Exercise 1

Group number: 22

Trym Grande

Hans Kristian Granli

Peer group number: 21

1. Introduction

We started by browsing the provided website and looked for feature candidates. These were written down after a brainstorming session with the peer group, and exchanged with them. We then elaborated on the features we received by separating into functional and non-functional requirements, as well as implementing relevant use cases and goals for the features.

2. Requirements, Goals and Use Cases

2.1 Functional Requirements Specification

RequirementID	Description
FR1	Generate password reset link
FR2	Update user's password
FR3	Like an exercise
FR4	Remove like from exercise
FR5	Upload a file to server
FR6	Link between exercise and file
FR7	Download file related to exercise
FR8	Parse file related to exercise and handle error

2.1.1 Reset Password

- Users who have forgotten their password and need a way to gain access to their account
- Users should be able to change their password if their password has been breached on another website

2.1.2 Like exercises

- Users should be able to like other users's public exercises
- Coaches should be able to like their athlete's private exercises

2.1.3 Upload & Parse traning data

- Athletes should be able to back up their workouts online in order to prevent data loss.
- Athletes should be able to share the workout data with their coach.
- Athletes should be able to view/edit their training data inside the application.

2.2 Goals

Reset password

As a *user* I want to *reset my password* so that *I can regain access to my account.*

Like/unlike exercises

As a *user* I want to *be able to like/unlike an athelete's workout* so that *I can show my support*.

Upload and parse training data
As an *athlete* I want to *upload my training data* so that *me and my coach can view the data directly in the applicatiassn or download it, and analyze my effort*.

2.3 Non-functional Requirements

2.3.1 Reset password:

- We have to be certain that the password isn't reset without confirmation that the right person has been issuing the reset-command.
- The method has to properly update and be robust.

2.3.2 Like exercices:

- The system needs to respond quickly to user requests.
- Simplicity for greater user experience.

2.3.3 Upload and parse training data:

- The system has to be flexible to support multiple data formats.
- It must be simple to use and understand.
- The parsing must be robust.
- Clarity both in times of success and failure

2.4 Use Cases

UC1: Reset Password

ID and name	UC1 - Reset password
Primary Actor	Any user
Descripton	If a user wants to reset their password, or has forgotten their password, this function will allow them to regain access to their account.
Trigger	When not being logged in, the actor can enter their email address into a form called reset password.
Precondition	PRE1: The actor has a user connected to the given email address. PRE2: The user is not logged in
Postcondition	POST1: The old password is removed (does not grant access to the account) a new one is inserted in its place.
Normal flow	*1.1: Reset account password* 1. The user enters the given email address and presses enter 2. The server receives the request, and checks whether the email-address has a registered account, if not - go to *1.2* 3. The server generates a link where the user can enter a new password. This link is sent to the given email-address. 4. The user can then go to the given url and enter its desired password twice. 5. The server receives the request. If the password is not typed identically twice go to *1.3*. 6. The server updates the password, and sends a message to the user informing them that they can now log in to their account.
Alternative flows	*1.2 No registered account* 1. If there is no registered account on the given email address, the server will **not** send an error message to the user. *1.3 Password not identical* 1. If the passwords aren't identical, the server should deny the request and ask the user to type in the desired password again. 2. Once the user has done so, a new request is sent to the server. 3. If the request is invalid go back to *1.3*, if it is valid go to *1.1.6*
Exceptions	None

UC2: Like/unlike exercise

ID and name	UC2: Like/unlike exercises
Primary Actor	Any user
Description	A user will be able to like/unlike an exercise if he/she wants to, meaning that the given exercise will have an incremented/decremented number of likes.
Trigger	A user indicates he/she wants to like/unlike an exercise by clicking the corresponding button.
Preconditions	PRE-1. The user is logged in, in order to obtain the user ID. PRE-2. The user has access to an exercise.
Postconditions	POST-1. The exercise's like count is incremented/decremented. POST-2. The user is added/removed from the list of users that have liked the given exercise

Normal flow	*2.0: Like exercise* 1. If the user has already liked the exericse go to *2.1* 2. User presses the like button. 3. Server receives the request. 3. The server adds a link between the user_id and the exercise_id in its internal database, 4. The server responds with a success-message indicating that the exercise has been liked.
Alternative flows	*2.1: Remove like* 1. User presses the un-like button. 2. Server receives the request. 3. The server removes the link between the user_id and the exercise_id in its internal database, 4. The server responds with a success-message indicating that the exercise has been un-liked.
Exceptions	The user sends an invalid request relative to the like-state in the databse. (e.g. A user who has liked sends a new like)

UC3: Upload Training data

ID and Name	UC3 - Upload and parse training data
Primary Actor	Athletes
Descripton	A user can upload one or more files with traning data related to an exercise. The server can then parse the data to create a readable human format if the training data is a supported file format.
Trigger	After completing an exercise, an athlete can naviage to the page of the exercise and upload a digital log of the exercise.
Precondition	PRE-1. The athlete is logged in PRE-2. The athelte has registered the exercise PRE-3. The athlete has a log of the exercise on the active computer/phone
Postcondition	POST-1. The file is stored on the server POST-2. The file is either parsed, or the user is informed that the file cannot be parsed.
Normal Flow	*3.0: Upload a supported file format* 1. The athlete uploads the file to the webpage of the given exercise. 2. The server receives the file. 3. The server checks whether the file is of a supported format, if not go to *2.1* 4. The server parses the file and generates a readable webapge for the user to access.
Alternative Flows	*3.1: Upload an unsupported file format* 1. The server recognices that the file cannot be parsed, and will return a message, along with a download link for the file itself in place of the parsed content.
Exceptions	3.1.E1: The file of a valid format is corrupted, and cannot be parsed.

3. Evaluation of the requirements documents of our peer group

We gave our peer group the following proposals:

- View and edit user information (name, email, address etc.), and add profile picture.

- Register meal, meal nutrients and meal templates.
- Possibility to add events.
- Possibility to add friends, and allow friends to view exercise log.

After some discussion with us, our peer group ended up selecting the requirements involving "register Meal Nutrition", "view personal information", and "edit personal information". Since the nutrition features is large, we agreed that it could be split into two different use cases.

Both participants of group 21 attended all meetings regarding this exercise.

3.1 Functional Requirements

The functional requirements described are accurate to what was requested. They are short and consise, while keeping the essense of our users will need. It could, however, be less vague as to what exactly is required. Comparatively, the use cases and non-functional requirements are more detailed.

3.2 Use Cases

The use cases all follow the boilerplate outlined in the lecture.

The use case ID-2 has too broad of a postcondition however, "Input is validated" should not be given as a postcondition since it is a dedicated step in the normal flow with an error handler and alternative flow.

Use case ID-3 does however lack description for alternative flows and exceptions.

3.3 Non-functional Requirements

The non-functional requirements seem to be blurred between functional and non-functional requirements. Otherwise, they are well described.

3.4 Modelling goals, constraints and requirements

The goals outlined describe the goals of the requirements vaguely. They could be more specific, but they are acceptable for the requirements.