](#_Toc450524995)

[Table 6‑19 UC4 GRASP Operation 3 55](#_Toc450524996)

[Table 6‑20 UC4 GRASP Operation 4 56](#_Toc450524997)

[Table 6‑21 UC4 GRASP Operation 5 57](#_Toc450524998)