AALBORG UNIVERSITY
COPENHAGEN

Aalborg University Copenhagen
A.C. Meyers Vænge 15
2450 København SV

Secretary: Maiken Keller

Title: Automated Sequential Recommendations of Personalised Weight-Training Plans for Fitness Enthusiasts

Semester Theme: Master Thesis

Semester: 4th

Project Period: Fall 2019 - Spring 2020

Supervisor: Jannick Sørensen

Members: Michail Gratsias

Pages: 108

Finished: 30th of May 2020

Abstract:

The aim of this thesis is to address the problem of making automated recommendations of exercise plans for people involved in fitness and especially in training with weights, based on their personal preferences and existing training principles. Therefore, the following problem statement was formed:

How can a recommender-system application support fitness enthusiasts by producing automated and personalised weight-training exercise plans based on proven training principles?

Additionally to that, a few sub-questions were added, which lead to further investigation of the following topics: recommender techniques that can be used, fitness domain information that will be needed, ways to personalise the solution, applicable recommendation algorithms and how such an solution could be designed, implemented and tested.

The final solution prototype uses a hybrid recommendation system based on constraint based techniques and similarity heuristics to achieve the expected results.

TABLE OF CONTENTS

1. Introduction

Recommending a weight-training exercise program for a visitor of a gym club can be a complicated task that requires expert knowledge and guidance. There are so many factors that are involved and so many different kinds of exercises, that the intervention of a special trainer is often necessary so that many unwanted phenomena like injuries, drop of interest and stagnation can be avoided and the fitness enthusiast can happily continue with his training and goals.

This report has the purpose to examine a possible recommender system that can be used in the form of an application in order to make automated recommendations of appropriate exercise plans for gym enthusiasts, avoiding the pitfalls due to lack of enough experience or a lack of a personal trainer or just lack of time to figure this out on one's own.

Choosing a correct exercise program, (or a 'training routine' or 'workout' or 'plan') for athletes, can depend on various things: a) how new or advanced the person is in physical training, b) what kinds of equipment are available, c) whether the person is a man or a woman, d) their physical capabilities and many other factors.

Choosing the correct sequence of exercises depends also on various training principles and even on what exercises have been done previously, because we want to avoid overtraining certain muscles with too much training and we want to surprise our physiques with new exercises when we get stagnant and lose gains from a specific kind of exercise.

These types of principles are based on earlier research work from endocrinologists like Hans Seyle [12] as he described in his theory about the "General Adaption Syndrome (GAS)", where he explains: a) how the body responds to new physical stress, b) the body's attempt to adapt and get used to and adjusted to the stress and c) the stage of exhaustion where the body cannot take any further stress.

It is also based on the most recent "Linear and Non-linear Periodisation" approach method to physical training [13], where what is used is a periodic variation in exercises and weights in order to help muscles of weight-lifters to avoid plateaus and sticking points and continue to grow in strength.

Also, as a famous bodybuilding trainer once wrote: "**Part of constant growth is never allowing your body to fully adapt to one specific training routine.**" - [1, pg. 97]

So, the ability to have new exercise programs at your fingertips, with correct sequence of exercises that are also created with tested principles, when the old ones have stopped generating adequate results due to adaption, is a vital factor in the success of weight-training and the results that can be achieved with it.

1.1 Problem Formulation

Based on such a background of knowledge about physical training, our proposed solution should be able to constantly recommend training routines based on a user's training profile, and choose exercises based on proven training principles that need to be followed so that a desirable result can be attained.

From this description then, we can create our problem formulation as follows:

How can a recommender-system application support fitness enthusiasts by producing automated and personalised weight-training exercise plans based on proven training principles?

And to further enhance this main formulation we would also add the following sub-questions:

- Which recommender techniques can be used?
- What type of domain knowledge about fitness training will be needed?
- How can the application be personalised to the users?
- Which recommendation algorithms are applicable?
- How can the solution be designed, implemented and tested?

1.2 Motivation & Delimitations

The weight training domain is very broad and we will be describing later-on more extensively some of its most fundamental principles, but it is necessary to also describe our own choices of limitations from the beginning, because we will not be able to cover every aspect of this broad field in one single paper.

As we said in our introduction, we will concentrate in this work especially in the area of weight training (where there is also more personal familiarity and experience), but also other types of sport or fitness activities could be covered in similar works and recommendations could be drawn for them too. We are just not going to deal with any other fitness activities here in this current work.

Even the activity of weight training alone includes a lot more variables than we will try to encompass in this current work, which will only and simply concentrate in the *training exercises to be recommended to the users, as part of one or many exercise programs*.

Other variables not included in this work but which are also very relevant and important in doing weight training, include: 1) the amount of times each single exercise is being performed (also known as: exercise sets), 2) the amount of repetitions of motions in each set of exercises 3) the amount of weight (resistance to the motion) being applied to each set of the exercise performed, etc. And finally other factors to achieving good results from such training include aspects of: the person's nutrition and rest habits that accompany the overall training regiment. [4, pg.76]

So without forgetting the importance of all these other areas of research, in the current work we will concentrate in one of the most complicated and significant of these areas which is the selection of the different exercises and the overall exercise training 'routines' to be performed by the fitness enthusiast.

To explain the importance of this choice of concentration a little better, and what motivated us to deal with it, let's quote the creators of the "Fitbod" training application (which we will describe more extensively later in the section on "Existing Solutions") who proclaim the following:

"**Constructing progressive resistance training plans is hard. One reason why there's no defacto gym app, is the difficulty in constructing progressive resistance training plans. Consumer fitness tech today is like Google Maps without directions. We have data-rich maps of people's physical activity, but fail to help them navigate towards real results.**" - [5]

We can see from that, that past gym apps have generally failed to properly navigate their users with *progressive*, *automated* weight training plans. Previous efforts to resolve these problems had been through the use of personalised coaching sessions with live coaches - which could become quite expensive for some. Or by requiring the athlete or enthusiast to educate himself in manually developing his own training plans without external guidance - also quite hard for most.

Finally, there was also the option of letting the person select some hopefully good exercise routine from a huge and confusing library - which can include a good possibility of error in making the right choice.

These were the main reasons why it was decided that it is time we use our engineering knowledge to offer more creative solutions in this area.

Next we will describe the Methodology which we will follow and with which we will try to answer our main research question and sub-questions.

1.3 Methodology

Here we will take up our research approach and data collection and analysis techniques as well as the software development method used to arrive to a working prototype. This will give the reader a good idea of the methodology that we followed throughout the thesis.

1.3.1 Documentary Research

In order to answer a lot of our technical questions about Recommender systems, techniques and algorithms the collection of Secondary research data through Academic journals was quite necessary. [45]

Many research papers concerning Recommender systems were studied in order to have a good grounding on the various methods and the context in which they are being applied.

At the same time, there was a thorough study of the Weight-training domain through various books on the subject so that the important training principles can be extracted as well as detailed information about related exercises and training routines and how these affect the person's physical condition and muscle strength and development.

1.3.2 Qualitative Interviews

Another important part of the research was creating scenarios and semi-structured interviews in order to get a better understanding of user requirements and involve the users themselves in the research process. [46]

Five unmoderated interviews were conducted online and were recorded (see Appendix C and software attachments) where users had to test an original form of our prototype and answer questions which provided us with valuable information in the process of collection of Primary data and in the critique and analysis of our system.

Another three moderated final interviews were conducted after development on a high-fidelity prototype to validate our results and ensure our satisfaction of user requirements.

1.3.3 Data Analysis

The Primary and Secondary data collection were followed by the needed analysis in order to come up with the appropriate techniques that will answer our research questions and which will support our choices and decisions and help us form and validate our Requirements Specification.

After each cycle of analysis, design, critique, we would come up with a better prototype until we arrive at a final solution.

1.3.4 Software Development Process

The overall development process we used was based on a Prototyping Model (see Figure 1) where users are involved in the whole process and various prototypes of the system are developed and tested until they meet the user requirements. [44]

The original requirements which we have previously collected through background research, lead to a basic design which will end up into a prototype which will be critiqued by the users themselves. The results of that are used to analyse again and adjust the

requirements which will be followed by a new design which can be implemented and then tested.

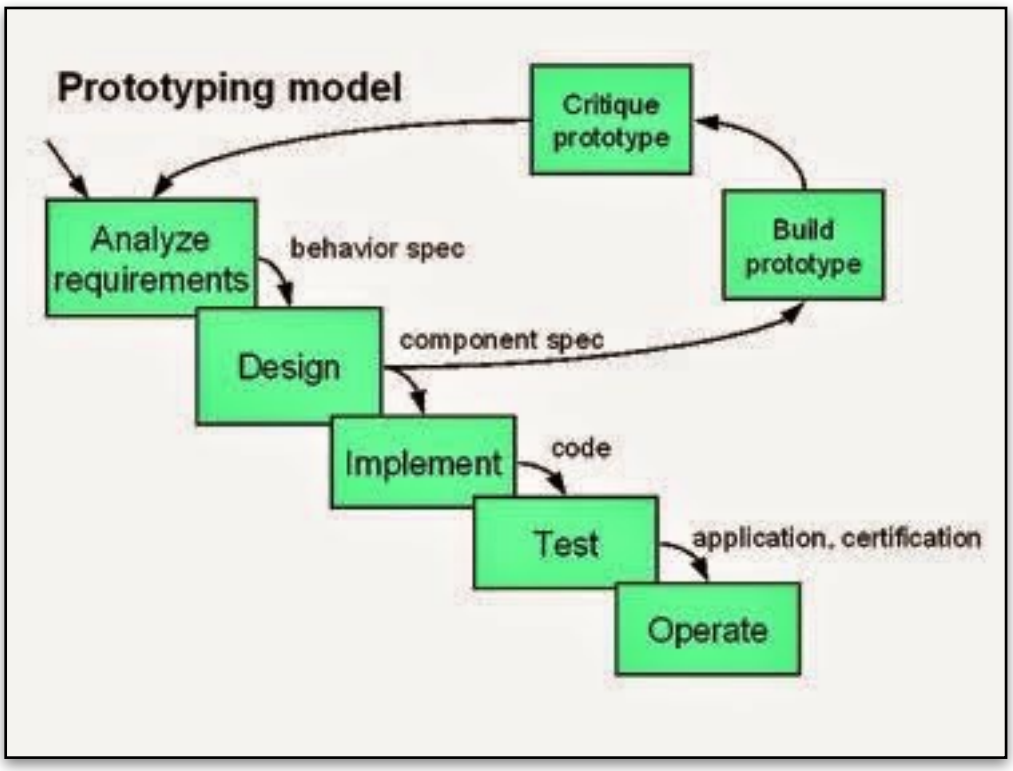


Figure 1, The Prototyping Model [44]

After completing the description of our project Methodology, we will be looking in the next chapter at background information about different recommender systems, the basic weight-training concepts and principles and we will also describe and compare existing software solutions in the market today.

2. Background

2.1 Recommender Systems and Techniques

Recommender systems in a few words can be described as systems that guide a user, usually in a personalised way, into choosing useful products from a big and often confusing pool of options. [14] Such products or items can be books or films or songs or things to buy etc.

When the recommendations are not personalised, then they just cover the most popular items. But when they are personalised, they try to predict what the specific user would most be interested in, based on his own personal preferences.

The first types of recommender systems designed, made social recommendations using a community of users that agreed with the current user in taste. Items that the close social community had liked, would be recommended to the current user. More recently, to deal with a huge amount of seemingly similar products creating an information overload, recommender systems use collected data about users and items and about previous user transactions (ratings, purchases, etc) to make their recommendations and they can record the users' feedback to these recommendations for improving their results. [25]

Apart from finding singe items that are useful or interesting to some user, recommender systems can also: recommend *whole sequences of items* (often called **Sequential** systems), like eg. recommending musical playlists to listen to. They can also allow the user to provide more specific information about himself and his likes or dislikes, they can allow users to rate or evaluate items they have experienced and they can even allow a user to choose and test different recommendation methods.

The main recommendation approaches [25] that are used today are the following:

2.1.1 Collaborative filtering (CF) technique

This refers to the recommendations from the social network of the user which we described earlier as the first and most popular type of recommender technique. All the users rate the items according to their degree of liking and recommendations are made in two different ways:

- **User-based CF technique**

In this technique, users who have rated items similarly to our current user (called "neighbours") are found and used in recommending to the current user items that he never had but which he may like also.

- **Item-based CF technique**

In this technique, we find items that are similar to items that the current user has liked (based on overall ratings) and we recommend those.

For the collaborative filtering technique to work best, we need a lot of ratings of the items by many users and this requires time to accumulate (known as the “cold start” problem). Also an important difference between the two CF methods is that item-based techniques tend to offer better accuracy in recommending items, while user-based techniques result in more originality in the recommendations given.

2.1.2 Content-based technique

In this technique, the system tries to recommend items that are similar to items the user liked in the past and is often applied in recommending textual documents (books, articles, movies, etc.). The attributes of the profile of the user will be matched with the attributes of the items in order to arrive to a recommendation.

The attributes of the items can be extracted as keywords from a longer description of these items (or the whole content), before a recommendation is made. [25]

This technique does not have the “cold-start” problem of the previous technique as it does not depend on user ratings, but on the other hand, it can only recommend items that are similar to the user profile.

2.1.3 Knowledge-based technique

Here, user requirements are elicited through explicit user feedback (eg. answering questions), and then a domain knowledge about how certain item features are useful to the user and meet the user preferences is exploited in order to arrive to recommendations. There are two ways to do knowledge-based filtering:

- **Case-based technique**

In this technique many past solved recommendation cases are collected and a similarity to past cases is calculated for any new recommendation case. Each case models a product or item to be recommended. If the user is not satisfied with the recommendation, he can modify his requirements and try again until an acceptable similar case is found. [26]

- **Constraint-based technique**

Instead of using a similarity metric, the constraint-based technique extracts predefined rules and principles of a domain and with these it matches the item features to the specific user profile and requirements in order to provide the proper recommendation results.

This technique is more suitable when we are dealing with complex products such as computers, financial services, etc., where traditional recommendation approaches based eg. on social collaboration may not be the best choice. In such complex cases, knowledge-based systems which elicit user requirements and exploit a knowledge-base of item properties would be a better choice and would also help overcome the common “cold-start” problems prevalent in collaborative systems [10], [14].

2.1.4 Hybrid Systems

In Hybrid systems more than one techniques can be used at the same time, usually in order to balance the disadvantages of one technique with the advantages of the other.

Hybrid systems can be one of various types [27], [53]. The three main ones are:

- **Weighted** Hybrid Systems, where each recommendation component is given a certain weight when combined with another one from a different technique in order to calculate the final rating of the items,

- **Mixed** Hybrid Systems, when recommendation results from different techniques are presented mixed together in a combined list,

- **Meta-level** Hybrid Systems, when one technique produces some model or output which then becomes the input for the next technique [27].

Having described the most common types of Recommender Systems and what they do, we will also mention some of the most important qualities or properties of these systems.

2.1.5 Properties of Recommender Systems

Recommender Systems have certain qualities that influence user satisfaction in using these systems. Here are some that may be most related to our system [41], [54]:

1. Prediction Accuracy

Recommender systems try to predict how much a user will like or dislike an item before they recommend it to them. Each system can use different methods and algorithms to accomplish that, as we have shortly described above. When their predictions are accurate, the user is generally satisfied with the recommendation.

2. Novelty

When we are talking about Novelty in recommendations we are talking about the recommendation of items that the user did not know about before. If a system only recommended items that the user knows, it would not be considered very useful to the user. So Novelty is a quality that is often very much sought after in such systems.

3. Diversity

In some cases, a user may not want to get similar items but may prefer a greater or lesser diversity in recommendations in order to discover more unusual items. If a system only recommends similar items, such a user would not be satisfied, so the element of diversity must often be taken into account.

4. Trust

Trust is the credibility of the user for the system and is usually achieved by using transparency and explanations of how the system works and what is the reasoning

behind the recommendations. When the users can understand the logic in the recommendations, they are more willing and feel more safe to use the system.

After this introduction into recommender systems we will continue by describing the basic terms and training principles of the Weight-training domain which we will be using later for our analysis and design.

2.2 Basic Terms & Principles of the Weight Training Domain

A clarification of some basic terms and principles of this domain, like: what is an exercise and a training routine, is necessary at this point as they may not be familiar terms to people not directly involved in this area of fitness activity.

2.2.1 Weight Training Basic Terms

Here we will discuss what are exercises, exercise programs, what equipment are being used, what are the different muscle-groups being trained etc.

First of all, **weight training Exercises** are specific movements performed repetitively with some resistance (weight) in order to train a specific muscle-group of the body. The movement repetitions are usually performed in sets of about 3 to 12 or more repetitions, with a short rest intervals between sets. (Eg. 3 sets of the exercise with 8 movement repetitions for each set is quite ‘common’.)

Fig 2, Some examples of free-weight equipment
(The exercise Bench on the left and a rack of Barbells on the right)

A **training program or routine** is defined as a whole list of exercises, with their sets and repetitions, performed within one training session (also called: a “workout”) lasting for about one hour, or covering several one-hour training sessions spread over several days, with rest periods between sessions usually within a framework of one week. The program can be repeated for several weeks until a specific training goal is achieved and then a

new program can be created and performed after that, for the next goal to be achieved. [2, pgs. 12-18]

Examples of training goals can be: to be stronger, to be able to lift 100kg (for weight lifters), or to lose 10 kg of fat, to gain 5 kg of body muscle, etc.

In resistance training or weight-training there are different types of equipment being used for performing exercises but they all fall under two main categories: 1) **free weights**, that you can carry with you and which give more opportunity of motion while requiring better ability to balance the weight and 2) **exercise machines**, which limit the potential motion you can make and do not require that you maintain any body balance while performing the exercises.

Examples of **free weight** equipment (see Figure 2) are: 1) **barbells**, which are long metal rods to which adjustable or fixed weights can be attached, 2) **dumbbells**, which are short barbells which you can hold in your hands, and 3) **exercise benches** where you can lie down or sit to perform exercises.

Exercise machines (see Figure 3) exist in many varieties and some of them are: 1) **Cable machines** which require the pull of a cable with a handle and 2) **Special machines** which allow a very limited range of possible motion, controlled by the machine.

Fig 3, Exercise machines (Cable machine on the left and Special machine on the right)
Image Source: [3]

After discussing common equipment, we will now discuss muscles and muscle-groups. The main body parts and **muscle groups** being exercised, with their Latin names, are the following (starting from the bigger muscle-groups to the smaller ones):

1) **The Thighs** - **Upper Legs** (this body part includes (in Latin terminology) the Quadriceps and Hamstrings muscles),
2) **The Back** (includes the Latisimus and Trapezius muscles),
3) **The Chest** (includes the Pectoralis muscles),
4) **The Shoulders** (includes the Deltoid muscles),
5) **The Outer Arms** (includes the Triceps muscles),
6) **The Inner Arms** (includes the Biceps muscles),
7) **The Forearms** (includes the Brachioradialis muscles) ,
8) **The Calves** - **Lower Legs** (includes the Soleus muscles) and
9) **The Midsection** (includes the Abdominal muscles or "Abs" in short).

There are exercises that train only specific muscle groups, and these are called: **isolation** exercises, while other exercises can train 2 or more different muscle groups at the same time and are called: **compound** exercises. [4, pg. 161] For example, with one compound exercise one can exercise both the leg and back muscles.

The workout routines (or programs) themselves are categorised based on how many workout sessions they include. The more advanced the athlete, the more sessions he will need in order to complete his workout routines. The reason for that is that he spends more time on each muscle group, doing more exercise sets, etc., so he needs to have more sessions often in different days in order to complete the whole exercise routine.

So we have the following types of routines or exercise plans:

1) **Full-body (1-day) routines**: here the person trains all the muscle-groups for a little bit (one set per exercise), in one training session. This is intended for beginners. A rest day follows the training session and then the program is repeated with different exercises this time, again covering the whole body.

2) **2-day split routines**: here the training of the 9 muscle groups is separated into two different days, so that the muscles can be trained individually more intensively each day. Then a rest day follows and the two-day routine is again repeated but with different exercises.

3) **3-day split routines**: the training program is split into three separate days and this is intended for people who have advanced to intermediate level. Again a rest day follows and the routine is repeated with the same exercises for several weeks.

4) **4-day split routines**: the program is split into four days or sessions, and is followed by a rest day before it gets repeated again.

5) **5-day split routines**: the program is split into five training sessions, (this is for more advanced athletes) and after a day of rest, the program will be repeated again.

6) **6-day split routines**: the program is split into six training sessions, followed by a rest day and then repeated as usual. [4, pgs. 86-91]

2.2.2 Basic Weight Training Principles

Now, we will discuss the most known and basic athletic training principles for weight training, as these were collected from our study of the domain:

- **PRINCIPLE #1. “In order for the muscles to grow, they must be repeatedly subjected to increasing forms of physical stress”,**

- **PRINCIPLE #2. “Sufficient rest time must be taken between workouts, for full recovery to take place”** - [4, pg.76]

The reason for these two principles, is that without increased stress you cannot achieve increases in strength in the muscles and it is of vital importance for the muscles to have adequate time to rest from this stress and grow in strength. This is based in the theory of the General Adaption Syndrome, as described in the Introduction. [12]

Since in our own case we are more interested in principles related to creating Exercise programs, we will point out the most relevant principles for creating such programs:

- **PRINCIPLE #3. The major (bigger) muscle groups should not be trained more than about 2 times a week, while smaller muscle groups (eg. Abs, Calves, Forearms) can be trained up to 6 six times a week.** [2, pg. 24] [4, pg.77] [12]

The reason for this principle is that rest and nutrition are important in order for the muscles to recuperate from a heavy workout and increase in strength. Daily intense training of the same big and major muscle groups will inevitably result in weaker and traumatised muscles, as there is not adequate time for the muscles to properly recuperate from the intense training. Smaller muscles on the other hand are more dense and recuperate much faster.

- **PRINCIPLE #4. Bigger muscle groups require about 4-5 different exercises to be fully trained while smaller ones require only 2-3 to be fully trained with an exercise routine.** [4, pg.67] [2, pg.73]

The reason for this principle is that bigger muscle groups (like Legs) are composed of more muscles that need to be stressed and trained in many different ways (using different exercises) in order to fully develop.

- **PRINCIPLE #5. Exercises for the same muscle group should be performed together (one after the other) for maximum gains to be obtained.** [1, pg.98]

By doing exercises of the same muscle group together or in a sequence, more blood in flushed into the same muscles, which results in accelerated growth. While when mixing muscle groups the blood leaves the area and the gains are smaller.

- **PRINCIPLE #6. Exercises for the bigger muscle groups should be done early in the training session, while small ones (eg. forearms) should be left last.** [1, pg.72-3]

The reason is that in the beginning of the training session one's energy levels are higher and it is easier to train the big muscles at that time. As the workout progresses one gets more and more tired and can only train small muscle groups in the end, which require smaller energy levels.

- **PRINCIPLE #7. As a person advances in training, he should split his routines into more and more separate days.** [4, pgs. 86-91]

The reason for that is, as a person advances and trains heavier, using more sets and weights, the muscles will require more training time and resting period between training sessions. The solution to that is to separate the training of different muscle groups in different training sessions, so that there will be more time to spend on each muscle and longer time intervals between the training of the same muscle groups and in this way there will be no fatigue and overtraining of the muscles, despite the heavier volume of training.

- **PRINCIPLE #8. Training routines should be changed occasionally (on the average every 4-6 weeks) in order to avoid plateaus and keep one interested and continuing to make gains.** [2, pg.70]

The reason for this, is that a routine followed for too long would make one adapted to it, and result in training plateaus and diminished gains. See: the General Adaption Syndrome and the Linear and Non-Linear Periodisation Approach. [12] [13] So the solution is to change the exercise programs after a few weeks and use different exercises.

From all these principles we can see that there are rules that can be extracted about HOW OFTEN different muscles should be trained, HOW MANY EXERCISES they require for their training, HOW MANY DAYS the program should take to complete, and what should be the CORRECT SEQUENCE of exercises in an exercise program.

We will be using these principles when we design our automated software solution.

2.3 Existing Software Solutions

In this last section we will review existing software solutions for recommending weight training programs and we will discuss their apparent advantages and disadvantages.

One can find many software solutions on the market offering recommendations of training programs and we chose some of them based on the following criteria:

1) We chose solutions with high-rating on the App Store or Google Play Store, as voted by the users, 2) we chose solutions based on their overall scope which was the "weight-training" type of fitness activity, 3) we chose solutions that we had personal familiarity and experience in their use for many months, and 4) we chose solutions based on the variety of approaches they take in solving the problem of exercise recommendations for users.

Here are the four chosen existing solutions:

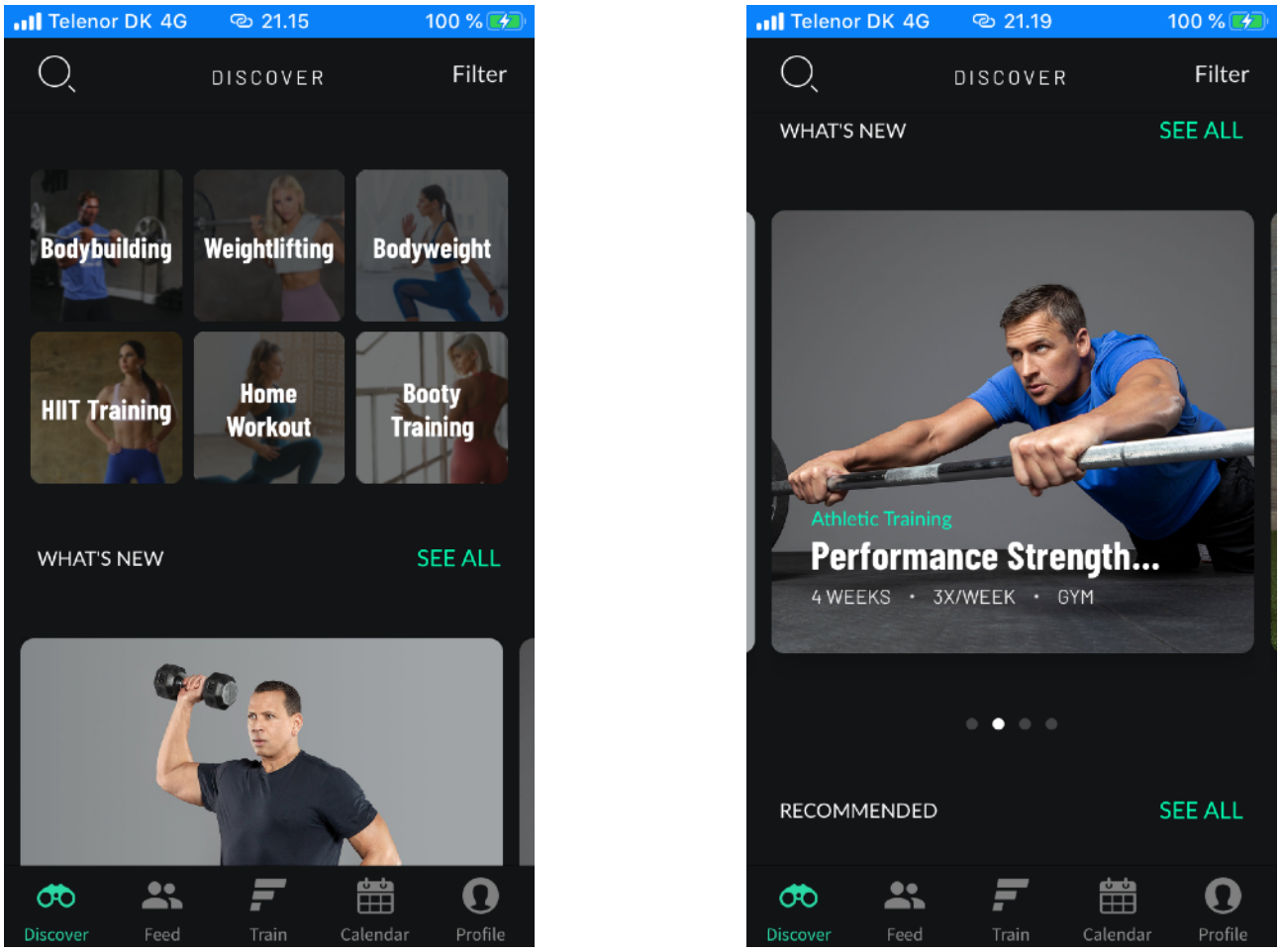


Figure 4, The Fitplan app

2.3.1 Fitplan - Gym and Home Workouts

Fitplan with a rating of 4.7 (see Fig.4) uses a variety of known personal coaches or trainers which you can choose from when you want to have a workout recommendation. These known coaches will recommend training programs which you can then do to achieve certain stated goals (eg. train at home, lose fat, etc.)

The exercise instruction is all done through HD videos and you can keep logs of your progress as you go along- also through your own smartwatch. All this functionality is offered for 7 days for free and then an annual subscription of $84 per year will be required in order to continue using it. [6]

It is obvious here that there is no specific automation with the plans recommended and little direction on how to choose the next most appropriate program. You are only expected to manually choose your favourite coach who will propose some routines to help you achieve some current fitness goal. You also do not create any personal profile of your own workout preferences.

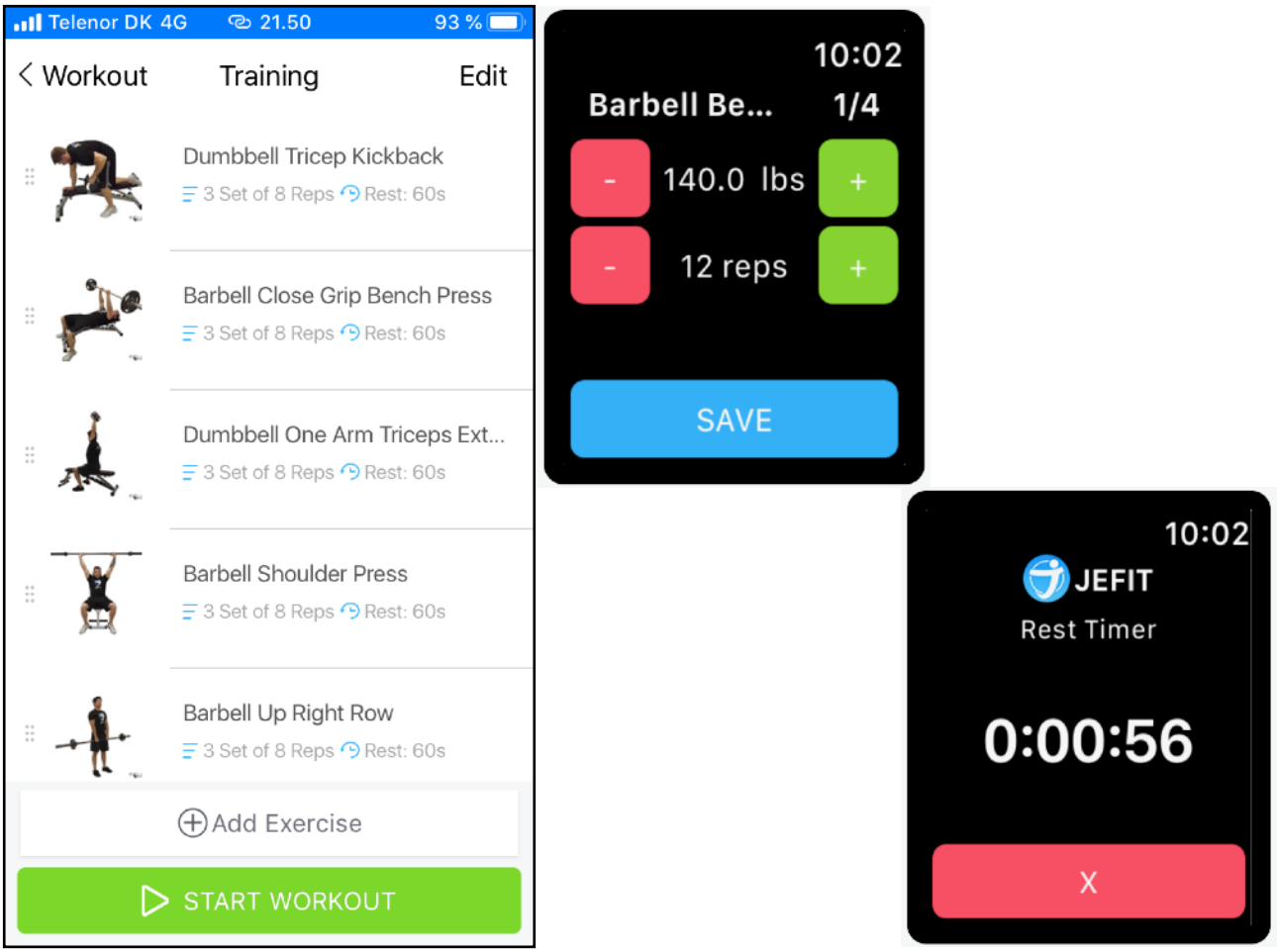


Figure 5, The JEFIT app on phone and watch

2.3.2 JEFIT - Workout Planner Gym Log

With this software (Fig.5) you are recommended specific popular (among users) weight training workout plans or you are recommended programs based on a few goals (like loosing fat, gaining muscle) or based on your own level of advancement. You also have the possibility to create your own workout programs (if you can).

JEFIT which has a rating of 4.5 on Apple store, holds a library of over a thousand possible exercises and you can log your workouts also through a smartwatch.

Additionally you can keep some history of your progress with graphs of how much weight you lifted and what exercises you did in the past. When you download the app, there is a short free period and then you can get a premium version with a $40 per year subscription. [7]

In this app we can see there is a big variety of programs and some amount of personalisation and possibility to change exercises that you cannot do or you do not like, in the proposed recommendations. But again the choice of programs is manual and not automated and there is no explanation of what, if any, principles the recommended programs are based on.

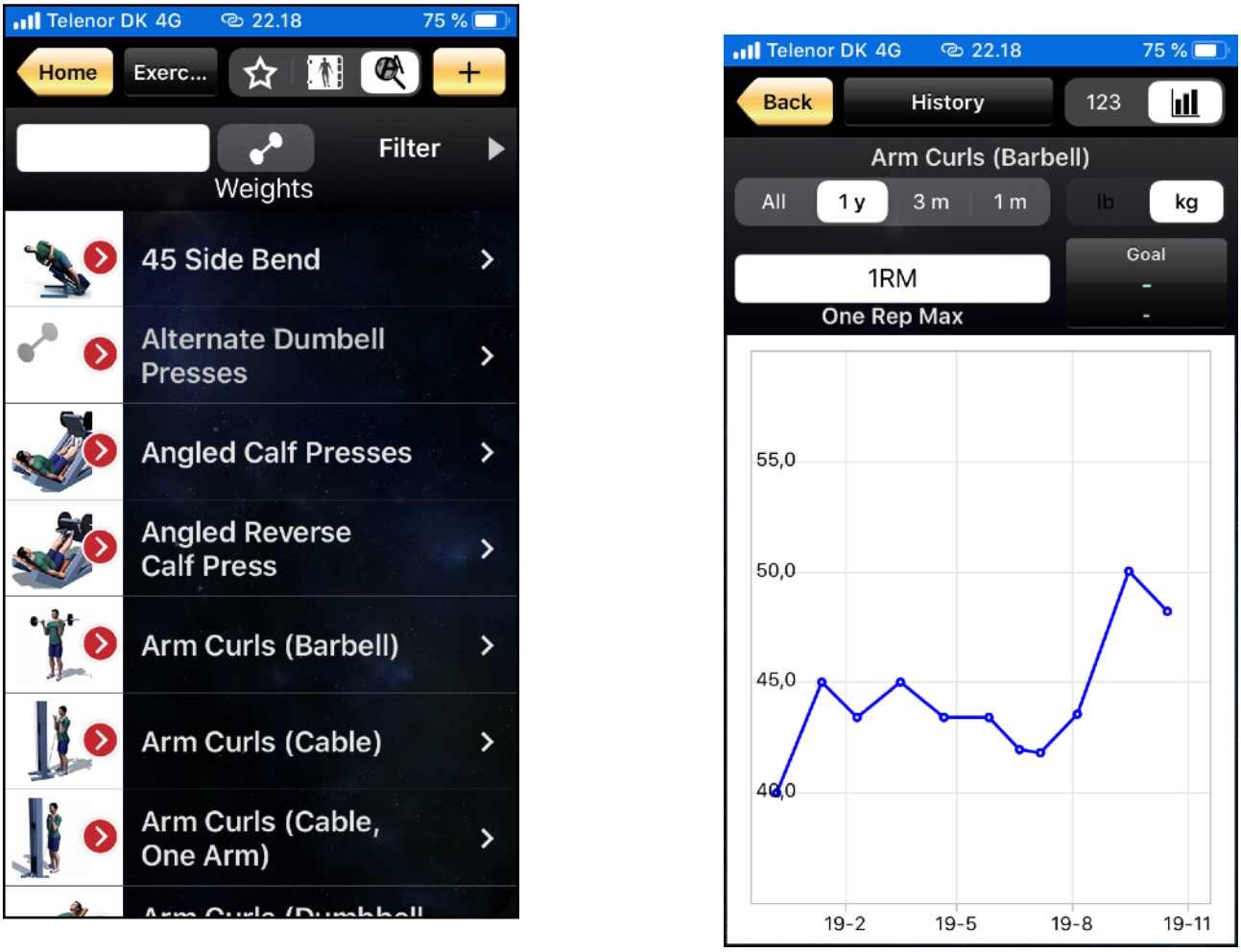


Figure 6, The GymGoal Pro

2.3.3 GymGoal Pro

This is an app we have used for quite some time. With this software (Fig.6) you have some ready programs recommended to you based on your level of advancement but you can also create your own programs and do manual changes in them as time goes on, which is the main emphasis of the app. It also keeps track of everything you need and offers graphs of all metrics so you can see how well you are achieving your fitness goals.

On GymGoal Pro which has a rating of 4.5, you can also use your smartwatch to track your workouts and you can record your heart rate and GPS locations. A big plus is that there is NO yearly subscription needed, but you only pay a fixed price of $9 for downloading the app. [8]

Definitely we have some of the best tracking features here and perhaps the best payment plan. But again there is no personalisation of profile and no continuous recommendations for your next workout routine. It all has to be done manually, and the app relies a lot on your personal knowledge and experience.

2.3.4 FitBod - Weight Lifting Workout

This final weight training software (Fig.7) which we also used for some months apparently adjusts to your current level of ability and training equipment and eliminates any guesswork in creating workout sessions by using its own AI capabilities. Small HD demo videos show you all the movements and a body-heat map is offered where you see the impact of your workout on your body. In this way you can see which muscles need more rest before they are trained again, by tracking muscle recovery percentages. [9]

The app which has a rating of 4.7 on the App Store seems to be the most advanced in terms of automation from all those that we tested.

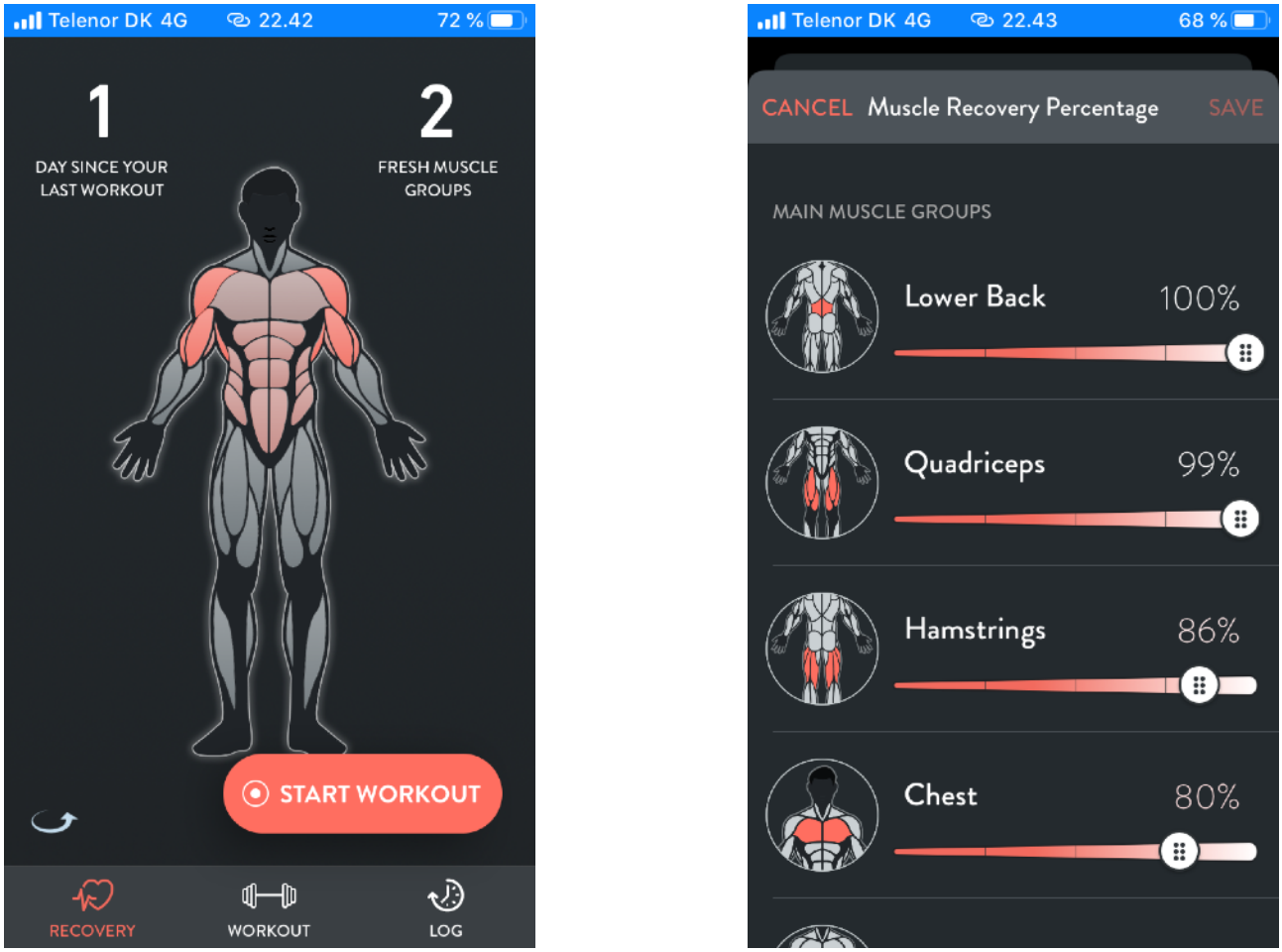


Figure 7, Fitbod app with muscle heat-map and muscle recovery states

Within the app, you can also create your own personal profile including: available equipment, fitness goal, fitness experience, excluded exercises, etc. (see Appendix D for full list) and following that, you are recommended automated, sequential workout plans.

When you go to their website, you can read that the recommendations are based on your body-heat map and training periodisation principles [13]. And you are also allowed to make fine-grained changes to the final recommendations.

One obvious missing elements in this software seems to be some visualisation graphs of your progress over time, while reaching your fitness goals, which do not yet exist because the app is still new on the market and under development. There is a 30 day free testing period offered and then a $60 yearly subscription plan.

2.3.5 Conclusions

From this analysis of the state-of-the-art applications, (see Table 1) we can see that most apps do not keep a personal profile based on which to give their recommendations and only 1 application on the market incorporates the principle of recommending automated and sequential workout routines based on personal preferences and a muscle recuperation scheme.

Most of the apps recommend workouts based just on user popularity or popular coaches and maybe also based on a few training goals. The problem with popular workouts is that they tend to not be very personalised to the individual and often they require changes to fit the current user.

In terms of presentation, there is a wide variety of media being used for showing how exercises are performed, starting from simple text descriptions and expanding to photos, GIFs, and HD video clips. Some apps keep a history of past workouts and can provide graphs showing progress through time, and other apps don't.

App / Features	Pgm Automation	Recommendation Basis	Fine-Tuning	Presentation
Fitplan	No	Favourite Coach, Training Goal	No	HD Videos, No Graphs
Jefit	No	Popularity, Some Personal Preferences	Yes	Pictures, Graphs
GymGoal Pro	No	Training Goal	Yes	GIFs, Graphs
Fitbod	Yes	Many Personal Preferences, Periodisation, Fresh Muscle Groups (Body Heat-Map)	Yes	Small HD videos, No Graphs

Table 1

Finally, we can conclude that getting correct automated personalised recommendations in the absence of a personal coach is not an easy task, but it is an important area in helping fitness enthusiasts to progress and achieve their goals, and not become lost on the way.

3. Analysis

Here we will inspect what are the most applicable recommendation techniques for our system and what its functionality is supposed to be. We will conduct some low-fi tests with users and we will arrive at a requirement specification with argumentation on the rationale and prioritisation of the requirements with the MoSCoW technique.

3.1 User Scenarios

Scenarios describe a set of tasks that users may want to perform and give an idea of how technology can support those tasks. They are an important starting point from where later design decisions can be made, [28] and for this reason, we will describe here two User Scenarios that gives an idea of how we could implement our exercise recommendation system.

Scenario 1 - Beginner

Peter is a fitness enthusiast who just subscribed to his local gym and plans to do regular weight training. As he is not so familiar with the gym and what exercises he is supposed to do, he found an app which is supposed to recommend him how to get going.

The app has a Settings section, where he is asked to give some information about himself and his gym in order for the recommendations to be adjusted for himself and his current environment. Peter gives the needed information about his fitness status (eg. male, beginner), including what equipment are available to him in his gym and what exercises he can do, and as a result of that he receives a recommendation of his first exercise program!

He is happy to see that the program is tailor-made for him and that he can actually perform at his gym all the exercises that are described in it. He continues with his training satisfied in knowing that he did not need to hire a personal trainer for receiving such a fitting program recommendation.

Scenario 2 - Advanced

Hans has finished with his earlier exercise program which he has been doing for several weeks now and he is starting to realise that he has nothing more to gain from that and that he needs something new.

But there is nothing to worry about, since with the touch of a button he can update his level of advancement and be recommended a new program to do, that is following the one he just did. He is happy to see that the new program is full of interesting and new exercises. Also he realises that the exercises are created with such a sequence that he can do them without a problem and without getting tired or giving up in the middle.

But as he inspects the exercises listed in his new program, he finds one exercise that he is not sure he would be very willing to perform due to a temporary injury.

That's not a big problem, because he can tap on the exercise and be given the possibility to replace it with a different one that is just as good as the last and that he is willing to do! After choosing the one that he likes from the list, his overall program is adjusted and he can just go on.

He is surprised that his training becomes smooth and easy just like having some expert to guide him all the time on what to do next! That's a good relief for him and he continues his training happily.

3.2 Use Cases

From the above scenarios, we can already see the main Use Cases that we will be involved in (see Figure 8).

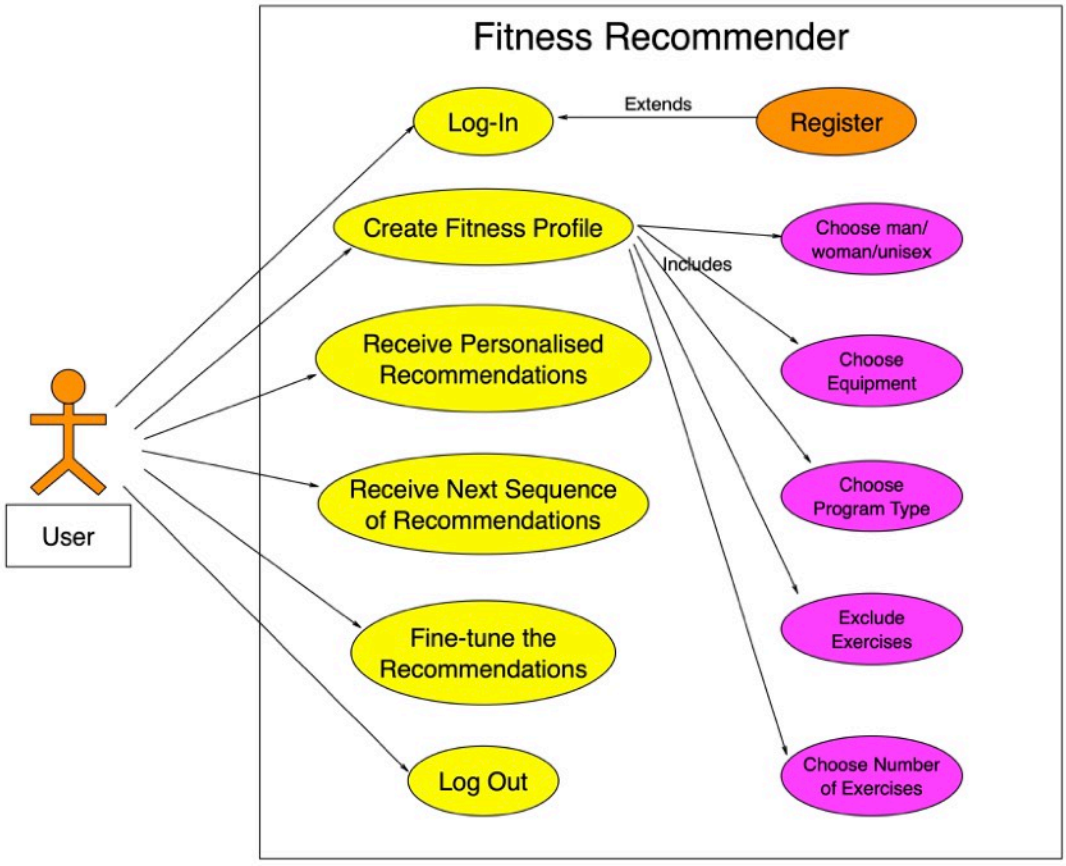


Figure 8, Use Case Diagram (made with UMLModelerPro)

After Log-In, our solution will provide the option of creating a User training profile where the user can personalise the application to his own fitness preferences. This is different from most of the solutions on the market today (see Section 2.3) which are generally not very personalised to their users.

The personalisation options include: choice of recommendations for men, women or unisex, choice of equipment, choice of program type, choice of exercises to be

excluded and choice of number of exercises per muscle group (see later in section 3.4.2 for full explanation).

The user will also be able to receive automated recommendations based on his own preferences (or the app default settings if no preferences mentioned) and the recommendations will be given as a sequence of exercises, collected day by day based on proven principles as discussed before, until the program is complete.

The press of a button will create a new sequence, so that the user will not have to choose anything more by himself. This is different from existing solutions which require the user to make a lot of manual choices, may not incorporate specific straining principles in the choice of exercise sequences or use popularity as the way to recommend the next exercise program.

Finally the user will have the ability to fine-tune the received recommendations by making small, safe adjustments to them and will also be able to Log Out while his personal changes are recorded.

In the next sections, we will analyse further the main use-cases by describing the most relevant recommendation techniques and algorithms and also how exactly the personalisation can be done.

3.3 Choice of Recommendation Techniques

In this section we will discuss the most relevant recommendation techniques for our own case and their details. We have discussed the most common techniques for recommendation in our previous chapter and their main pros and cons. Now it is time to concentrate in what will be more relevant for us and we will show the results of our analysis in a table (see Table 2).

1. The use of Collaborative filtering techniques (as described in section 2.1.1) would require that users give a rating to various exercise programs and we would recommend those programs that similar users have liked to perform in the past (highest rating). Although this could work to a certain extent and is being used in some systems, it is definitely not the best way for us to go about it.

The reason for that is, that the user would receive exercise programs that worked well for other users, but this does not guarantee that the programs would work for our current user too. On the contrary, these programs could cause him quite some personal difficulties if the only criteria were whether some friend liked the program or not. As a simple example, let's say that our user's gym does not even have the equipment that are used in the friend's program, so then the program cannot be done by the current user.

2. Content-based techniques on the other hand (as described in section 2.1.2) would recommend exercise programs that are similar to those the user had in the past, based on his previous behavior. The recommendations given in this way would be limited and would not result in enough variation and real successful training.

3. Knowledge-based techniques and specifically Constraint-based techniques are more suitable for recommending complex products and our exercise programs are definitely complex and cannot be easily recommended by the previous techniques. By matching all the rules and principles of the weight-training domain to the user preferences, we can arrive at a great variation of suitable recommendations.

4. The Case-based techniques on the other hand would not be applicable, because we are not necessarily interested in recommending only the limited programs we have in our library based on similarity with user requirements. [14] We want to be able to construct and choose from a big variety of exercise programs.

5. Additionally to the above techniques, our recommender system would have to recommend programs with *long sequences of exercises* instead of just single exercises, because an exercise program is broken into many exercises for each muscle group and each muscle group in turn follows the next muscle group in an exact sequence.

Therefore we will need to use a technique that can provide us with a list or lists of exercises that are to be performed in a proper sequence, resulting in "Sequential" Recommendations of items, instead of recommendations of single items.

6. Finally, Hybrid Systems are recommender systems where different recommendation techniques are combined in some way for better results. As we need to combine *Constraint-based techniques* with techniques for *Sequential recommendations* we could say that Hybrid systems are applicable to our case.

Most specifically the *Meta-level* type hybrid systems would apply most, because we will have to input the personalised results of our constraint-based technique into our sequential technique for generating our final exercise sequences. [27]

These choices we have made can be seen in Table 2.

Techni-que	Collabo-rative	Content-based	Knowledge-based		Sequen-tial	Hybrid Systems		
			Case-based	Constrai nt based		Weight-ed	Mixed	Meta-level
Appli-cability	✗	✗	✗	✓	✓	✗	✗	✓

Table 2, Choice of Recommendation techniques

As a next step to choosing the most applicable techniques, we will examine in more detail what types of algorithms are included in these techniques, so that we can find the most applicable ones for our case, just as we did with the recommendation techniques already.

3.3.1 Algorithm for Constraint-based systems

We already discussed how constrained-based systems, which are a type of knowledge-based system, require that an engineer encodes a deep knowledge of a complex domain into an executable representation and then relate customer requirements with item properties. What follows is the generic algorithm of how this happens.

Firstly, [15] one would collect all possible properties of the Product to be recommended into a variable, which we can call Vprod, and in our case it would include a list of all possible exercises that can be performed and be part of any exercise program together with their specific properties eg. what equipment they require etc.

Then one would collect all the possible Customer properties which need to be instantiated and personalised into the specific customer requirements. This variable we can call: Vc and in our case it would include for example all the possible equipment that different customers may be using or missing in performing the various exercises.

When the Customer sets his own requirements, eg. what equipments he can use, these instantiate the: Vc variable into specific constraints: Cc, for example: cable-machine equipment. Another constraint: Cr can represent possible restrictions on certain combinations of customer properties, eg. female customers should not use barbells as equipment.

And another constraint: Cf can represent possible filter conditions between Products and Customer properties eg. certain exercises cannot be performed without cable equipment being available. Finally the: Cprod constant would represent an instantiation of Vprod and in our case would represent a specific set of exercises which CAN be recommended as part of an exercise program. [15]

So the algorithm used in Constraint-based recommender systems requires that these types of constants or constraints (Cc, Cr, Cf, Cprod) and variables (Vc, Vprod) are used in solving a *constraint satisfaction problem* (CSP), through instantiating the variables and fulfilling all of the constraints. [14], [16]

So, an acceptable recommendation (Cprod) in a constraint-based recommender system would then be: an instantiation of the variables Vc and Vprod (Cc, Cprod) which does not violate any of the set constraints (Cr, Cf). [15]

3.3.2 Similarity algorithms

In recommender systems similarity algorithms are very widely used, as discussed already in section 2.1. One very common metric of similarity used in such systems (but not the only one) is the Cosine Similarity. [43]

One can create vectors from various attributes of items and compute the similarity of these vectors. This can be given as the cosine of the angle between the vectors, from which came the name: Cosine Similarity.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Figure 9, Cosine Similarity of vectors A and B [42]

The formula can be seen in Figure 9, where A and B are the vectors, θ is the angle between the vectors, n is the dimension of the vectors and Ai and Bi are the various components of the vectors. Also ||A|| and ||B|| are called the “lengths” of the vectors, or “norms”.

When the vectors are “similar”, they would then “point” in the same direction and the angle between them would be zero, while the cosine similarity cos(θ) would be 1.

3.3.3 Algorithms for Sequential Recommendations

As we already pointed out, sometimes the items to be recommended are consumed in a sequence, for example: songs in music playlists, or exercises in training exercise routines.

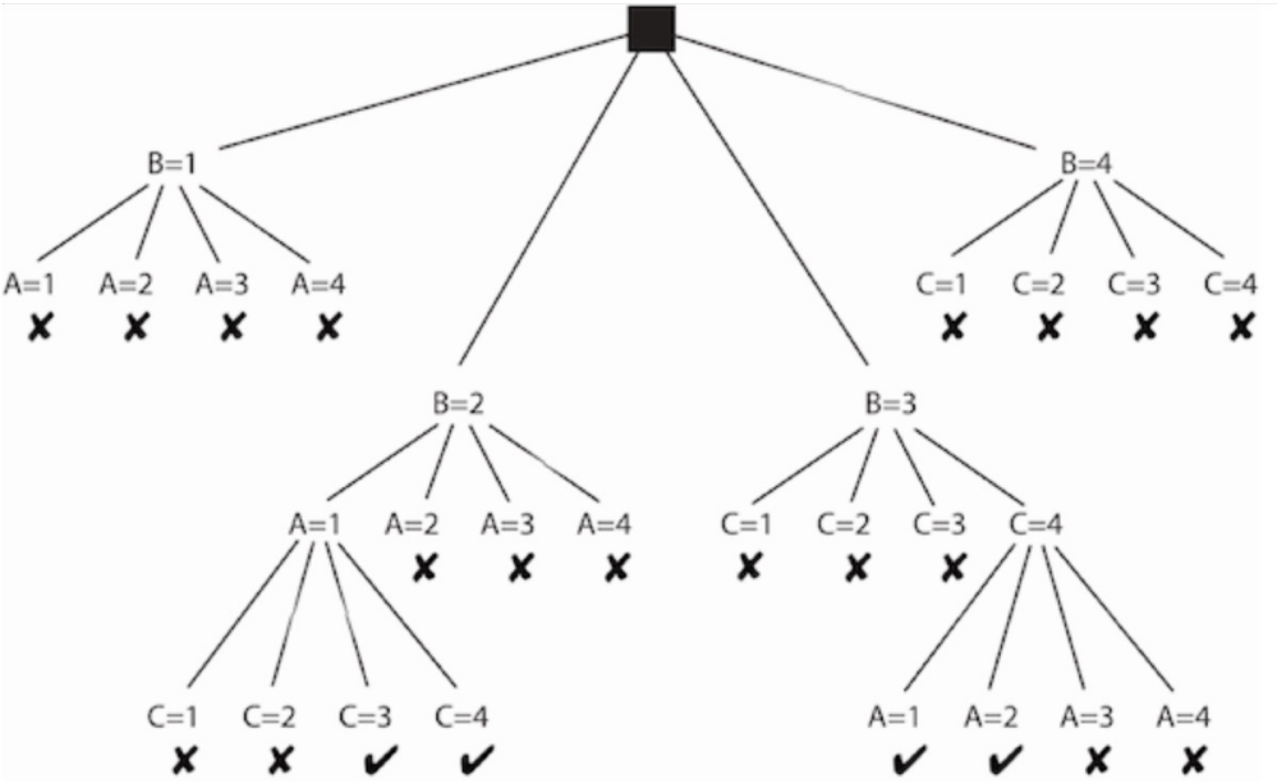


Figure 10, Search tree to solve a Constraint Satisfaction Problem (CSP) [29]

In order to automatically generate such sequences of items as recommendations there are different methods and algorithms that can be potentially used. Here we will examine the three main ones which are used among others in solving song-playlist generation problems: 1) the Constraint satisfaction techniques, 2) the Similarity heuristics and 3) the Machine Learning approaches. [11]

1. **Constraint satisfaction techniques** are used in sequential recommendations when the sequence needs to conform to certain “rules” and constraints due to regulations and accepted practices.

The sequence is a list of variables Vi where each variable represents the i-th item to be recommended on the sequence. The possible values of the variables can be taken from a long catalog and need to satisfy different types of constraints in order to become part of the list.

The various types of constraints used here, can be: Unary constraints that apply to single recommended items, Binary constraints that apply to two adjacent recommended items and Global constraints that apply to all of the items in the list. [17]

As an example, let's consider 3 variables: A, B and C which take values from the domain {1, 2, 3, 4}. If we set as Binary constraints that A<B and B<C then we can arrive at 4 possible solutions for the *sequence* A-B-C, as can be seen in the Search Tree of Figure 10. By going through all the combinations we can see that the only solutions to the problem are: [1,2,3], [1,2,4], [1,3,4] and [2,3,4].

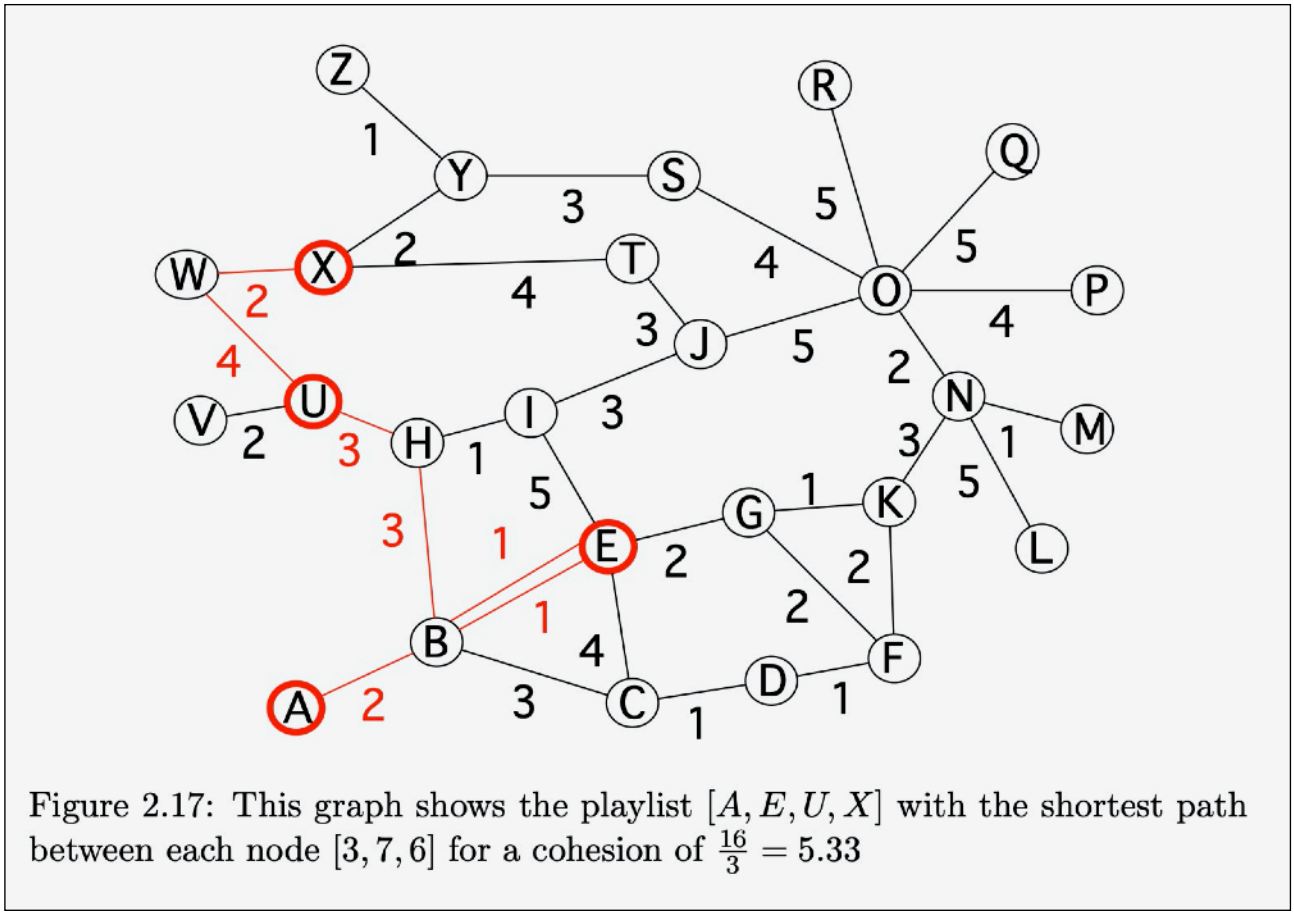


Figure 2.17: This graph shows the playlist $[A, E, U, X]$ with the shortest path between each node $[3, 7, 6]$ for a cohesion of $\frac{16}{3} = 5.33$

Fig. 11, Shortest path between nodes or seeds of songs [20]

These constraint satisfaction techniques are always combinatorial and constitute a feasibility problem. [11]

2. **Similarity heuristics** are practical algorithms which in their simplest form are used to compute sequences based on similarity to a “seed” item, eg. find songs of the same/similar music genre, rhythm etc. as the one of the seed song. [18] (A seed here is an item that is input into an algorithm which will then generate further items).

In their most advanced and sophisticated approaches, a graph is constructed over various seed items and the sequences are computed by calculating shortest paths between seeds. The whole idea here is to use a path-finding algorithm like the “travelling salesman” algorithm in order to navigate between seeds [19], [21], [22] (see Figure 11 for an example of a path finding algorithm applied to constructing a music playlist).

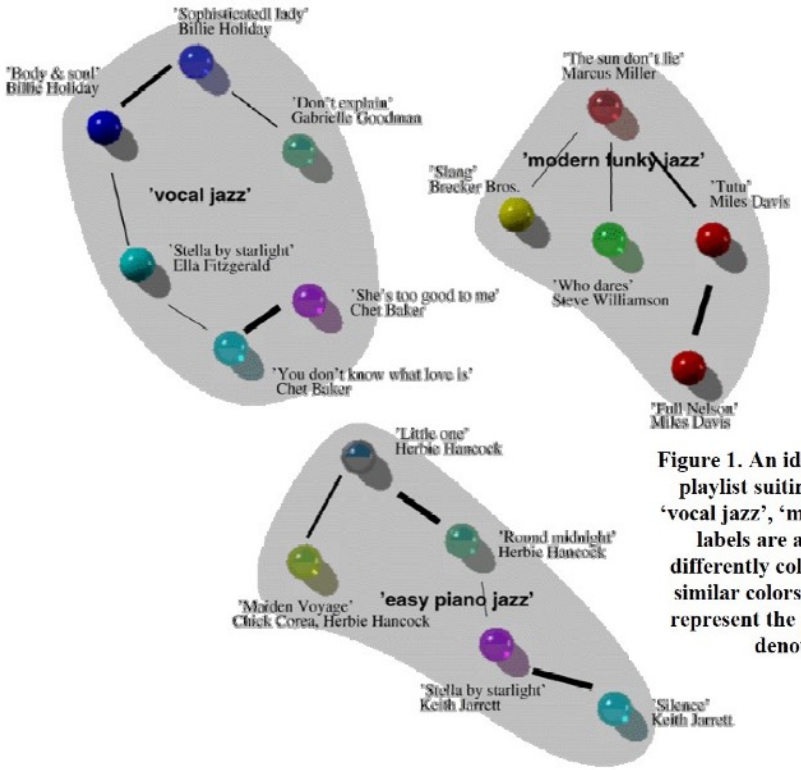


Figure 1. An ideal cluster result of songs that may represent a playlist suiting a particular context-of-use for listening to ‘vocal jazz’, ‘modern funky jazz’ or ‘easy piano jazz’ (cluster labels are added manually). Songs are represented by differently colored (or shaded) marbles. Similar songs have similar colors (shades). The lines connecting these marbles represent the grouping of songs in a cluster. The line width denotes the similarity between two songs.

Figure 12, Song cluster generation [24]

Also, with the technique of clustering items based on seeds, one can attain both the objective of coherence by maximising within-cluster similarity, and the objective of variation by using very different seeds. [24] (see Figure 12 for another playlist example with item clustering). In this way clustering can help us visualise the concept of sequences of items within sequences of items (the seeds of the clusters).

3. **Machine Learning approaches** include algorithms like Markov Chains and the K-nearest neighbours.

Markov chains describe the sequence of possible states (or events) where the probability of the next state depends only on the previous one. [30] What we want to find is what will be the next state in a sequence of possible states, based on known probabilities.

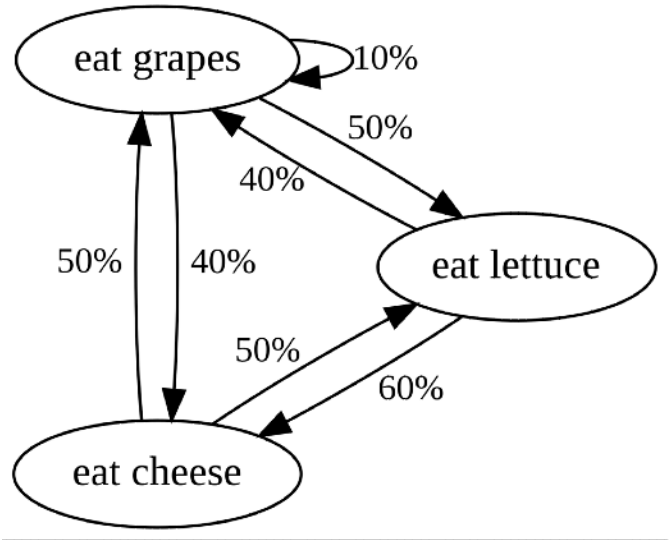


Figure 13, Markov chain with States and Probabilities [30]

An example is seen in Fig.13 where after knowing the current state and the probabilities of transitions to other states we can predict for example the dietary habits of some bird animal that eats only three kinds of food.

In order to optimize the model parameters (in this case the probabilities) a training set of observed state transitions needs to be used. From the training set one can create frequency counts (transition probabilities to the next state) and use them to give a correct weight to these transitions (or state sequences), in this way optimising the algorithm.

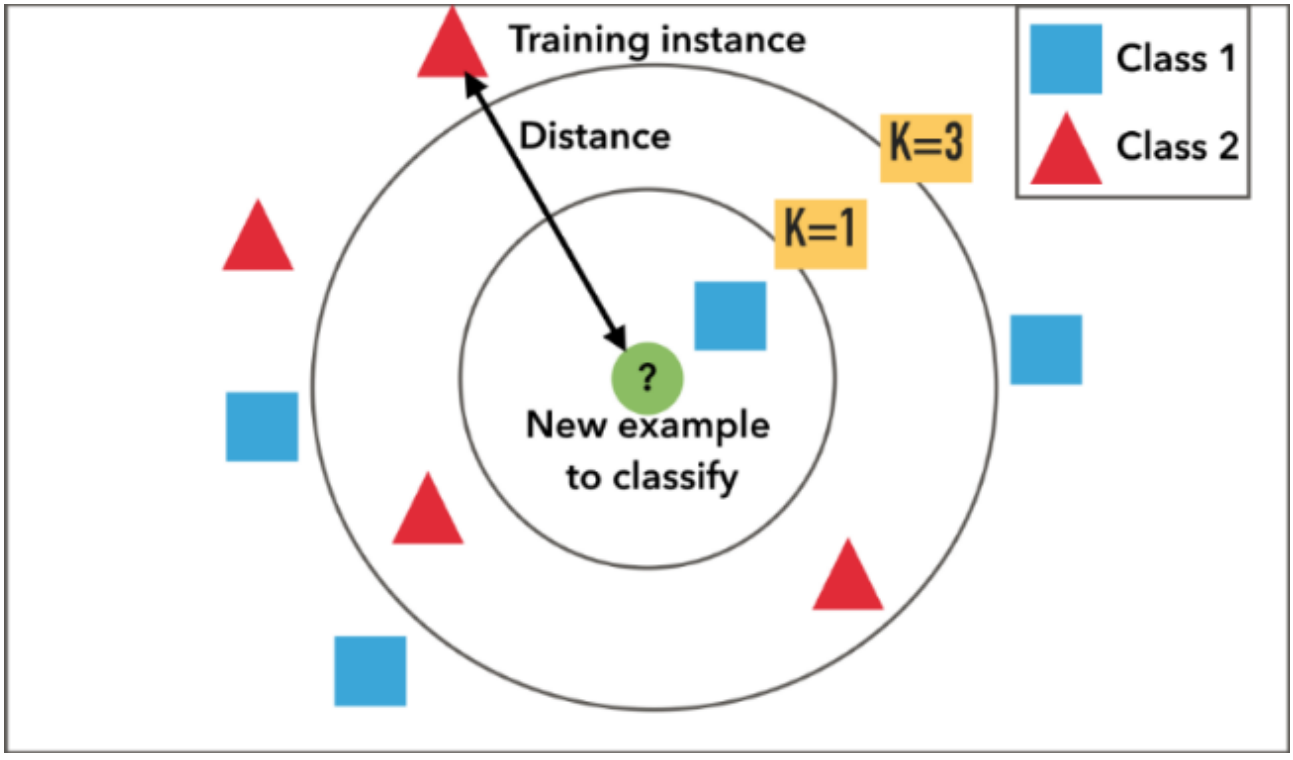


Figure 14, KNN example diagram [31]

In our case the states would represent exercises and the transitions would represent the sequences of exercises from one to the next inside an exercise program.

Another algorithm that could be used is the “K-nearest-neighbours” algorithm where we can start with a training set of sequences of items and find co-occurances of items in these sequences and then calculate the similarity (or nearness) between a these training sets. [23] Then, based on certain given “seed” items, one can find the best sequences that are most similar (or nearest) to the seeds. (See Figure 14 for an example of KNN finding the nearest neighbour of a given item for purposes of classification).

Such machine learning approaches, though, would tend to generate only previously observed (similar) sequences of high popularity (highest observed frequency or probability) and this would result in our case in recommending very popular programs where the diversity element would be missing. [10] We would also have to have a training set to train our algorithm too, which at this point we don’t.

In our case, we want to be able to set certain rules according to our training principles about how to choose and create sequences of exercises, so the constraint satisfaction techniques for sequential recommendations will be the most useful. We also want to be able to give different “seeds” of muscles for clustering exercises in different muscle groups, so the similarity heuristics with item clustering will also work well. And we finally want to be able to compute similarity between exercises, so the cosine similarity algorithms are useful also.

On the other hand, the machine learning approaches as we already explained, will not be appropriate at this stage to give us the results that we want.

So, based on the above analysis we have arrived at a choice of the most applicable algorithms that we will use for doing our recommendations. (see Table 3)

Algorithm	Constraint-based systems	Cosine Similarity	Constraint satisfaction techniques	Similarity Heuristics	Machine Learning approaches
Applicability					

Table 3, Choice of recommendation algorithms

In the next section we will analyse how we can start building a prototype for recommending exercise routines, with the first step being the creation of a knowledge base and methods to elicit customer requirements.

3.4 Constructing a Knowledge Base

3.4.1 Exercises & their combinations

As a starting point for our recommender system, it would be necessary to accumulate a knowledge base starting with all the potential training exercises that can be recommended in a weight-training exercise program and what the specific attributes of each exercise are (what equipment are needed, what muscle groups are affected by the exercise, etc.) This is a necessary part of the Constraint-based technique algorithm as has been discussed in chapter 3.2.1 and refers to the variable Vprod, where the product to be recommended is sequences of weight-training exercises or ”workout plans”.

Exercise	Muscle Group	Men/Females	Extra muscle groups	Variations of Angle	Equipment variation	Total Variations
Hip Extensions	Legs (Glutes)	Females	-	-	Cable Machine/ Special Machine	2
Hip Raise	Legs (Glutes)	Females	-	-	Barbell	1
Leg Curls	Legs (Hamstrings)	Men/Females	-	-	Cable Machine/ Special Machine	2
Stiff Leg Deadlifts	Legs (Hamstrings)	Men/Females	Back	-	Barbell/Dumbbell/ Smith Machine	3
Hip Abductions	Legs (Outer Quads)	Females	-	-	Cable Machine/ Special Machine	2
Hip Adductions	Legs (Inner Quads)	Females	-	-	Cable Machine/ Special Machine	2
Calf Raises	Calves	Men/Females	-	-	Barbell/Dumbbell/ Special Machine	3
Calf Presses	Calves	Men/Females	-	-	Special Machine	1
Seated Calf Raises	Calves	Men/Females	-	-	Barbell/Smith Machine/Special Machine	3
						165

Table 4, Weight-training Exercises & Properties Excerpt (see Appendix A)

After an in-depth study of available material about the domain [1] [2] [3], which included an encyclopaedia of weight training and other relevant books by recognised trainers, such a knowledge base was collected and the results can be seen partly in Table 4. (The full 6-page Table is available in Appendix A).

This table in its complete form contains 70 basic weight training exercises that cover all the main muscle-groups of the human body, together with 95 more exercise variations, which makes a total of 165 exercises which can be used as an input to our recommendation system for recommending weight-training programs.

These can be further broken down to: 23 chest exercises, 16 for outer arms, 22 for the back, 14 for inner-arms, 8 for forearms, 22 for the shoulders, 27 for mid-section, 26 for legs an 7 for calves. Their variations are dependant on the different types of training equipment can be used, on the bio-mechanical position of the body (angle of motion) during exercise and whether the exercise is intended more for male or female users or both.

The output of our system needs to be a training program with a recommended sequence of exercises that will be constructed in a way that follows the customer given requirements. When the workout goal has been accomplished (after some weeks), a new program can be recommended to follow up the last one.

Consulting our Principle #4 (see section 2.2.2) each training program will finally contain *by default* 32 exercises, which will cover all of the 9 muscle groups, in a proportion of: 4 Chest exercises, 5 Back exercises, 5 Thighs-Legs exercises, 4 Shoulders exercises, 3 Outer-Arm exercises, 3 Inner-Arm exercises, 3 Calves exercises, 3 Forearm exercises and 2 Abdominal exercises, which equal: 32 exercises in total.

To make sure that we have a knowledge base that can give us an *adequate* amount of recommendations with high degree of *novelty* (not the same exercises again and again), we will need to calculate for each exercise group: how many combinations **C** of distinct exercises **n** (which can only be used once in each program) can exist when taken in the default amount of **r** at a time, as described above per muscle group. This can be given by the combinatorial equation **nCr = n! / r! (n-r)!** as seen in Figure 15 [32]:

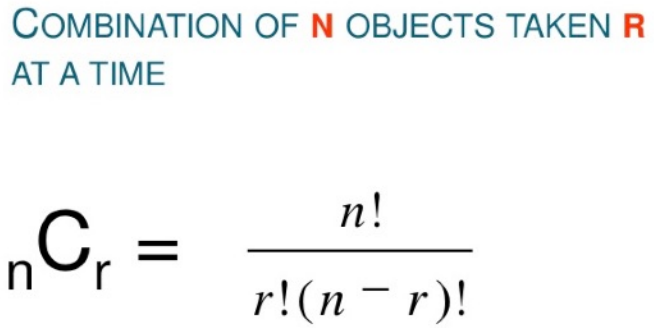


Figure 15, Combinatorial Equation [33]

So eg. for the chest muscles for which we have n=23 and r=4, we can have: 23!/ 4! (23-4)! = 8855 combinations of 4 chest exercises. Calculating that for each muscle group we get the combination amounts shown on Table 5.

Muscle Groups	Legs	Back	Chest	Shoulders	Triceps	Biceps	Abs	Forearms	Calves
Combi-nations	65780	26334	8855	7315	560	364	351	56	35

Table 5, Combinations of exercises per muscle group

When considering the exercises for each muscle group independent from each other, the total number of unique exercise programs that can be constructed then from our knowledge-base are: 65780 * 26334 * 8855 * 7315 * 560 * 364 * 351 * 56 * 35 = **15,734,979,985,584,545,279,193,600,000** programs of 32 exercises in each one.

This gives enough room to generate an almost infinite amount of programs and making sure that the users will not become adapted and stagnant by only doing very *similar* exercise routines.

These default values (amounts) of exercises per muscle though, will be possible to be changed (configured) to a certain degree by the customers who want to put special emphasis on certain muscle groups that are lagging behind and have not been trained in correct proportion with the rest, or muscle groups which need extra development due to necessities of a specific sport.

3.4.2 Personalisation Options

In order to personalise our solution and collect some initial customer requirements, we will need to provide a list of questions which when explicitly answered will give us the specific customer attributes or properties that will be used to filter our possible recommendations into a smaller set, that satisfies the customer requirements (see section 3.3).

The questions that will be asked will constitute the customer variable Vc (see section 3.3.1) in our knowledge-base and the answers will constitute the customer constraint Cc.

According our Scenarios (see section 3.1), the questions that will be asked are the following:

1) **Do you want to receive recommendation plans intended only for men, only for women or unisex plans?**

The rationale here is that there are certain exercises that are applicable especially to females (eg. Exercises that target the hips and legs areas which are usually weak areas for females) and there are other exercises that are more applicable to men (eg. Exercises that strengthen the neck and upper-body areas). Depending on the outcome of this question, certain exercises that are not applicable for the user can be removed from the constraint satisfaction solution of our recommender system, by adding a filter constraint Cf (see section 3.3.1).

2) **Which types of equipment are unavailable to you for your training?**

Obviously it would be impossible to do training without the necessary equipment, so exercises that require equipment that are not available to the customers, will be removed from the constraint satisfaction solution also.

As we are dealing with a weight training solution, we will assume that the users have the basic equipment of barbells and dumbbells, otherwise we would not be able to recommend any workouts at all. But there are certain gyms that do not provide all the types of machines for training, (eg. cable machines) and the user will be able to choose to exclude exercises that require these types of equipment that may be unavailable.

3) Which of the listed exercises you would not be able to perform (i.e. due to injury, inability, dislike, etc.)?

The exercises that cannot be performed for any other possible reasons, will be also removed from the solution set, by use of the same filtering method.

4) What type of exercise program do you need based on your level of advancement in training (beginner, intermediate, advanced, etc.)?

Here we are trying to find out what is the level of experience or advancement of the user into weight training, so that we will start our recommendations with types of programs that fit that level, and so guide him gradually into higher levels of development.

This question will adjust the output of the recommended program to fit a different schedule format, as described in sections 2.2.1 (types of routines) and 2.2.2 (Principle #7). It will split the result into consecutive blocks (workout sessions) to be performed in the recommended sequence while fitting the level of advancement of the user.

5) Is there some muscle group that you would like to target more or less than the usual (the default) and how much?

Here we will present our default values of the number of exercises per muscle group as described in the previous section (see 3.4.1) and we will give a possibility to the Users to adjust these values in case they have special needs to stress a muscle group in different proportion than the others.

As a final note, it is important to mention that the personalisation options described above are an *Optional* step and a user will not have to answer any of these questions or create any profile in order to get recommendations for exercises.

3.5 Automated Exercise Sequence Generation

In order to set up an automated exercise sequence generation, we need to clarify the rules and principles based on which this sequence will be created. For example, on music playlists one can generate a sequence based on songs of the same genre, rhythm, artist, etc. In our case, because we are dealing with biological factors, we will need to rely on the main training principles of the weight-training field, as we have described them in section 2.2.2 and which have been proven through years of practice and related research.

Some of the principles are related with sequences of exercises and the main ones that we will use are Principles #3, #5, #6. Per these, we need to group exercises for each muscle group together, starting with the bigger muscle groups and we must make sure that especially the bigger muscle groups do not get trained many times in the same week, so they can have time to rest from training and grow [2] [4].

So when we are going to train the next muscle group in an exercise sequence, we will need to choose the muscle group that is the most fresh and which has been trained *the least* so far! If we would choose muscle groups which already had a lot of training recently, we would end up with tired, painful muscles and no actual progress, as we

already described. So a good solution would recommend a sequence of muscle groups to be trained, which had the least amount of training so far.

To accomplish this, one has to know and keep track of how much each muscle group gets *Trained* and this could sound easy but it is complicated by the fact that there are many “compound” exercises (as we explained in section 2.2.1) which train many different muscle groups at the same time! So the choice of exercises for one muscle group has a compound effect on how much another muscle-group gets trained!

MUSCLE GROUPS/ Exercises	CHEST MUSCLE GROUP	TRICEPS MUSCLE GROUP	BACK MUSCLE GROUP	BICEPS MUSCLE GROUP	FOREARM MUSCLE GROUP	SHOULDER MUSCLE GROUP	ABS MUSCLE GROUP	LEGS MUSCLE GROUP	CALF MUSCLE GROUP
Bench Press	2	1	0	0	0	1	0	0	0
Chest Bentover Dips	2	1	0	0	0	0	0	0	0
Pullovers	2	0	1	0	0	0	0	0	0
Cable Flyes	2	0	0	0	0	0	0	0	0
Dumbbell Flyes	2	0	0	0	0	0	0	0	0
Chest Press	2	1	0	0	0	1	0	0	0
Pec Deck Flyes	2	0	0	0	0	0	0	0	0
Punch Press	2	0	0	0	0	1	0	0	0
Upright Dips	1	2	0	0	0	0	0	0	0
Kickbacks	0	2	0	0	0	0	0	0	0
Pushdowns	0	2	0	0	0	0	0	0	0
Close-Grip Bench Press	1	2	0	0	0	0	0	0	0
Triceps Extensions	0	2	0	0	0	0	0	0	0
Bentover Triceps Extensions	0	2	0	0	0	0	0	0	0
Bentover Row	0	0	2	1	0	0	0	0	0
Chinups/ Pulldowns	0	0	2	1	0	0	0	0	0

Table 6, Exercise training effect on muscle groups excerpt (see Appendix B)

To know how much each exercise affects each muscle group, we will need to extract this information from a thorough study of the domain [1], [2], [4] and we will then represent it in a new Table made by ourselves, which can be seen partly in Table 6 (and fully in 4-pages in Appendix B). This table lists all the main exercises by muscle group (without their variations - which have the same values) and it contains a rating or value of how much each exercise affects or trains the same and other muscle groups.

Those muscle groups that do not get trained by the exercise will get rated with a value of 0, those that get trained the most by the exercise (the Primary Muscle Groups) will get rated with a value of 2 and those that get trained a little, will get rated with a value of 1. These values, although they are an approximation, are still based on proven bio-mechanical factors [4].

As an example: an exercise that is called “Deadlift” primarily trains the lower Back muscles, but because it is a “compound” exercise, it trains also the upper Leg muscles. One would perform such an exercise normally together with other similar exercises that train the Back muscles (Primary Muscle Group), as per Principle #5 (see section 2.2.2).

When the time comes to train the Leg muscles, perhaps in the same session, the Legs have already been trained a little by the “Deadlift” exercise, so in order to avoid *overtraining* the Legs per principle #3, the Leg exercises have to be moved later in the sequence of training sessions, so that there will be an adequate rest interval between Leg training sessions and another more fresh muscle-group will be chosen instead.

The techniques and algorithms that we will use to accomplish that (as per section 3.3.3) are the following:

First, we will be using **Constraint satisfaction** methods to determine our sequences by implementing the following constraints:

1) *Unary Constraint:*

- Each unit exercise used in recommendations must come from the list of personalised exercises that has been filtered per the user requirements (if the user did any personalisation). This means that the results gotten from personalisation (see sections 3.3.1 and 3.4.2) will be input into the exercise-sequence recommendation algorithms in our Meta-level *Hybrid* recommendation system (see Table 1).

2) *Binary Constraint:*

- Adjacent exercises in the same cluster must be of the same muscle group, per our Principle #5 (see Section 2.2.2) until the maximum limit is reached for the muscle group per the user requirements and per Principle #4 by default.

3) *Global Constraints:*

- Each new exercise program recommended must contain a lot of new exercises, per Principle #8, so the exercise list must be shuffled each time to ensure adequate degree of Novelty.

- The total training of the muscles for the whole program (and for each exercise day), must not exceed the limits set per user requirements or by default per Principle #4.

- The amount of training days it takes to complete the exercise program must depend on the advancement of the user, as per the user requirements and Principle #7.

Then, we will be using **Similarity Heuristic** methods to determine:

1) the “seeds” in our sequences (which in our case will be: the next muscle group to be trained),
2) the “cluster” of exercises that will go together with the “seed”, and also:
3) the “replacement” exercises that will replace a recommended exercise, when a user wants to do fine-adjustments to his program.

- The “seed” muscle will be the most FRESH muscle group (the one with the least training) as much as possible, in order to avoid overtraining (especially the big) muscle groups, as per Principle #3.

By keeping track of how much each muscle group gets trained by each exercise (based on the training values we have given as per Table 5) we can use our intelligent algorithm to find the muscle group that had the least training and make that our next “seed” muscle. This will be the most “distant” (in terms of amount of training) muscle group, from the previous one in our sequence.

- The cluster of exercises that will follow the “seed”, will be determined based on the simple similarity attribute of “belonging” to the same muscle group (cluster), as per Principle #5.

- The exercise which will replace an already recommended exercise in the sequence of a program, will be determined based on similarity of how much it affects (trains) all the other muscles, as given on Table 6. The reason is so that it does not cause unnecessary overtraining to another muscle-group in the sequence.

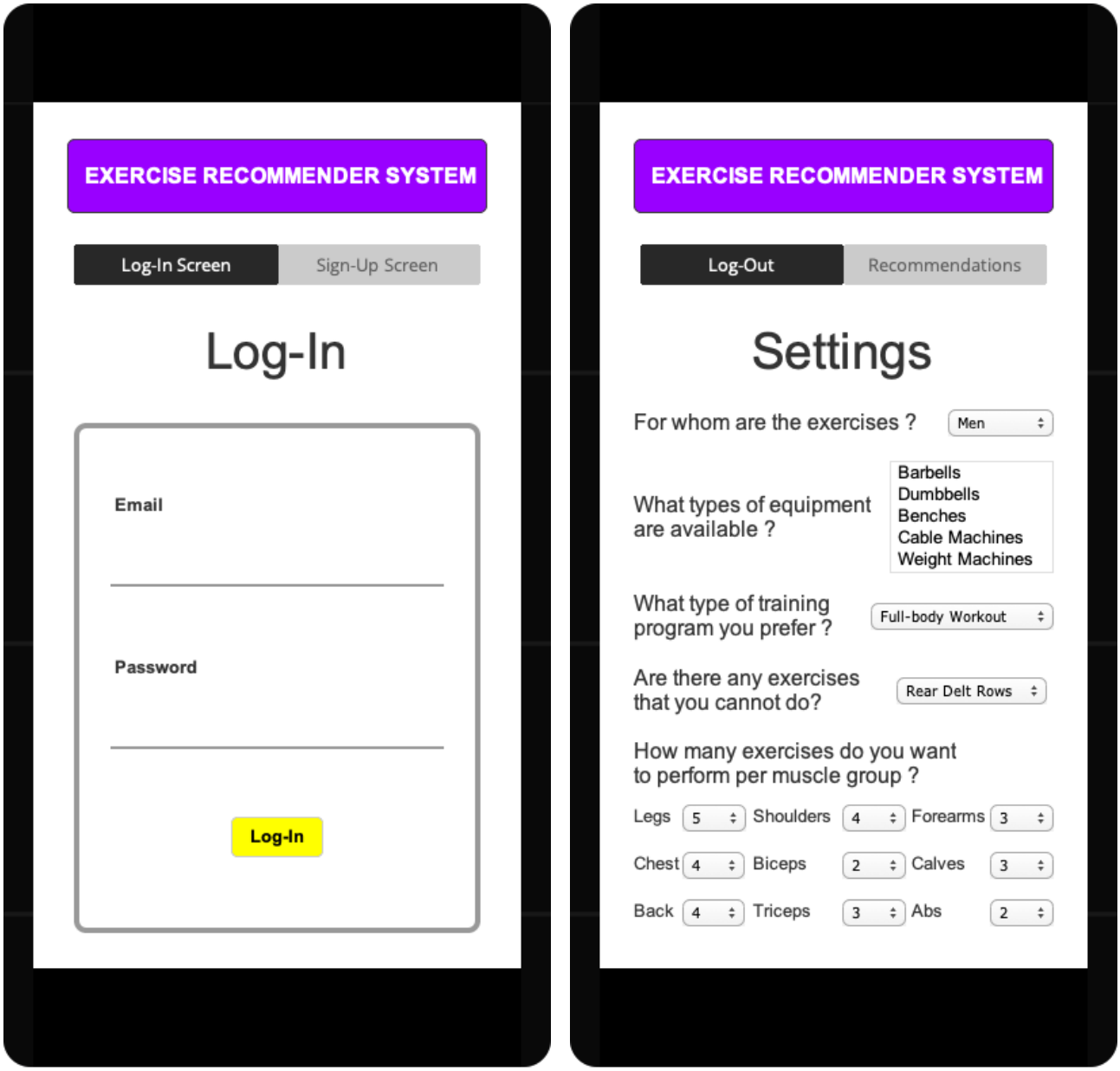


Fig. 16, Wireframes of the Log-In and Settings Page (made with JustInMind)

Using the exercise training vectors provided in Table 6 (9-dimensional vectors), we can calculate the similarity between exercises with the help of a similarity metric like the: Cosine-Similarity (see section 3.3.2). The exercises with the highest cosine-similarity to the one the user wants to replace, will be recommended in a sorted (ranked) list.

So even if the user gives the feedback that he wants to have changes in the recommended program, the algorithms can take this into account and provide improved recommendations for each individual user.

3.6 Hi-Fi Wireframes & Low-Fi Prototyping

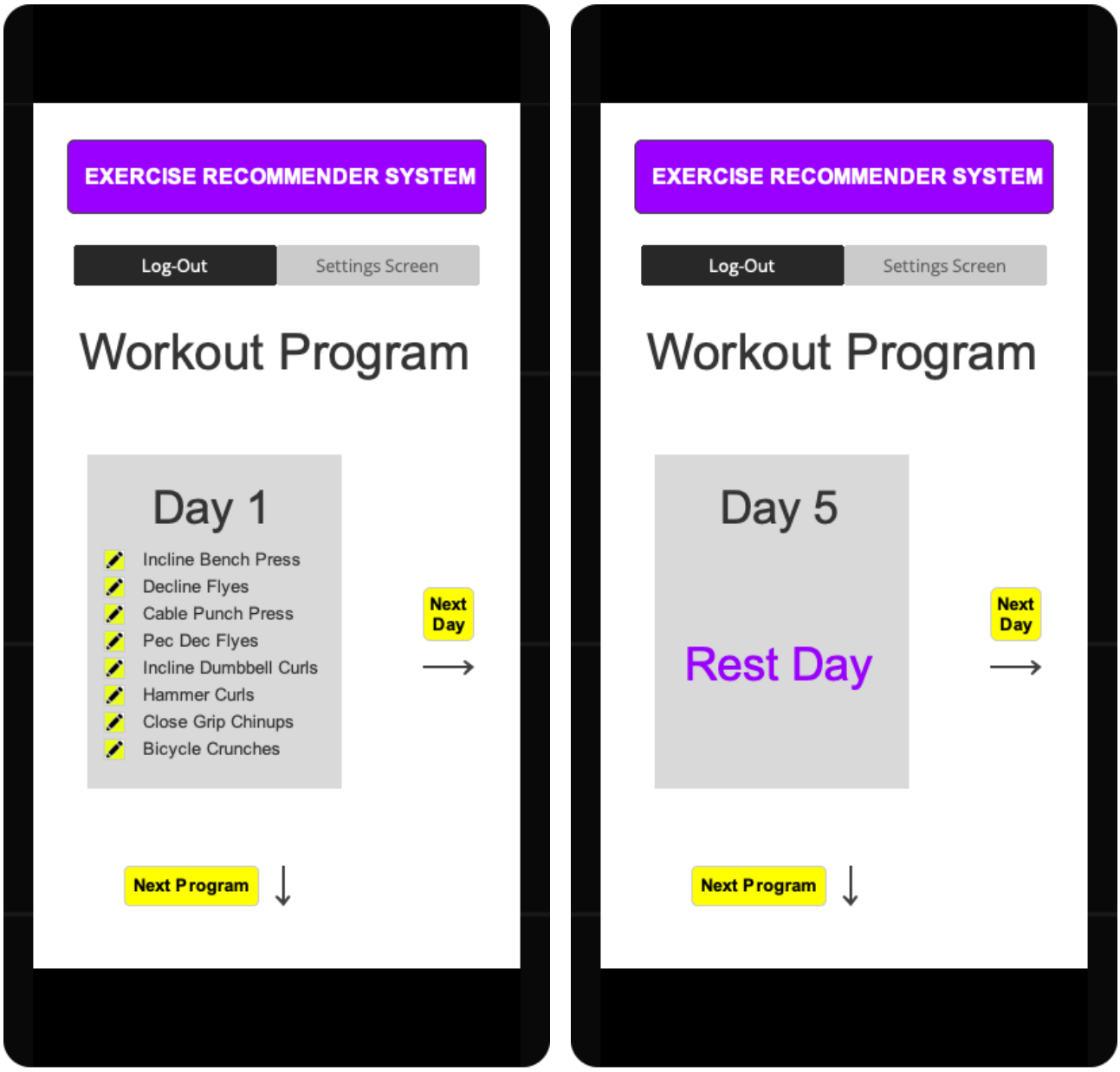


Fig. 17, Wireframes of the Recommendation Page for Days 1 and 5 (made with JustInMind)

As a next step, we wanted to create a first prototype and conduct some user studies to find out what the users think about our user interface, personalisation choices and the overall functionality of our solution and if they have something they want to contribute on these themselves. Then we can include the results of these tests into our Requirements Specification for our final solution.

We started off by creating some original wireframes which we can use in the tests. Wireframes can show the user interface, the functionality that is available and the behaviour of the proposed software, without including any actual code. Hi-fidelity wireframes (like the ones we used) can be computer generated, they show greater details and can also be interactive. [36] So we decided to go with high-fidelity wireframes while the resulting prototype had no code behind it and was therefore: “low-fidelity”.

The wireframes were made with a wireframe prototyping tool called “JustInMind” [35]. They concern the User Interface of a mobile solution because our solution is more likely to be used when the user is mobile at a gym.

Our first Hi-fi wireframes made with “JustInMind” show the Log-In and the Settings Page (Figure 16) where all our personalisation happens.

Specifically, the user here will have the opportunity to personalise the solution by adjusting: the types of equipment that are being used, the type of the program he will receive, any exercises that cannot be performed, the number of exercises per muscle group and whether the program is for men, women or unisex, exactly as described on our section 3.4.2 about Personalisation.

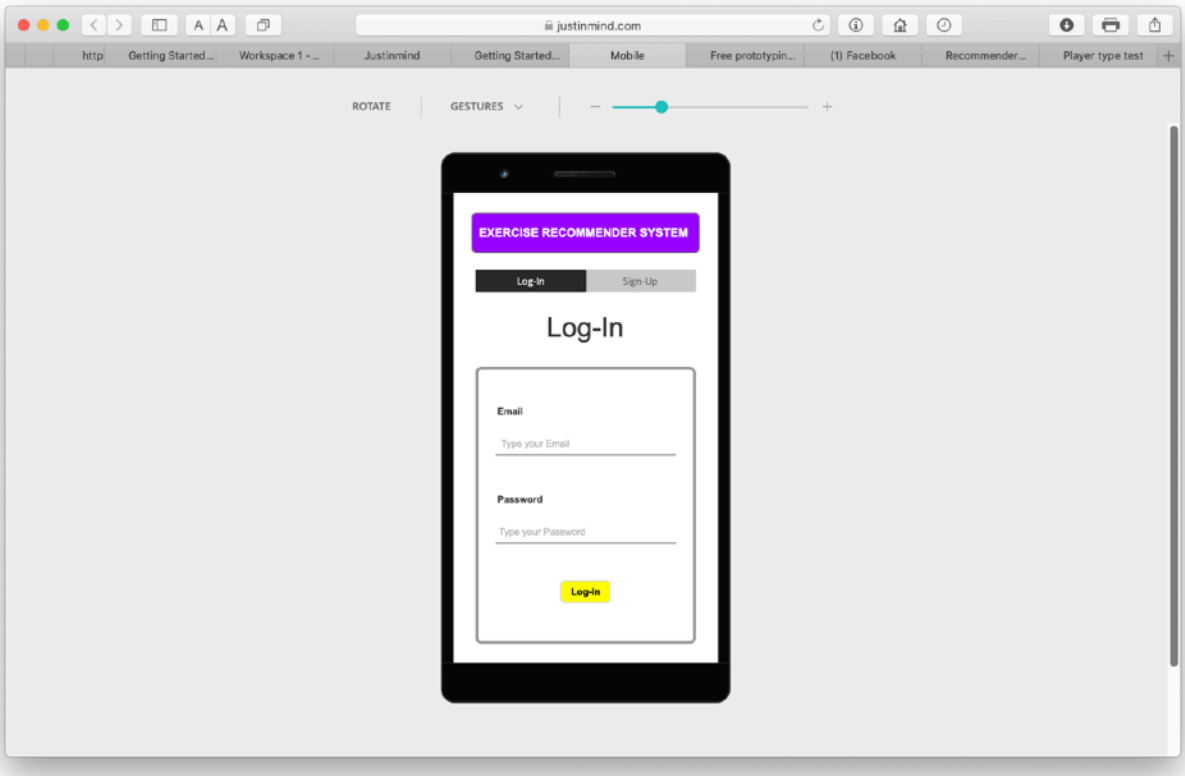


Figure 18, Interactive Simulation of our Low-Fi prototype published on the Web

Finally, there will be a Recommendation Page where the User will receive his recommended exercise program Day by Day (Figure 17) and he will be able to make some adjustments in case he does not like certain exercises recommended. The solution should then be able to recommend other equally good exercises.

When that original program is completed, the “Next Program” button can be used to provide a new program that is recommended as a sequence to the last one. Again adjustments can be possible for individual exercises, if this is wanted (Fig.17).

These wireframes were combined into an interactive simulated Low-Fi prototype with the help of our tool, which can be used for testing. The interactive prototype simulation can be published publicly on the web, and we are given a web-link which we can use to interact with our simulation online through a browser (see Fig.18).

Based on such a prototype, we can construct a Walk-Through or Think Aloud test [51] where we can record the opinions of users, as it will be described in the next section.

3.7 Setting-up User Studies

In order to find out if our proposed solution makes sense for the users, if the user interface is acceptable and whether any adjustments need to be made in regards to design and functionality, we want to perform a Think-Aloud Usability test. [55]

In Think-Aloud tests the users are using the software through the prototype while verbalising their thoughts. In order to do that, they have to be given specific tasks to perform and then let to do the talking. [34]

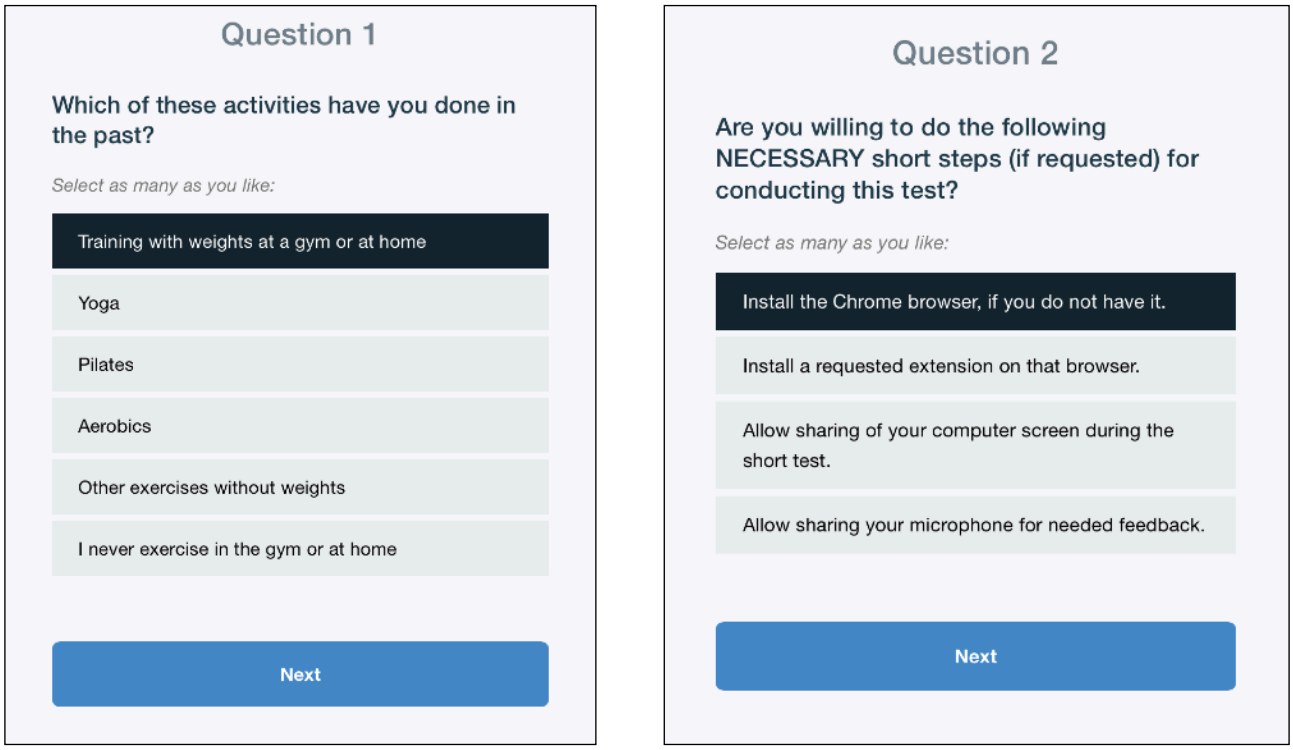


Figure 19, Screening questions (taken from screenshots)

Due to social distancing regulations imposed by the state in order to deal with a virus epidemic, these tests had to be performed remotely (online). We chose a platform to help us with that, called “Validately”, which offers possibilities for Remote user testing and automated reporting.

With “Validately” software, one can build and perform moderated or unmoderated user tests. In moderated tests a moderator observes the test participant and asks questions in real-time while in unmoderated tests the participants are left alone to complete the tests on their own time. [37]

We took the option of performing unmoderated studies in the beginning, due to the advantages in recruiting participants who can perform the test at their convenience, without putting a big stress on their schedules by arranging real-time meetings.

The unmoderated study is based on specific tasks to be performed while using a website or prototype through a web browser. At the same time there can be video and sound recording through screen and microphone sharing.

Through “Validately”, we gave specific instructions for each task, which need to be followed while “thinking aloud” on the microphone. After each task, additional questions were posed to collect more information verbally or in writing. These questions may require a rating or may be multiple-choice or just answered in simple words. When all the tasks are completed, we added some follow-up questions at the end.

Before a study is started, we made sure there are some “screener” questions to be asked to the participants to ensure that we are dealing with the correct target audience and that the participants are qualified for the test. (Fig.19)

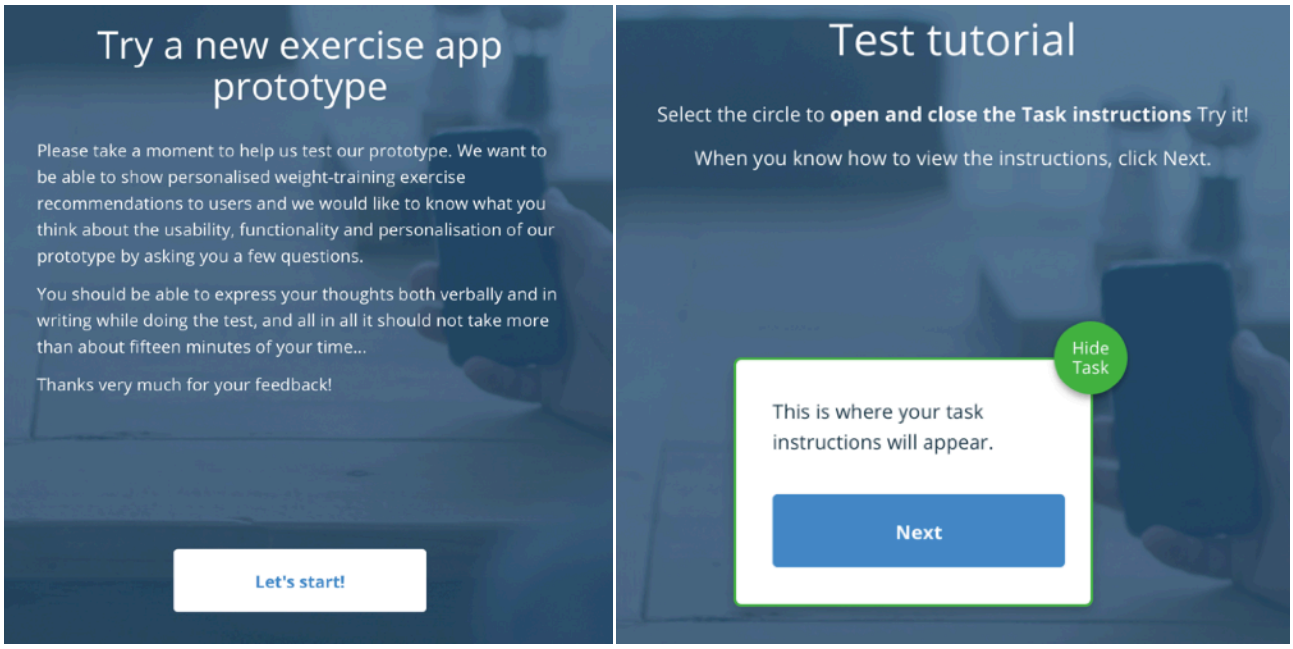


Figure 20, Some of the greeting and instruction screens (screenshots)

Then they see a greeting from us, to introduce the purpose of the study (Fig.20), and then there are some demographic questions to find out which parts of the population we cover

and some general behavioural information, before we start with the tasks to be performed.

Participant Screening

The target group for this test are people who have done training with weights in order to be able to understand most of the tasks, and of course those who are willing to have their computer screen and voice recorded, in order to collect the necessary information.

So the 1st screening question (see Figure 19) asks whether the participant has done training with weights in the past and also if he has done any other types of exercises like yoga etc, or if he never exercises. In case he never exercises or never done weight training, he will be automatically screened and cannot proceed with the rest of the test.

Same happens on the 2nd screening question, where the participant is asked for his consent to be recorded through microphone and screen capture and also for his willingness to install needed Chrome extensions. He will be screened from the study if he is not willing to give consent to any of these points.

Greeting and Instructions

After the screening, the participants are presented with some important information about the type and purpose of the test and they are also given a tutorial on how the test will be conducted and on how to set up their computer and browser for it. (see Fig. 20)

Pre-study Demographic Survey

The next part includes some demographic survey where we want to find exercise habits by asking how often they exercise and what training tools (if any) they use to help them along. (Fig.21)

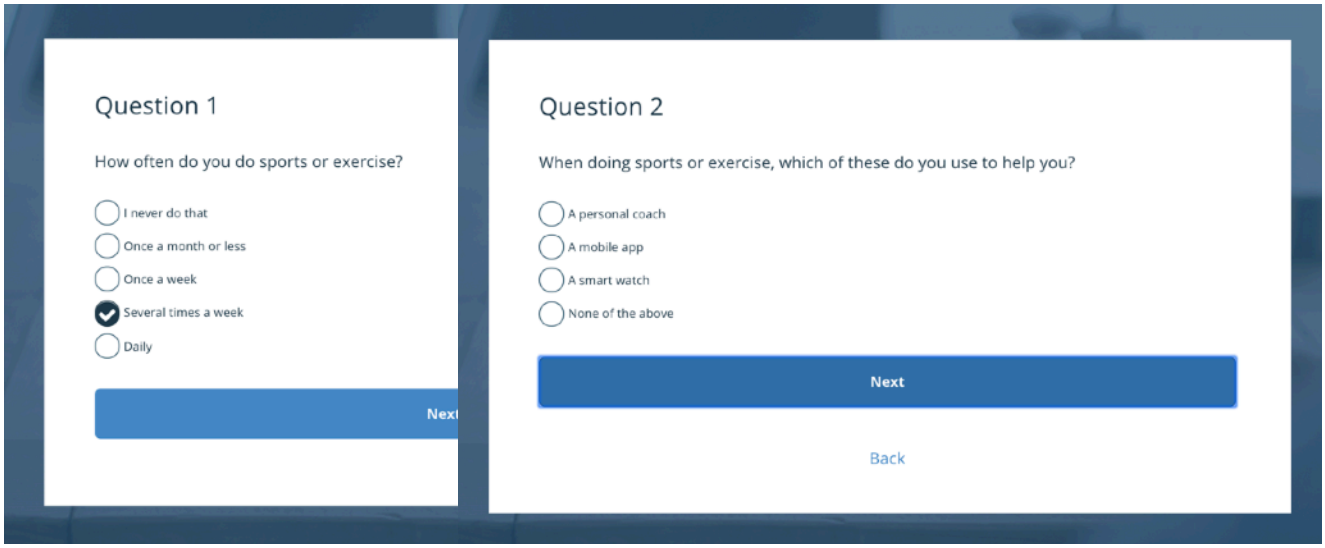


Figure 21, Surveying participant habits

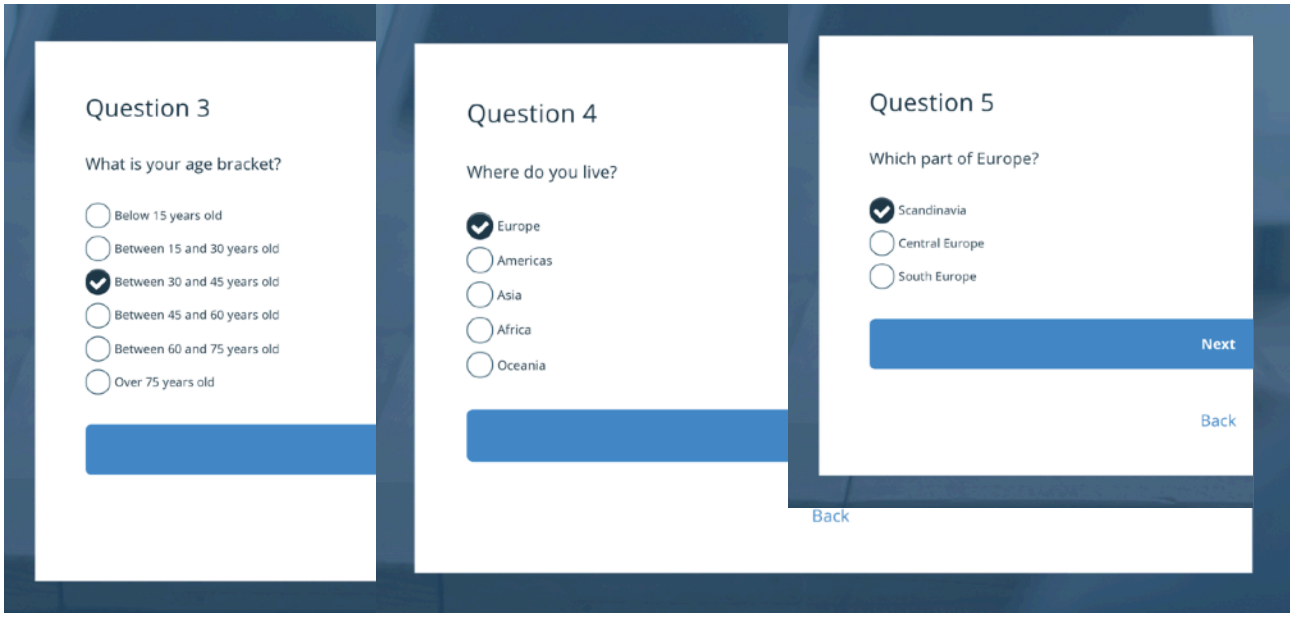


Figure 22, Demographic questions

Later on, we ask about their age bracket and place of residence. With these questions we get a better idea about who the participants are and making sure there are no obvious bias in our sample choice of participants. (Fig.22)

Tasks and Follow-up Questions

From that point on, we present the interactive prototype and give the participants specific tasks to perform while “Thinking Aloud” (see Figure 23). These Tasks include the use of all the “pages” of our prototype and its user interface and after each task we ask more detailed questions regarding the usability but also functionality of the prototype: Was the task easy to perform? Were there any difficulties? Are there any suggestions of how it could be improved? etc.

The participants are expected to use the prototype, configure their Profile Settings and receive exercise recommendations and in the end they are asked some final follow-up questions where they give their overall thoughts and ideas.

Then their test is uploaded on “Validately” servers where the recorded video can be evaluated by us in terms of whether the tasks were Passed or Failed, and notes can be kept about what the participants said or answered while “Thinking Aloud”. Finally, overall reports are generated with the full results from the test, for documentation purposes.

In the next section we will evaluate the results and then use them in our construction of a final Requirements Specification for our solution.

3.8 Results from initial User Study

We finally got 5 respondents to do our Unmoderated “Think Aloud” Usability Tests, which gave us quite some insight early in the design of our solution. [38]

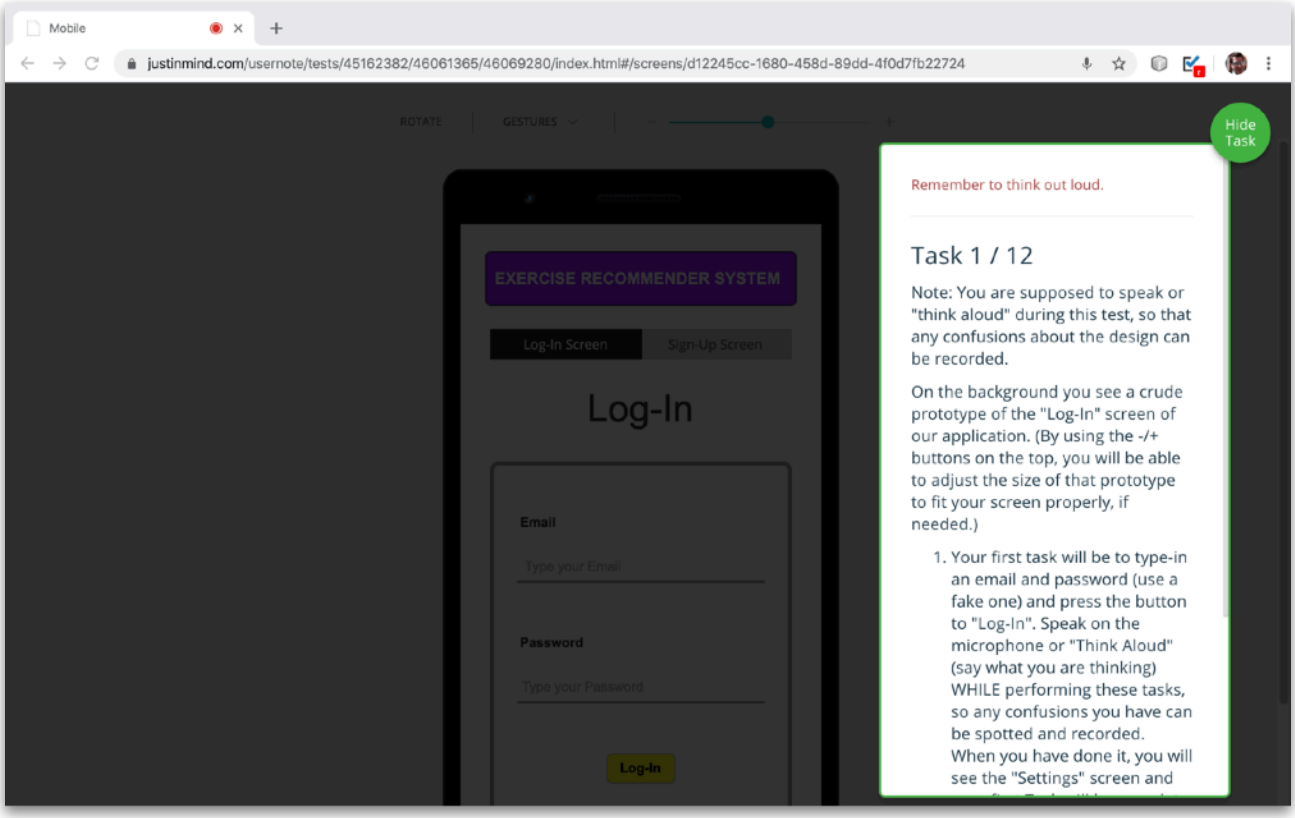


Figure 23, Description of a Task during the Recording of the “Think Aloud” Test (screenshot)

The general summary of the results can be seen in Appendix C and the full recordings can be found in the software attachments with this report. Here we will do our own analysis of the results and from this analysis we will collect user requirements for our requirements specification that follows in the next section.

Analysing Results

In our screener questions, we asked about what fitness activities our participants were involved in and we made sure that everyone had done some weight training. We also found out that some had done also Yoga, Pilates and other forms on exercises without weights.

Having qualified and screened our target group, we asked some more “pre-study” questions where we found that our participants exercise from once a month to several times per week and often use a mobile app (or less often a personal coach) to help them with their training.

Demographically we found out that we were speaking to an audience of 15 to 60 years olds living mainly in Scandinavia, but also we had one from Central and one from South part of Europe. Due to the fact that we were conducting these tests remotely, we had the opportunity to get results online from different countries with potentially different habits and behaviours. Also three of our participants were males and two were females, which means that we covered both sexes without a bias.

Then we started giving them tasks in writing which they would have to perform online while being recorded on video (see Fig.23 and Appendix C for full results). From these tasks we were able to collect both Quantitative and Qualitative data. The Quantitative data were the total time that the task took to complete and also whether the task was performed correctly and “passed” or not.

The Qualitative data were the answers that we received from the follow-up questions after each task. The 12 tasks given were the following:

- **1) Your first task will be to type-in an email and password (use a fake one) and press the button to "Log-In".**

The first task as seen above was related to Log-In and was passed by all participants. They also found it very easy to perform according to their answers on the follow-up question. The task took though more time to complete on average than the rest of the tasks (1 minute and 43 seconds) and this was because the users were still fiddling with the “Validately” interface and were not sure how to mark the task as “Complete” (per what appears from their Video recordings).

- **2) Your task is to set that the exercises that will be recommended are intended for BOTH men and women (Unisex).**

The second task which was related to personalising a profile variable, was completed quite fast by everyone, but 2 participants failed it because the original wording of the task was unclear and they understood something else than what was requested. The wording was subsequently corrected for the other users.

- **3) Your next Task is to set what equipment are available for weight training. You are supposed to mark ONLY these equipment as available: "Barbells", "Dumbbells", "Benches" and "Cable Machines".**

The third task was also completed very fast (14 seconds) and the participants reported no difficulties. Again two of them originally failed the task in some details because they misunderstood the wording of the task itself.

- **4) The next Task is to choose the type of Training Program. For the purpose of this Test, you are supposed to choose a "4-days split routine" program, which takes 4 days to complete.**

The fourth task again was performed fast (16 seconds on average) without any difficulties reported, except one participant mentioning that he did not understand what the term “split-routine” meant.

- **5) The next Task is to Set if there are any exercises that you cannot do for some reason. For this Test you need to choose only the exercise called: "Squats".**

The fifth task again was performed fast (17 seconds) and only two of the participants reported difficulty because they were required to choose from a list of items that was not alphabetised, and that caused some delay in finding the item.

- **6) The last Task on the Settings screen is to choose how many exercises should be performed per muscle group. There are 9 muscle groups listed. You are supposed to change only 2 default numbers: The exercises for the "Back" muscles need to change from 4 to 5 and the exercises for the "Biceps" muscles need to change from 2 to 3.**

The sixth task was performed also very fast (14 seconds) and without reported difficulties. One user failed because he did not understand the term “Biceps” mentioned in the task.

- **7) Now you need to Press the Recommendations Menu-button in order to go to the "Recommendations" screen. Here you will see the first day of your Recommended Exercise Workout Program. It will include a simple list of 8 names of exercises, grouped together based on their muscle groups.**

The seventh task was the easiest. Everybody understood it and passed it and they completed the task quite fast (21 seconds) without any apparent confusions.

- **8) Now we suppose that you performed the exercises of the first day and the second day of your program. Your Task is to see the Recommendations of exercises for the third day (3rd) of your program.**

The eighth task also was passed by everyone and was completed very fast on average (11 seconds). No difficulties reported.

- **9) Now we suppose that you have finished all the 5 days of your exercise program and you are going to repeat the same program again from Day 1. Your Task is to continue getting recommendations for the next days of the program until you arrive again at the same Recommendation for Day 1.**

The ninth task was also found “easy” by everyone and was completed fast but one failed it due to the wording of the task being a bit unclear.

- **10) Now we suppose that you have performed this exercise program for several weeks and you need to get the next one. Your Task is to request and get a recommendation for the next Program to do.**

The tenth task was also found easy by most of them, but again one participant failed it. The task itself was not well understood.

- **11) Ok. Now let's say that you have arrived to the second day of your new program and you want to see the recommendation. When you do that you realise that you are not satisfied with some exercise of the 2nd day and you want to change it. Your double Task is: a) to see the 2nd Day of Recommendations and then b) request editing of the recommendation of one of the 8 exercises, and receive a similar recommendation (not the same) for that 2nd day of exercises.**

Task eleven was the most difficult task of all, taking longer time to complete (1 minute 58 seconds), but still the majority of the participants passed it. There were some difficulties reported about being unsure on how to change recommendations for a given day.

The editing button with the “pencil” on it did not give any additional information about its function. There were also some suggestions that, when the editing button is pressed, one should see a pop-up screen with options of alternatives, instead of getting an immediate recommendation. And also a suggestion that there should be a cancel button if one wanted to delete an exercise.

- **12) Finally your last Task is to Log-Out for the day and be directed back to the Log-In screen.**

The last task number twelve was passed by everyone, in a very short time (12 seconds).

On the follow-up questions about the whole test, on a scale from 1-10 the majority said that they found it very easy (rated with 9) to use the prototype in general (see Figure 24).

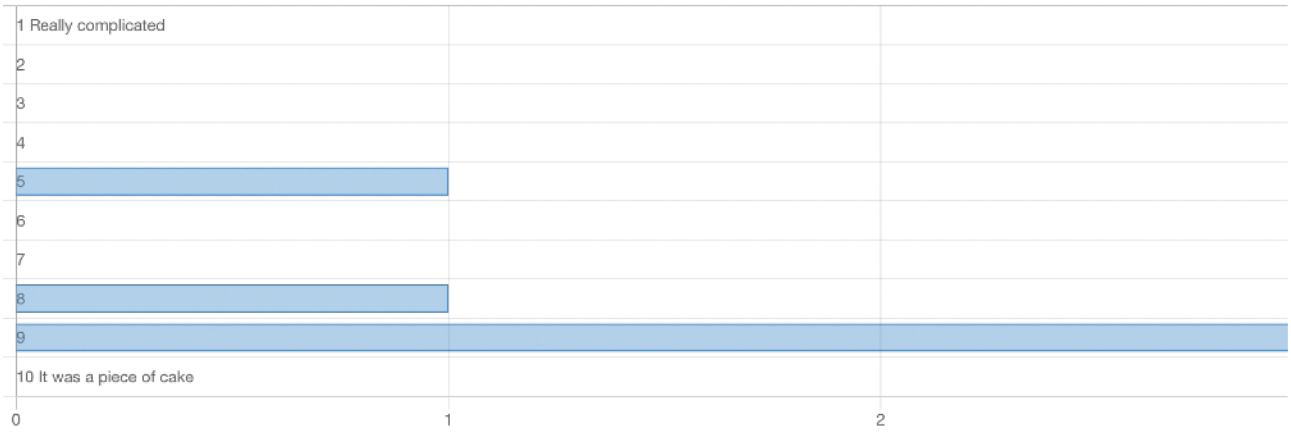


Figure 24, Prototype overall Ease-of-Use rating (see Appendix C)

On comments about the the personalisation options: these were found adequate. Only mention was that the drop-down menus could contain a “blank” option when the user needed to choose “nothing”.

On comments about the presentation of recommendations: again the point about the pop-up menu for editing was mentioned and that the “graphics” design could become more attractive in later high-fidelity versions of the prototype.

Finally, on the last question about any other suggestions, it was mentioned that it would help to have also some visuals about how to do exercises.

Conclusions

After this collection and analysis of the test results, we made some conclusions and tried to extract some requirements for our later work, as follows:

Some difficulties during the tests were observed due to some participants not understanding certain terms used that might require more explanation. This can be

handled by providing additional helpful information about the various options in the form of optional “notifications” for those who need them.

When long lists of items are presented, these need to be alphabetised for easy search and a blank option needs to be included when needed.

The recommended exercises need to include an edit and cancel button. When editing is requested, a pop-up is to be presented where users can choose from a list of possible alternatives, instead of being presented immediately with a recommendation.

Finally, we realised that some visuals of the exercises would need to be added to help the users better understand their choices of exercises.

3.9 Requirements Specification

In this section, we specify our requirements based on all the analysis we have done so far.

Functional Requirements				
ID	**Description**	**Reason**	**Link**	**Priority**
1	The user should be able to Register, LogIn and LogOut	It helps in creating personal user profiles	Section 3.1, Section 3.2	SHOULD
2	The user must be able to choose whether the recommendations will be for men, women or unisex.	It creates a more personal profile for the user	Section 3.1, Section 3.2	MUST
3	The user must be able to adjust the available equipment	It helps personalise the recommendation to the user’s environment	Section 3.1, Section 3.2	MUST
4	The user must be able to choose the type of the exercise program	It influences the final recommendation and how it is presented	Section 3.1, Section 3.2	MUST
5	The user must be able to exclude exercises that he cannot do	It helps personalise the recommendation to the user’s abilities	Section 3.1, Section 3.2	MUST
6	The user must be able to adjust the amount of exercises per muscle group	It helps personalise the recommendation to the user’s special needs	Section 3.1, Section 3.2	MUST
7	The recommendations must be personalised to the user based on his profile (phase 1)	It makes it easier for the user to attain his own, current goals	Section 3.1, Section 3.2	MUST

We will start with the Functional Requirements which will describe WHAT our solution must be able to do. Then Non-Functional Requirements will describe HOW this must done and any limitations and constraints on how it is done. [39]

Subsequently the requirements will be prioritised in what MUST be done to meet the customer/business needs, what SHOULD be done but which is not 100% necessary, what COULD be done if it does not cause many problems, and what WOULD be done if there was enough time, but it will rather be left for a future iteration. This is called the MoSCoW prioritisation model. [40] (see Table 7 and 8 for the requirements)

Functional Requirements				
ID	**Description**	**Reason**	**Link**	**Priority**
8	The recommendations must be given in a continuous sequence from day to day and from one muscle and exercise to the next (phase 2)	It takes the guess-work away from the user	Section 3.1, Section 3.2	MUST
9	The user must be able to request small changes on the recommendations, once they are presented. (phase 3)	It makes the recommendations more acceptable to the users	Section 3.1, Section 3.2	MUST
10	The user must be presented with a list of best choices when changes are requested	It makes the recommendations more acceptable to the users	Section 3.7	MUST
11	The user must be able to delete exercises from his recommended program	It makes the recommendations more acceptable to the users	Section 3.7	MUST
12	The user must be able to request a completely new and novel program when the last one is finished	It helps in ensuring stable progress and removes any guess-work	Section 2.1.5, Section 3.5	MUST
13	The user could have the possibility to save his exercise programs for later use	It helps the user to not lose track of his progress	Section 3.1, Section 3.2	COULD
14	The user could have the possibility to load past programs that he has saved	It helps the user to not lose track of his progress	Section 3.1, Section 3.2	COULD

Table 7, Functional Requirements Specification

Non-Functional Requirements				
ID	**Description**	**Reason**	**Link**	**Priority**
15	The system should provide additional information as a Help for users who need it	It helps in the Usability and Learnability of the solution	Section 3.7	SHOULD
16	The system should present long lists of items sorted for easy search	It helps in the Usability and Efficiency of the solution	Section 3.7	SHOULD
17	The system should include visuals of exercises	It improves Usability and Satisfaction	Section 3.7	SHOULD
18	The system must be designed and presented as a mobile solution	It fits better to the environment of the fitness user, which is usually mobile	Section 3.6	MUST
19	The system design could include a Database where the exercise data and user data can persist	It offers protection from loss of personalised information	Section 3.4, Section 3.5	COULD
20	The Recommendation techniques must be Constrained-based and Sequential	It offers personalisation and better automated recommendations	Section 3.3	MUST
21	The Recommendation algorithms must include Constraint-satisfaction and Similarity Heuristics	It avoids errors based on popularity and the lack of training data	Section 3.3	MUST
22	The Recommendation system should be Transparent on how it works	It helps to create trust with the users and encourage its use	Section 2.1.5	SHOULD

Table 8, Non-Functional Requirements Specification

In this specification, we try to show the reason why the requirements were chosen and also refer to the section of the report where this requirement was mentioned or previously analysed. Finally, we marked also the priority of the requirements based on the MoSCoW model as previously mentioned.

After describing our requirements, with the data we collected and analysed from our documentary research and the preliminary user studies, we are ready to start designing our final solution.

4. Software Design

In this Chapter we will present our design for the solution following the Requirement specification that we just outlined. We will present the System Architecture, the Database Model, Class diagrams and Algorithm flow charts and pseudocode.

4.1 System Architecture

In the first diagram we will be describing the architectural design for the whole system and how the components communicate with each other. [56]

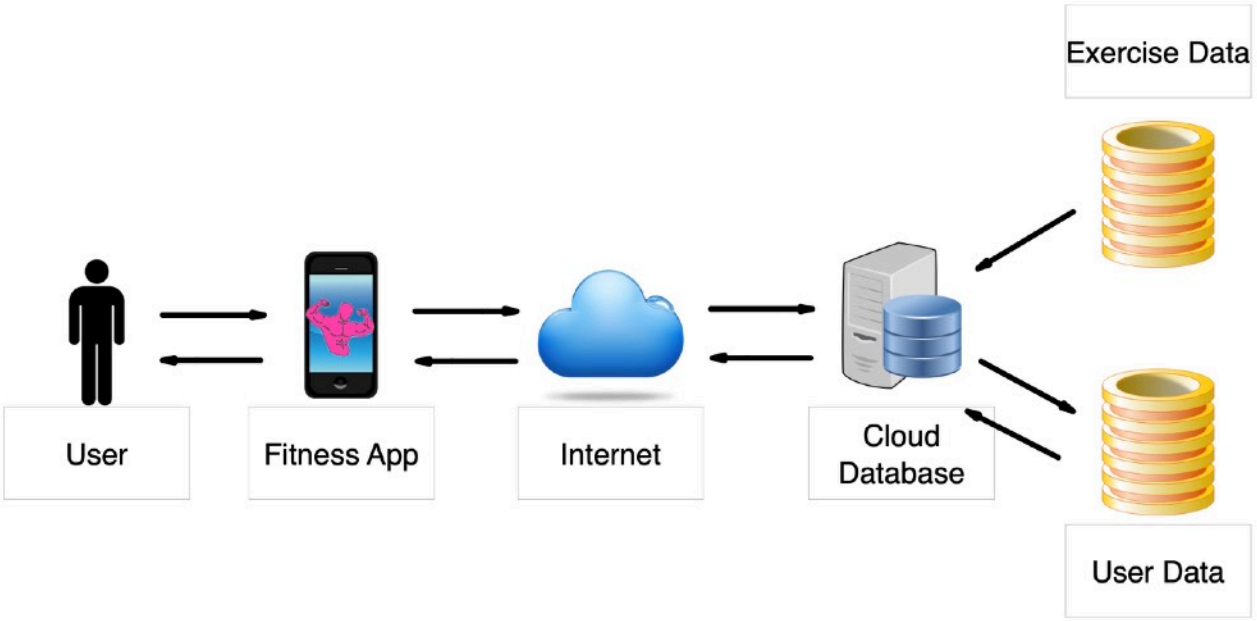


Figure 25, Architectural Design (made with VisualDesigner)

As can be seen on Figure 25, the solution will include the development of a mobile application that can be installed on a mobile device (per REQ#18) and which will handle all the interaction with the users.

The app will send user data through the internet to a remote Cloud Database (per REQ#19) where they will persist and also data from the weight training domain about exercises will be input into the database and used by the app upon request.

In this way we can keep many User Profiles and be able to make periodic updates to our solution by adding more exercise data, etc.

4.2 Database Design

Next action will be to design the data that will be held in the Database. What follows is a diagram of the database model as seen in Figure 26.

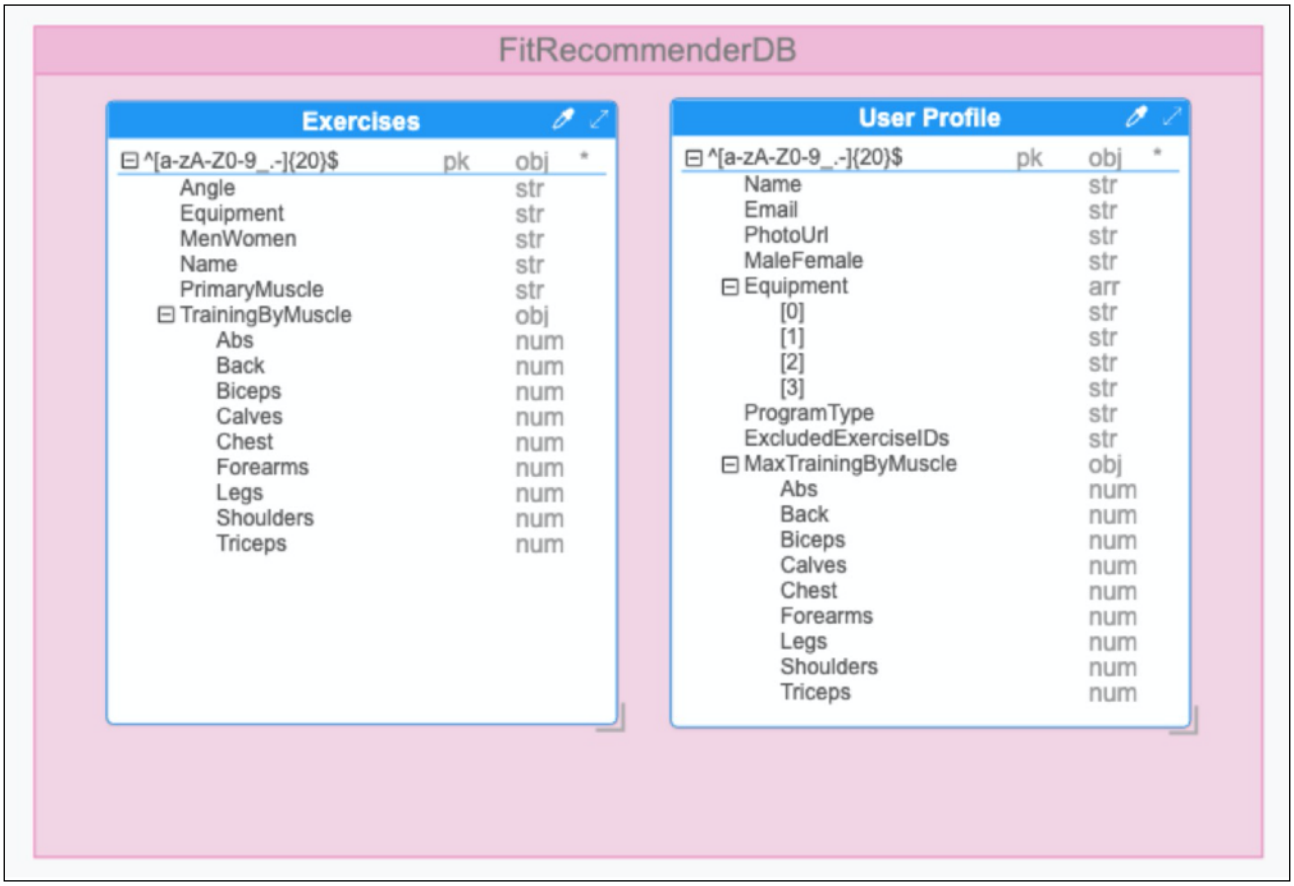


Figure 26, Database Model (made with Hackolade)

Here you see two collections (tables) of data: one for the Exercises and one for the User Profile. The first collection is supposed to contain documents (records) of all the 165 exercises that will make up our knowledge base for this app. Each exercise document will contain data strings about the name of the exercise, the equipment used, if it is intended for men or women or both, the bio-mechanical angle of motion, the primary muscle being trained and the amount of training delivered to each of the 9 muscle-groups. This is data that will be taken directly from our Tables 4 and 6 (see Appendices A and B).

The next collection will contain documents of all the User Profiles. Each profile will contain the name of the user, the email, a photoUrl if it exists, and user requirements about the exercises and the program to be recommended. These include: whether the program is for males etc. what equipment are missing if any, what is the program type based on the advancement of the user, any ids of exercises that will be excluded, and what is the maximum amount of exercises wanted per muscle group. This information will be enough to personalise the recommended programs to the individual user.

4.3 Class Diagram

Next we will describe the main Classes that will be needed for our app to hold the necessary information that will give us the final Recommended Program - and later we will describe the algorithm that can be used to handle that information and give the recommendations. The emphasis here is on the Classes' attributes and operations.

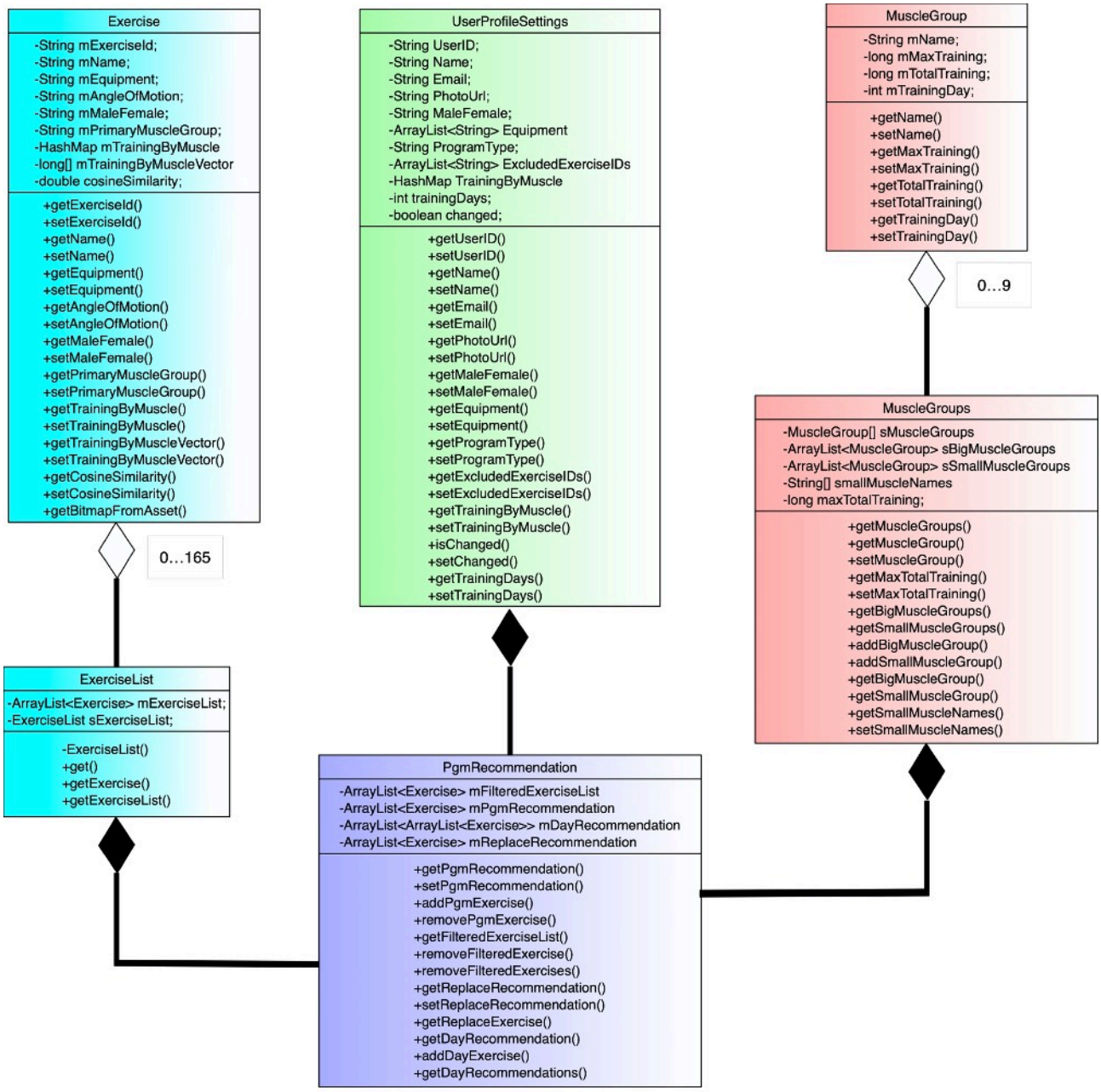


Figure 27, Class Diagram (made with VisualDesigner)

On Figure 27 we can see that we will have an "Exercise" Class which will hold all the necessary Exercise attributes that will specify the qualities of each exercise. These include the name and id of the exercise, the equipment with which it is performed, the variation based on biomechanics (angle of motion), if it is intended only for men or women, the primary muscle group that it trains and the amount of training that it performs

on all muscle groups per Table 5, and the cosine similarity with other exercises. These (165 exercises) will be aggregated into an “Exercise List” Class which can then be filtered according to our needs.

At the same time, we will need to have a “Muscle_Group” Class that will hold the attributes of the different (9) muscle groups including the name, what is the maximum allowed amount of training to be accomplished by the exercises for this muscle group (MaxTraining), as well as what is the total amount of training currently accomplished by exercises for this muscle group (TotalTraining) and to which training day it belongs (TrainingDay). These will also be aggregated into a “Muscle Groups” Class which will hold all the muscles groups, their maximum total training and will also separate them into Big and Small Muscle Groups.

Then we will have also a “User Profile Settings” class which will hold all the profile settings given by the users, including User id, Name, Email and Photo Url (if they exist), Then also the choices about equipments, program type, male/female/unisex, any excluded exercises, the total exercises per muscle etc.

Together with the “Exercise List” Class and the “Muscle Groups” class, the data will be input into the “Pgm Recommendation” class which will hold the exercises of the Program Recommendation, as well as: 1) the exercises that are Filtered after the User personalised Settings, 2) the exercises that belong to a specific training Day, and 3) the exercises that will be recommended as a Replacement to those that the user may want to change at some point.

After describing the data structure of our solution, we can go into the intelligence that will be used by describing the algorithms for filtering and recommendations.

4.4 Algorithm Flow Charts & Pseudocode

The algorithm for the Recommender System according to our Requirements should include 3 main phases: 1) Personalisation through Constraint Satisfaction per User Settings, 2) Automated creation of a Recommended Exercise Program whenever requested, and 3) Fine-grained Adjustments to the Recommended Programs based on User Feedback/Request.

These Phases are independent from each other and can be repeated at will, but they influence each other and the final results obtained. We will now describe the algorithm by using Pseudocode or Flow Charts for illustration, for each of these phases:

4.2.1 Personalisation through Constraint Satisfaction

After our full Exercise List and the User Profile Settings have been initialised by reading the data from the Database, the full list of exercises will be filtered based on the user requirements to remove exercises that: are not intended for the user, that require equipment that are not available, or that the user cannot perform for some other reason, as described in section 3.3.3 and with the basic algorithm described in section 3.3.1.

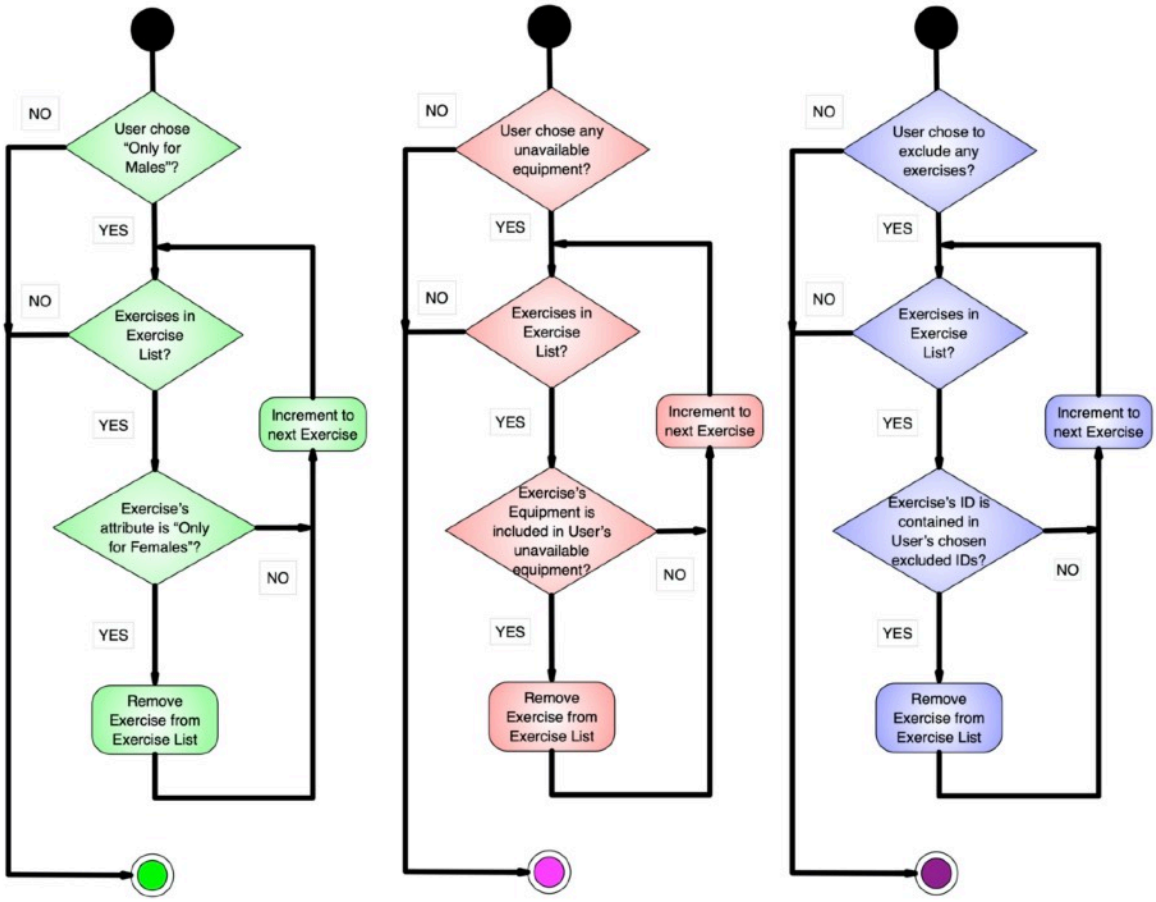


Figure 28, Algorithm Flow Charts for Exercise Program Personalisation (made with VisualDesigner)

As seen in Figure 28, if the User has chosen exercises “only for males” then as long as there are exercises in the Exercise List, the exercise’s attribute will be checked and if it says: “only for females” then it will be removed from the list and we will continue with the next exercise until there are no more exercises to check. The opposite would happen if the user chose “only for females”. Then the exercises with the attribute saying: “only for men” would be removed etc.

Then, in case the user has indicated that certain equipment are unavailable to him, then for each exercise in the exercise list we check if the exercise’s equipment is included in the user’s list of unavailable equipment, and if so, we again remove it from the list.

Finally, in the case that the user has chosen certain exercises specifically to be excluded, we check for each exercise on the list: if their ID matches the IDs of the exercises chosen by the user and if so we remove them.

This process completes our filtering phase, and next we will describe the algorithm for the Recommendation phase.

4.2.2 Automated creation of a Recommended Exercise Program

As a first step in the recommendation phase (Figure 29), we need to initialize the data about the Muscle Groups and also divide them in Big Muscle Groups and Small Muscle Groups because as we have seen in section 2.2.2 there are some differences in the training principles that are applied to each.

We will also initialize some variables like: the number of Training Days of the recommended program which depends on the type of program chosen by the User, and the Maximum Daily Training variable which in its turn depends on the Maximum Training for the whole Program as set in the Settings (default is 64), divided by the number of Days in the current Program (see Global Constraints in section 3.5).

Then we will shuffle the sequence of the exercises in the Filtered Exercise List that we obtained in phase 1, as per our Unary Constraint and first Global Constraint (see section 3.5). The randomization is done in order to ensure Novelty in exercise selection and also according to our principle #8.

```
INITIALIZE TrainingDays, MaxDailyTraining, BigMuscleGroups, SmallMuscleGroups, MuscleGroups;
SHUFFLE Exercise List;
FOR day := 1 to TrainingDays DO
BEGIN
        TotalDayExercises := 0; DayMuscleTraining := 0;
        WHILE DayMuscleTraining < MaxDailyTraining DO
        BEGIN
                SORT BigMuscleGroups by min TotalTraining;
                SeedMuscleGroup := BigMuscleGroups( 0 );
                IF DayMuscleTraining + BigMuscleGroups( 0 ).MaxTraining > MaxDailyTraining THEN
                BEGIN
                        SORT SmallMuscleGroups by min TotalTraining;
                        SeedMuscleGroup := SmallMuscleGroups( 0 );
                        IF DayMuscleTraining + SmallMuscleGroups( 0 ).MaxTraining > MaxDailyTraining THEN BREAK;
                END;
                TotalSeedMuscleExercises := 0;
                FOR EACH Exercise in FilteredExercise List DO
                BEGIN
                        IF Exercise.PrimaryMuscleGroup == SeedMuscleGroup THEN
                        BEGIN
                                TotalDayExercises++;
                                TotalSeedMuscleExercises++
                                DayMuscleTraining := 2* TotalDayExercises;
                                ADD Exercise to DayPgmRecommendation;
                                FOR EACH MuscleGroup in MuscleGroups DO
                                BEGIN
                                        MuscleGroup.TotalTraining :+= Exercise.TrainingByMuscleGroup( MuscleGroup );
                                END
                                SeedMuscleTraining := 2* TotalSeedMuscleExercises;
                                IF SeedMuscleTraining >= MuscleGroups.MaxTraining( SeedMuscleGroup ) THEN BREAK;
                        END;
                END;
        END;
END;
```

Figure 29, Algorithm Pseudocode for Recommendations

Next, for each Training Day we perform a series of actions in order to collect the right exercises that will be included in the program for THAT training day. When this happens for all the training days of the program, we will have a full Exercise Program Recommendation to present to the User.

So for each day we take the following actions:

We initialize a counter for the Total Day Exercises for that day and for the total Day Muscle Training for that day. (The total Day Muscle Training refers to the total training of the Primary Muscles for that day and equals 2 times the number of exercises of that day.)

As long as the Day Muscle Training is less than the Max Daily Training for that day, we need to continue finding exercises to recommend for that day. We do that as follows:

We start with our Big Muscle Groups per Principle #6, and we sort them by least Total Training. (This means that we take the Fresher muscle groups first). We assign as our Seed Muscle Group, the first Muscle Group in the sorted list (the Muscle Group with the least amount of training, as per our first Similarity Heuristic in section 3.5).

If our Max Daily Training global constraint is surpassed by the Day Muscle Training plus the Max Training for the Seed Muscle Group then we should not accept this Big Muscle as Seed Muscle Group but we should look for a Small Muscle Group. To do this:

We Sort also the Small Muscle Groups by least Total Training and we take the first one, we assign it as Seed Muscle Group and check again our Global Constraint. If it is again surpassed then we do not recommend any more Muscle Groups for that day and break the loop for the next day.

Having chosen an acceptable Seed Muscle Group, we initialize the Total Seed Muscle Exercises variable and we start finding a sequence of recommendable exercises for that Seed Muscle Group. We do this in this way:

We look in the list of Filtered exercises for an exercise which has the Seed Muscle Group as its PRIMARY muscle group, as per our second similarity heuristic in section 3.5. This is a recommendable exercise.

We add the exercise to the Day Pgm Recommendation for that day and increment the variables of Total Day Exercises, Total Seed Muscle Exercises and Day Muscle Training.

Then we increment also the Total Training of each Muscle Group by adding the Training that the Recommended Exercise does to each Muscle Group.

Finally we calculate the Seed Muscle Training as 2 times the Total Seed Muscle Exercises and if that exceeds the Max training set for that Muscle Group then we need to stop recommending exercises for that Seed Muscle as per the second Global Constraint and break that loop. Otherwise we will continue looking for exercises to train that same Seed Muscle Group.

When the loop is broken, we will check again (per the previous loop) if we have exceeded our Max Day Training, and if not, we will find a new Seed Muscle Group. When

that loop is complete, we will go to the next Day of Recommendations until we arrive to a complete Recommended Program.

4.2.3 Fine-grained Adjustments based on User Feedback (Request)

In this final phase (Figure 30), we have the possibility to make fine-grained adjustments based on user request, because the user may still not like some of the exercises recommended, in which case we need to recommend a ranked list of the best replacement exercises for the one the user is not satisfied with. To accomplish this adjustment, we do the following:

```
INITIALIZE ExerciseToBeReplaced, ReplaceRecommendationList;
FOR EACH Exercise in FilteredExerciseList DO
BEGIN
        IF Exercise.PrimaryMuscleGroup == ExerciseToBeReplaced.PrimaryMuscleGroup) THEN
                ADD Exercise to ReplaceRecommendationList;
END;
FOR EACH Exercise in ReplaceRecommendationList DO
BEGIN
        DotProduct := 0;
        NormA2 := 0;
        NormB2 := 0;
        FOR i:= 0 to 8 DO
        BEGIN
                DotProduct :+= Exercise.TrainingByMuscleVector(i) * ExerciseToBeReplaced.TrainingByMuscleVector(i);
                NormA2 :+= Math.pow(Exercise.TrainingByMuscleVector(i), 2);
                NormB2 :+= Math.pow(ExerciseToBeReplaced.TrainingByMuscleVector(i), 2);
        END;
        NormA := Math.sqrt(NormA2);
        NormB := Math.sqrt(NormB2);
        Exercise.CosineSimilarity := DotProduct / NormA * NormB;
END;
SORT ReplaceRecommendationList by max CosineSimilarity;
```

Figure 30, Algorithm Pseudocode for Recommendation Adjustments

First of all we initialize the Exercise To Be Replaced and the Replace Recommendation List where we will put our recommendations of replacement exercises.

To find our replacement exercises, for each exercise in the Filtered Exercise List we find if its Primary Muscle Group equals the Exercise's To Be Replaced Primary Muscle Group, and if so we add it to the Replace Recommendation List. If not, we do not recommend it, because we would never recommend an exercise that belongs to a different Muscle Group (there is too much dissimilarity!).

Then, for each exercise in the Replace Recommendation List we calculate the Dot Product of the Exercise's Training By Muscle Vector and the same Vector of the Exercise's To Be Replaced.

Then we also calculate the Norms of these 9-component Vectors, and finally we calculate the Cosine Similarity (as was described in the third Similarity Heuristic in section 3.5 and in section 3.3.2).

Finally we Sort the Replace Recommendation list based on the Cosine Similarity of the Exercises included.

This completes the description of our algorithms for making sequential recommendations based on training principles of the weight training domain. The next chapter is going to describe the implementation of the solution.

5. Implementation

In this chapter we will describe how we developed the software application and the whole solution and we will start with the Database.

5.1.1 Cloud Firestore Database & Firebase Authentication

The database that we chose is a cloud NoSQL database called Firestore which is part of the Google Cloud Platform. [47] NoSQL databases are very scalable and open-source which makes them easy to use and free of charge for a certain number of data.

With Cloud Firestore you can share your app data among many different users and there are libraries for mobile development as well as for the Web. Another good point is that it can work in off-line mode also, which means that it does not keep users waiting for connection when the connection is temporarily not possible.

Additionally it includes Firebase Authentication [48] which offers software libraries with which to authenticate your users into your app and into the database itself. One can authenticate users through use of password, phone number or he can use identity providers like Google, Facebook and others.

A user who gets authenticated with Firebase Authentication will get a unique UserID which can be used to give him secure access to his data (see Fig. 31). Also the connection with the database happens over a secure https protocol.

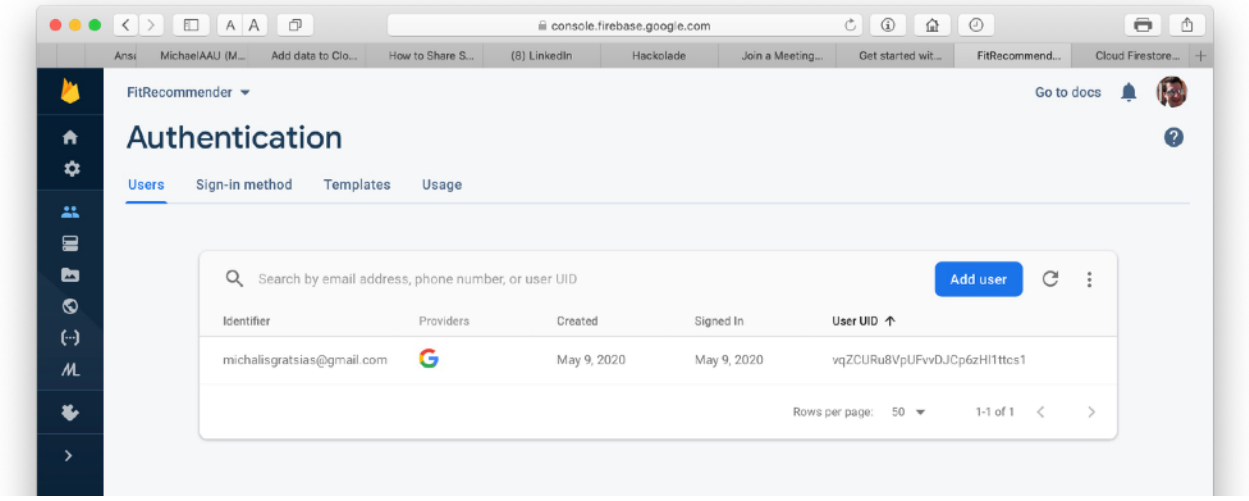


Figure 31, Firebase Authentication with Google IdP and User UID (screenshot)

The first actions that we took was to create a NoSQL data model (as seen already on Figure 26 of the previous chapter) and enter all our exercise data and the default user data into the database.

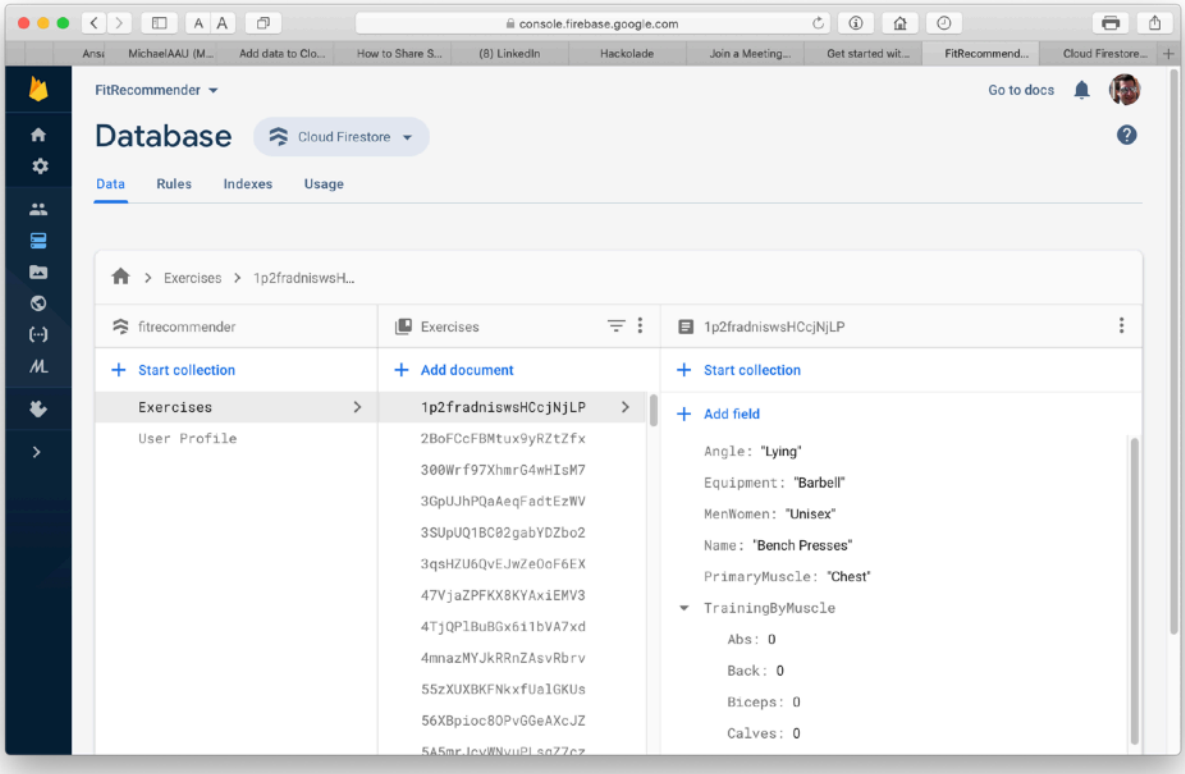


Figure 32, Database Collections (screenshot)

The result of these actions can be seen partly in Fig. 32 with our “Exercises” collection for exercise data and “User Profile” collection for user data. (Collections in NoSQL databases are equivalent to Tables in Relational databases and the records in NoSQL are called “documents” and they store objects of key-value pairs). Cloud Firebase supports documents of many different data types, including: maps, arrays, strings etc. Also each document is given a unique ID for easy reference.

After our data for the 165 exercises were entered into our database we had to use the provided software libraries in order to connect the database to our app and handle the authentication.

5.1.2 Android App Authentication UI and Android Studio

The app was developed with the Java Android SDK using the Android Studio IDE (Integrated Development Environment), which incorporates an intelligent code editor providing code analysis and suggestions among many other useful features. [49]

Our first actions involved: creating a Logo screen, setting up the Authentication UI, and connecting and reading data from our Database.

The UI of the Logo and Authentication screens can be seen in Figure 33. We have chosen two authentication methods: one is with the traditional email and password, while the other is with the Google Identity Provider. When Google is used, we need to choose our account and give our permission to share some of our personal information.

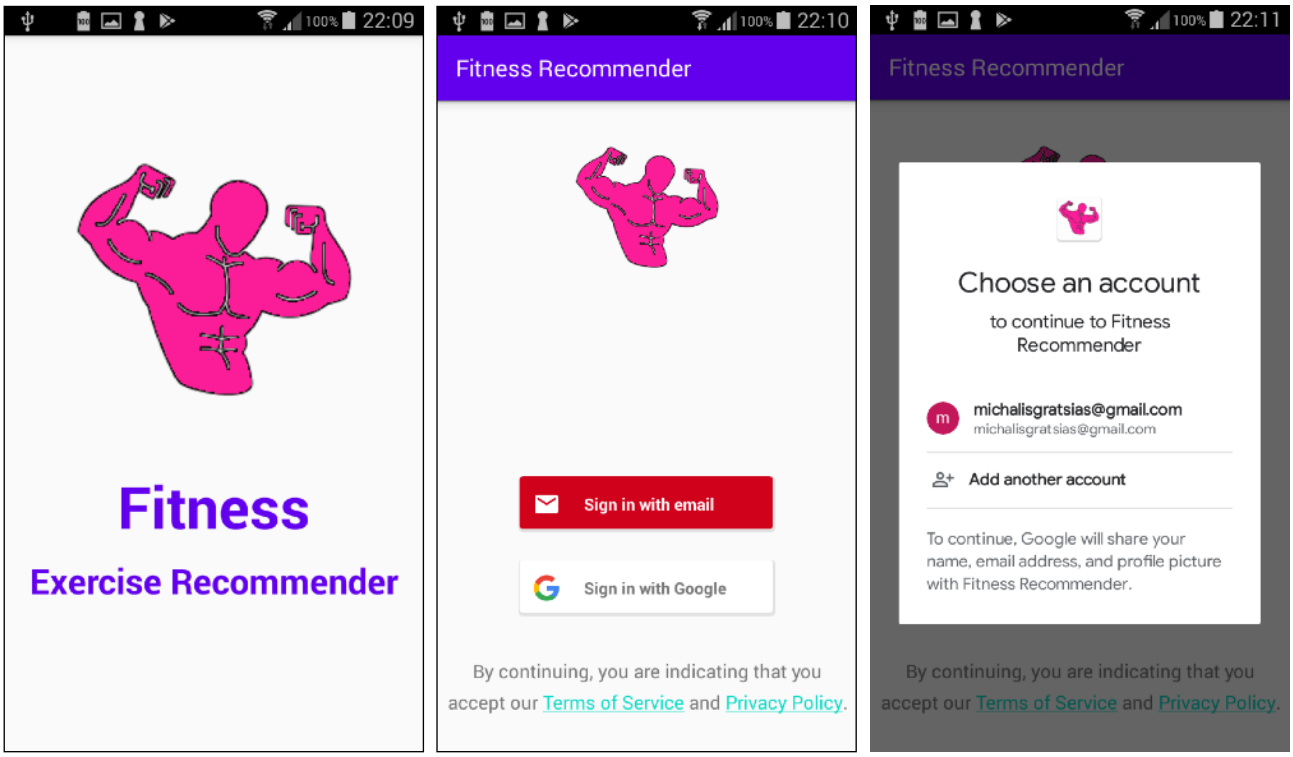


Figure 33, App Logo and Authentication screens (screenshots)

Next, in Figure 34 we show a snippet from the code used for reading the database data. Reading the Firestore database is a relatively simple action: We initialize our exercise list and we access an instance of the Firestore database. Then we specify our "Exercises" collection and use the get() method to retrieve all of the results. When this task is complete, we will iterate all the documents in the results in order to collect the data that we want and assign them to our Exercise objects and Exercise List.

```
mExerciseList = new ArrayList<>();
FirebaseFirestore db = FirebaseFirestore.getInstance();
db.collection("Exercises")
        .get()
        .addOnCompleteListener( (task) -> {
            if (task.isSuccessful()) {
                for (QueryDocumentSnapshot document : task.getResult()) {
                    Exercise exercise = new Exercise();
                    exercise.setExerciseId(document.getId());
                    exercise.setName((String) document.getData().get("Name"));
                    . . .           . . .           . . .
                    mExerciseList.add(exercise);
                }
            } else {
                Log.d(TAG, "Error getting documents: ", task.getException());
            }
        }
    });
```

Figure 34, Code snippet for reading our data from the Firestore database

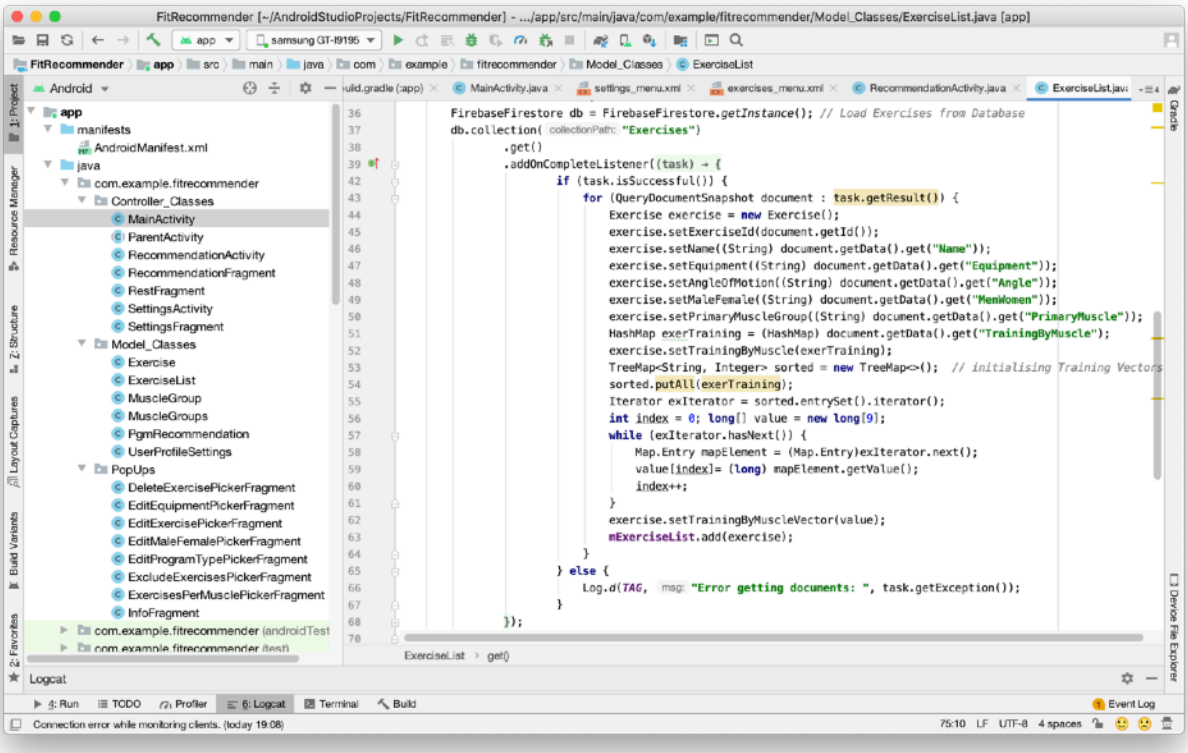


Figure 35, Reading from Database as seen on Android Studio (screenshot)

To get an idea of how this looks on Android Studio, we provide also a screenshot of the code in Figure 35.

5.1.3 Recommendations and User Interface

When the data have been read from the database, the app will make the first recommendation of an exercise program based on the default values set from beforehand in the user's profile. The algorithm for these recommendations has been described in section 4.2.2 and the full Java code can be found in Appendix E.

The app assumes by default that the user is a beginner in weight training and has no special preferences or limitations about equipment, exercises, etc. These default settings can be adjusted by the user at any time, as will be described later.

The first recommended program is a full-body program that is done within one day and repeated after a day of rest but with different exercises, as described in section 2.2.1. This program in itself, which takes 4 days in total (including the rest days) can be repeated for over a month before a completely new exercise program is requested.

As one can see in Fig.36, for each separate day the exercises are presented as a list of items which are scrollable and they contain a pictorial illustration of the exercise and the description of the exercise starting with the main muscle group that is affected by it. (The exercise images were taken from the JEFIT website https://www.jefit.com/exercises/ just for the demonstration purposes of this thesis).

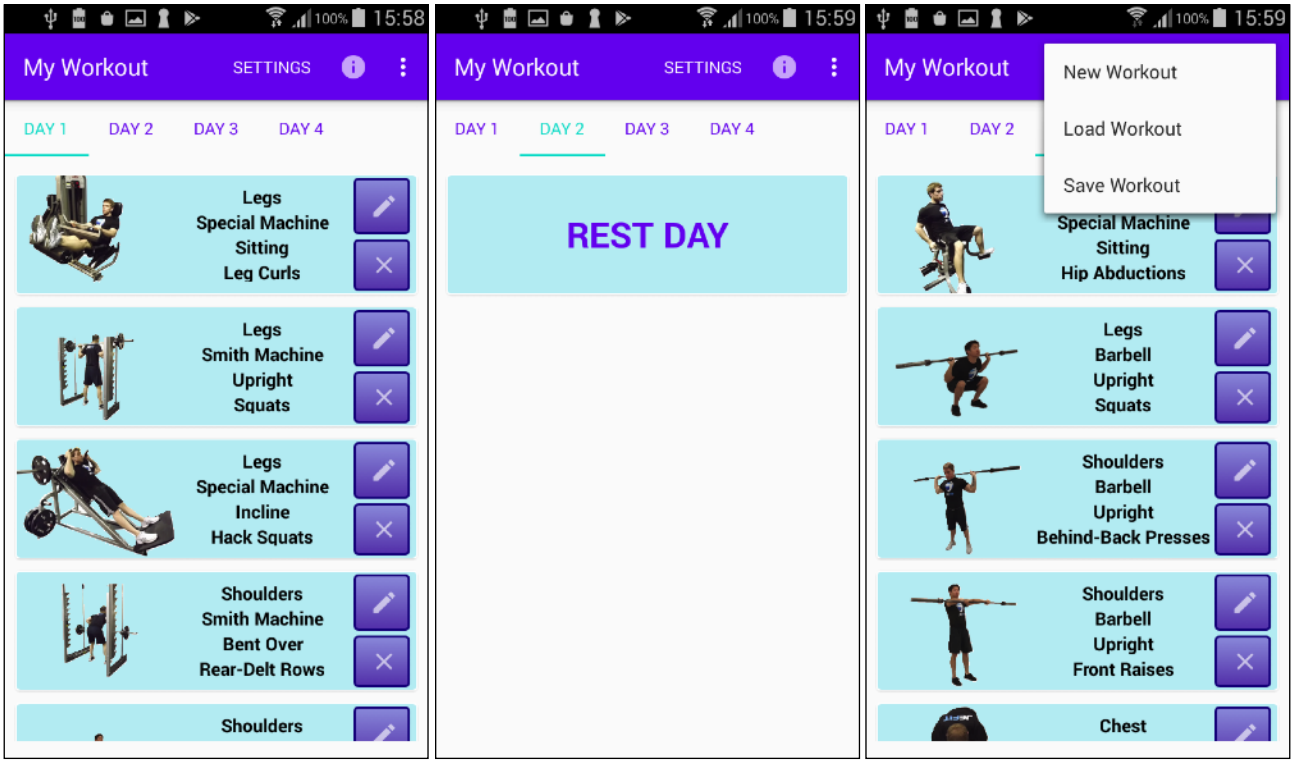


Figure 36, Program Recommendations (screenshots)

On top of the screen, one can see the “ToolBar” with Menu buttons that help in the navigation through the app. Right under that, there are Tabs for each day of the recommended program and these can be used to navigate through the program as well as to indicate which specific day we see on the screen.

The exercises themselves include two buttons each. One is used for Replacing a recommended exercise and the other is used for Deleting it from the program, according to our Requirements #9, #10 and #11. Additionally the user has the possibility to request a completely new program per Requirement #12 and this can be done by pressing the Menu button with the 3 dots and choosing “New Workout” from the drop-down list that appears. (see Fig.36 on the right)

To help the user understand better how to use the UI, we added a function per Requirement #15 with which when a user gives a “long-press” on a button (meaning that he holds it pressed for a second) then a little message appears which explains the function of the particular button, without actually activating (clicking) the button itself. This is a convention that is used often in Android apps.

Now we will explore some of the pop-ups of the Workout page. The first pop-up appears when we press the edit button to Replace an exercise (Fig. 37). What you see is a scrollable list of exercises with the original exercise on top and the recommended exercises following that, sorted based on the cosine similarity to the original exercise, as we described in section 4.2.3. The full Java code for this algorithm can be found in Appendix F.

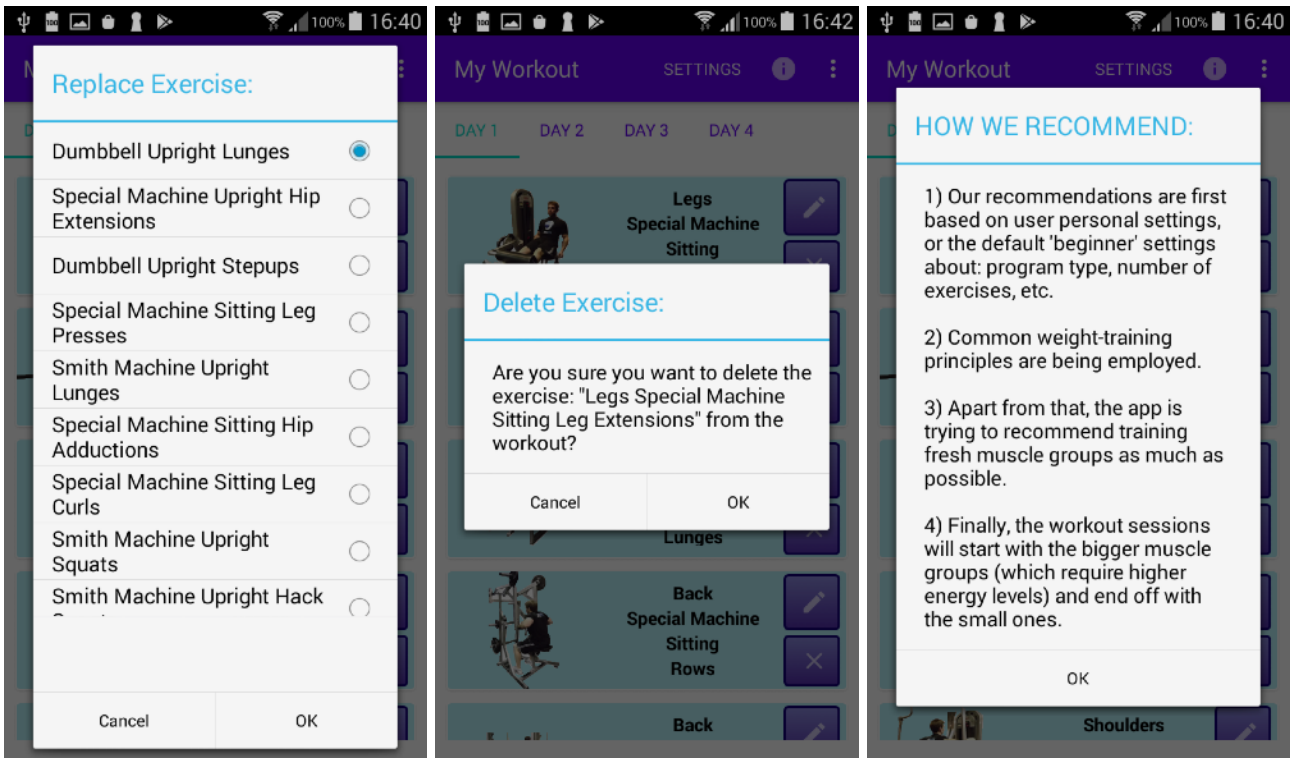


Figure 37, Pop-ups for Replace, Delete and Info (screenshots)

The next pop-up appears when we press the cancel button to Delete an exercise. The pop-up message will ask us to confirm the action before it actually takes place and then it will take the exercise out of the recommendation list.

The final popup will appear when we press the small “Info” button on the right side of the ToolBar. What that does is that it offers an explanation to the user about the way that the app is doing its recommendations, in order to build user trust and answer possible questions that the user may have. This follows also our Requirement #22.

5.1.4 Settings page and Pop-ups

When the user presses the “SETTINGS” Menu button on the ToolBar he will be directed to the page for his personal settings (see Fig. 38).

This is divided in 5 sections (as we described in section 3.4.2 on Personalisation). The first section concerns for whom are the exercises and the user is given 3 choices as can be seen in the pop-up (Fig.38 middle). Depending on the choice of the user, the exercises will be filtered to fit the user’s constraints.

The next section concerns non-available equipment and the user is given a choice of four different categories of equipment that may not be available at his gym, etc. (Fig. 38 to the right). Any exercises that include the use of equipment marked by the user, will be removed from the recommended programs.

Next, there is the choice about the type of program. As can be seen in Fig.39 the user is given 6 different types of programs to choose from. They are listed in terms of

level of advancement into the fitness activity and this choice affects how many days the program will last and how the exercises are distributed in each workout day.

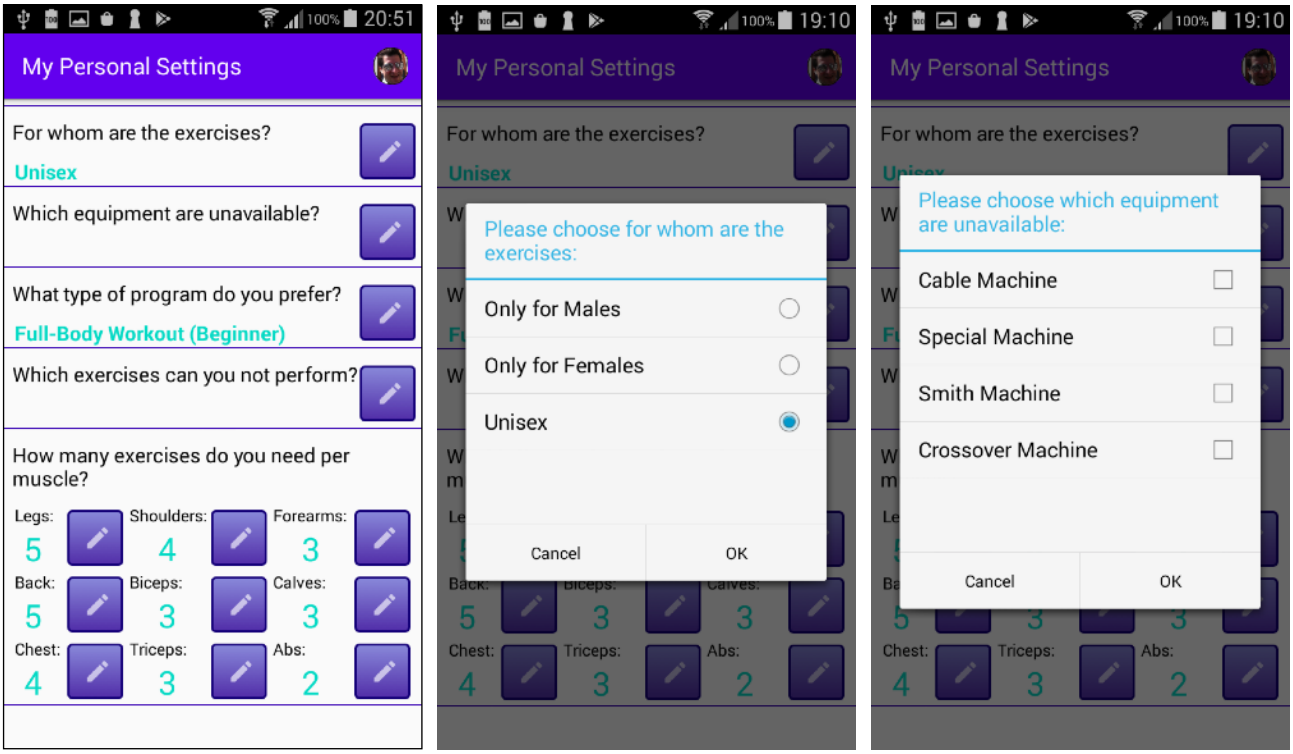


Figure 38, Settings page and Pop-ups (screenshots)

After that, the user is also given the choice to mark any exercises that he cannot, or does not want to perform for any reason (see Fig. 39 middle) and these exercises will be removed from the recommended programs also.

This exercise list includes all 165 exercises, so to make it easier for the user to find a specific exercise and per our Requirement #16, we sorted the list by muscle and exercise name.

Finally there is the final section where the user can choose how many exercises will be included in the recommended program, from each one of the muscle groups. For each muscle group there is a pop-up that will help the user to set the required number of exercises (see Fig.39 on the right).

We have already given default values to these exercise numbers as per our analysis in section 3.4.1.

And this completes the chapter on how our solution was implemented. The full code is included together with the delivery of this thesis and excerpts can be seen in Appendices E and F.

The next chapter will describe our final testing and evaluation of the hi-fidelity prototype solution with the help of more users.

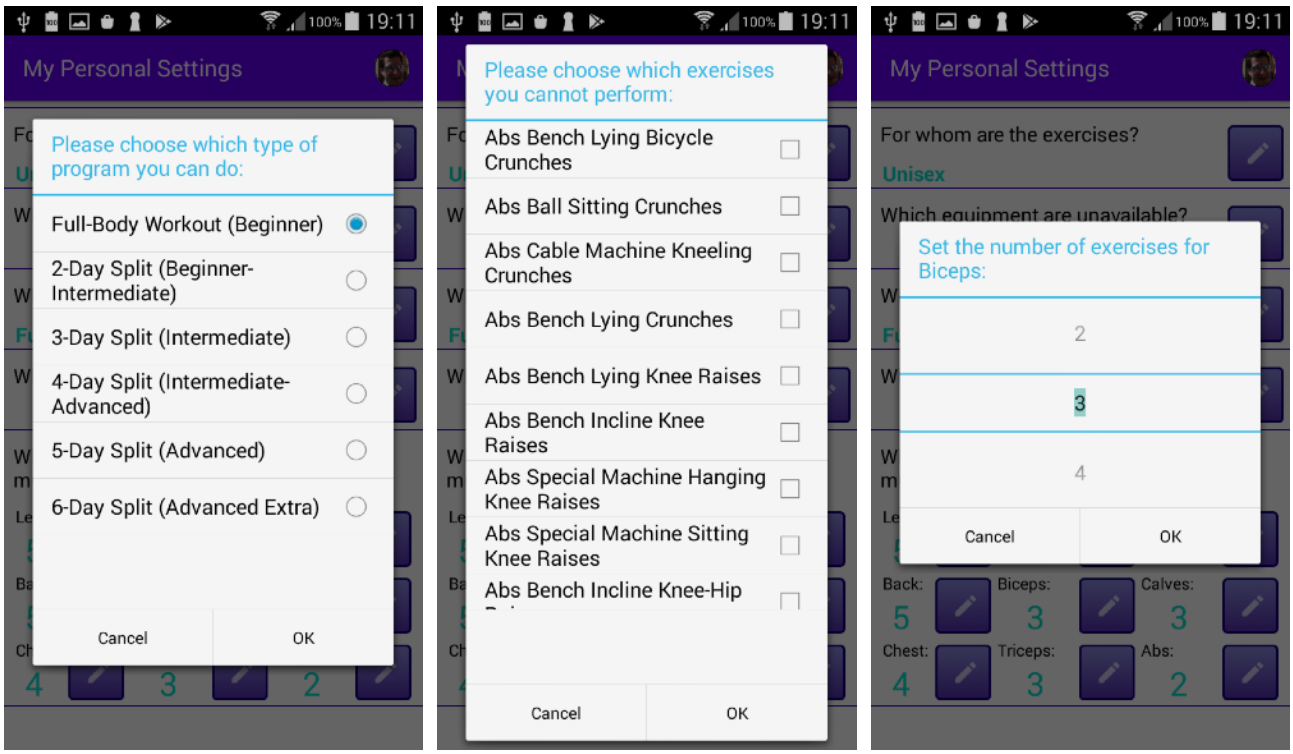


Figure 39, More Pop-ups for personalised settings (screenshots)

6. Testing and Evaluation

In order to control the quality of our code and ensure that our algorithms give the correct results, we conducted some initial code tests. Android provides two tools for such testing which are: Logcat and JUnit. [48]

Logcat is a messaging tool that you use to print out messages or stack traces and find out where a bug occurs. While JUnit is an automated unit-testing framework for Java. For reasons of efficiency and simplicity we used Logcat in our tests, but as part of future work JUnit should be used for more thorough unit testing.

6.1 Code Test: Filtering exercises

In the first phase of our recommendation algorithms, the full list of exercises is being filtered according to user requirements. To test that, we did a series of tests which among others they gave us information about how many exercises have been filtered.

The code for these tests can be found in Appendix E and the Logcat results can be seen in Figure 40.

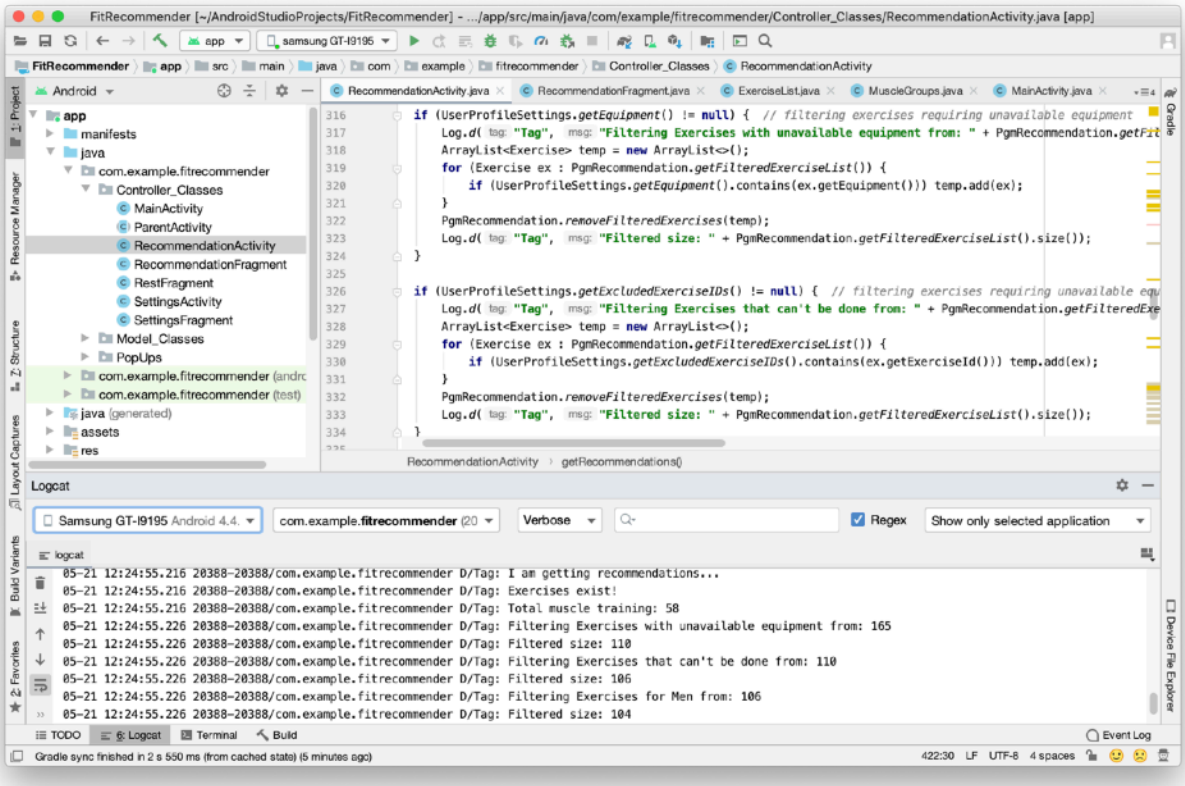


Figure 40, Logcat results for exercise filtering (screenshot)

Here we can see that the exercises were filtered from a total number of 165 to 110 after the user indicated certain unavailable equipment, then again from 110 to 106

because the user indicated 4 exercises that he could not perform and finally from 106 to 104 because the user did not want to perform 2 exercises that were labeled as: ”only for men”.

6.2 Code Test: Exercise recommendation algorithm

In the second phase of our recommendation system, an exercise program is recommended based on our training principles. One of the main principles is that the “seed” muscle groups that will be chosen first will be “fresh”, meaning that they will have the least amount of training.

To test that, we used the Logcat to log the amount of training that the muscles get from the exercises and then print it on the screen. (see Fig 41)

```
StringBuilder bigGroups = new StringBuilder();
for (int bigMuscle=0; bigMuscle < MuscleGroups.getBigMuscleGroups().size();
bigMuscle++) {
        bigGroups.append(MuscleGroups.getBigMuscleGroup(bigMuscle).getName()+" "+
        MuscleGroups.getBigMuscleGroup(bigMuscle).getTotalTraining()+" - ");
}
Log.d("Tag", "Sorting Big Muscles: " + bigGroups.toString());
```

Figure 41, Logging big muscles sorted by least training

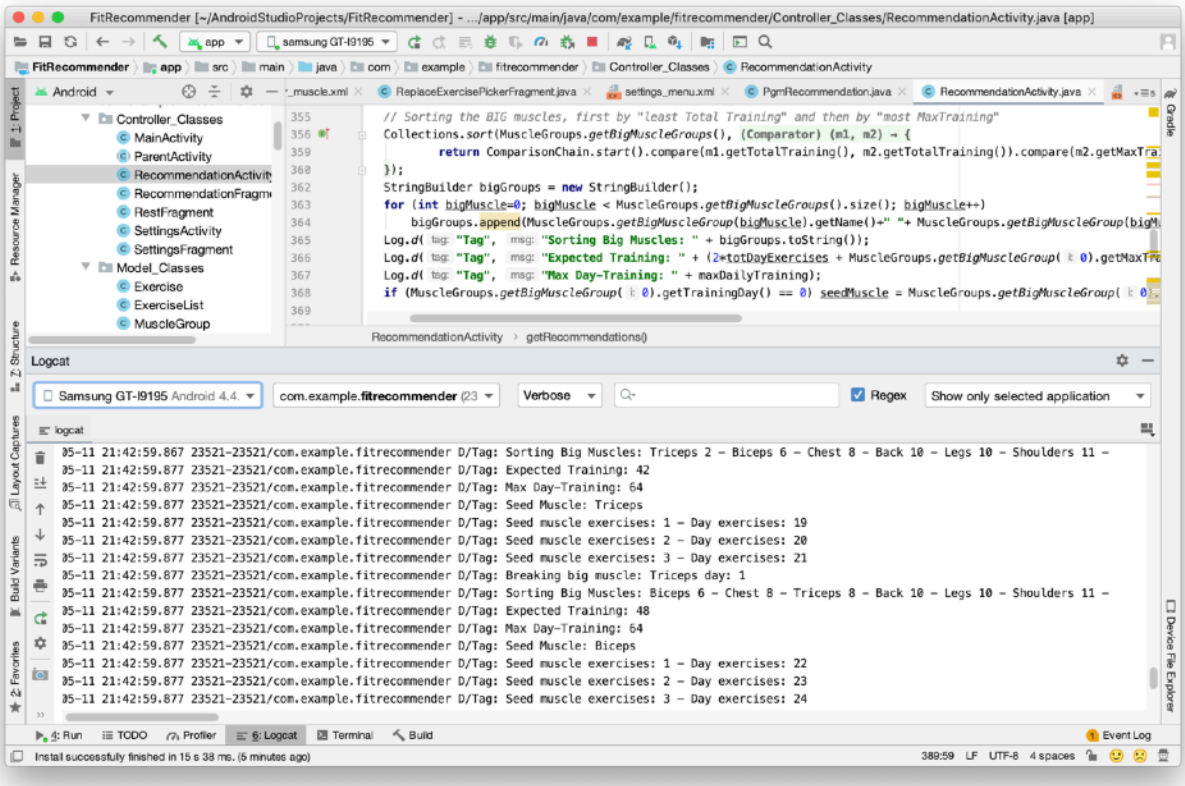


Figure 42, Logcat results for muscle training (screenshot)

The results can be seen in an example in Fig.42, where we get a list of the big muscles sorted by least training (Triceps 2 - Biceps 6 - Chest 8 - Back 10 - Legs 10 - Shoulders 11) and then we choose the "seed" muscle as the muscle with the least training (Seed Muscle: Triceps). For the full code, please check Appendix E.

6.3 Code Test: Exercise sorting by cosine similarity

In the third phase of our recommendations (see Req #9), it was important to test if the exercises we present to the user for replacing an existing exercise, were actually very similar to the original and if they are sorted by degree of similarity.

To conduct this test, we used Logcat to log to the screen the Vector components of the exercises recommended and their cosine similarity to the original exercise to be replaced (see Fig.43).

```
Log.d("Tag", "Vector : " + Arrays.toString(recommended.getTrainingByMuscleVector()));

Log.d("Tag", "Cosine Sim: " + cosineSimilarity(exercise.getTrainingByMuscleVector(),
                              recommended.getTrainingByMuscleVector()));
```

Figure 43, Logging Cosine Similarity

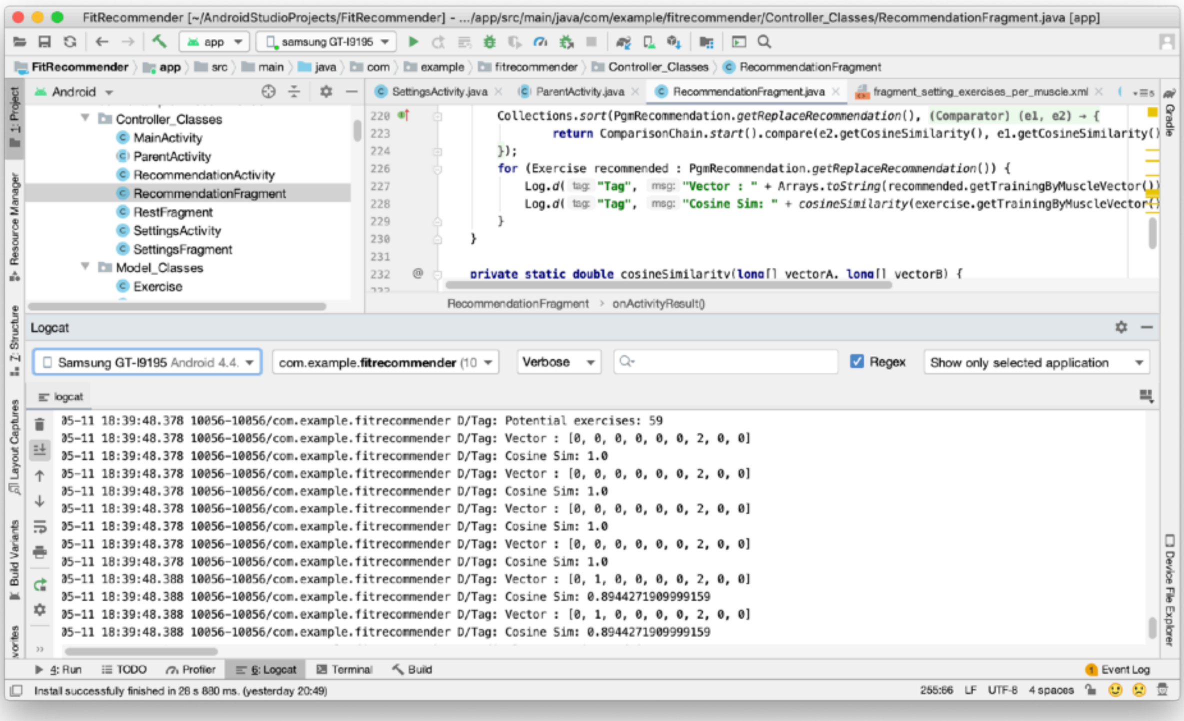


Figure 44, Logcat results for Cosine Similarity (screenshot)

As a result of that, we received a printout in Android Studio with the values (see Fig.44) where we can see that the first exercises recommended have a Cosine Similarity value of 1.0 with the original exercise, while later exercises have a value around 0.89 which is exactly what we wanted. For the full code please check Appendix F.

These kind of tests were done during the whole development phase to ensure the quality of the code and the algorithmic results were as we wanted.

The next thing that was done in order to evaluate our results, was to conduct some final user tests with our hi-fidelity prototype and see what the users have to say about it.

6.4 Final User Tests and evaluation

With the purpose of evaluating our high-fidelity prototype in terms of user interface layout and usability we conducted another series of user tests.

In these final user tests, we decided to change our approach a bit and conduct moderated tests instead of unmoderated tests. This meant that we would have online qualitative interviews where we can directly talk to our users and ask them questions, give them tasks and find out what they are thinking about our hi-fidelity prototype.

“Validately” was used again as a platform where we still had the possibility to conduct a few more free interviews.

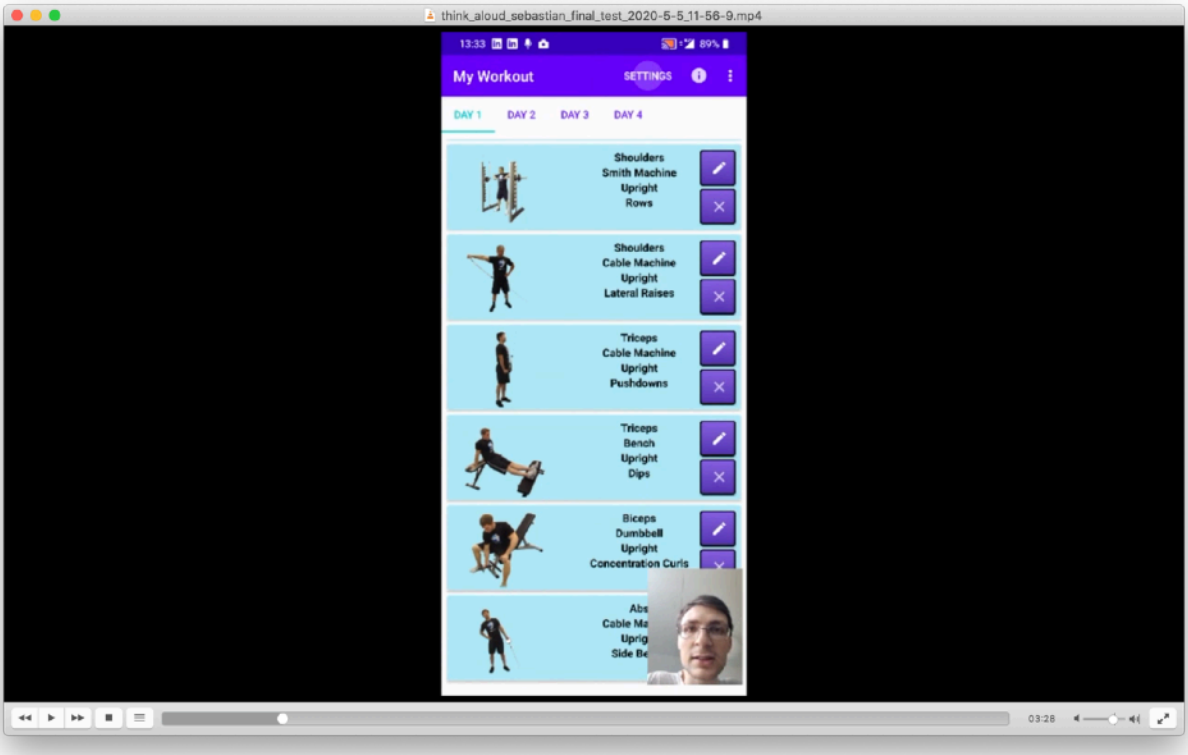


Figure 45, Moderated interviews with “Validately” (screenshot)

As can be seen in Fig.45, we had the possibility to record online the user's camera as well as the user's phone screen while the user was performing our test. In some cases the user preferred to only have his voice recorded together with his phone screen, but in every case we could have a conversation with the user in real time during the tests.

The way we conducted these tests is that we would tell the users to download our app and then give the users certain tasks to perform with the hi-fidelity prototype and find out their thoughts and comments in a "Think Aloud" test.

There were in total 3 moderated tests conducted (two on males and one on a female) where the users were to perform the following tasks:

1. Log on into the application with either an email and password or with the Google IdP.
2. Check through the whole exercise plan recommendation that they were given based on the default values.
3. Apply personal preferences to it, by setting some equipment as "unavailable".
4. See and verify the results after the changes that they made.
5. Replacing an exercise that they may not like with a different one.
6. Deleting an exercise and checking the results.
7. Receiving a completely new program recommendation.
8. Log out from the app, while their preferences are being stored in their account.

At the end of these tasks they were also asked to give their comments and opinions about the app, the amount of flexibility they had to make changes and the recommendations they got.

Results from the tests

All the participants could download our app on their Android phone and then download also the app from "Validately" which would help share their screen, microphone and camera with us. We were then communicating with them online through "Validately" website and we could hear and often also see each other through a small window. (Fig.45)

All the participants passed the tasks that we gave them and they gave us also valuable qualitative comments to think about, also for future work.

The first thing we found out from these tests was that the User Interface looks different from phone to phone, when different versions of the Android operating system is being used. We tested the app in 5 different OS versions and phones (3 phones from the users and 2 phones belonging to us) and discovered that in some cases our code had to change to accommodate for these API differences.

Another error that was found was that when exercises were deleted from a recommendation and the user returned back to the same page after a while, the exercises were still there, because a line was missing from our code that caused the deletion to not become permanent.

One user asked for the default values in the user settings to be always visible to the users so they can get back to them and set them when they want to.

Also, it was mentioned that it was not always obvious when a change had been made in the UI (eg. when an exercise was replaced, or a new program was created) and additional sound effects as well as notifications were added in order to accommodate for that point.

Overall, the tests added valuable contribution to the final prototype and we also got some nice final comments from users (see video recordings in the attachments to this report).

For example, user Sebastian (a DTU Masters student in Data Analysis) mentioned during the test:

"**The app is well thought. It has an interesting and clean layout which I appreciate.**" (23:40) and "**I especially like the layout. Blue color encourages you to do exercises**." (13:15) Also, "**I appreciate that you tried to put real images of how it would look. So much better than having the text!**" (20:50)

And about the recommendations, he said: "**Oh, this is very cool! Is there some machine learning behind it, or something? Very, very nice. I like it!**" (10:30)

Another user: Ilias from Greece mentioned: "**It is very simple. I can organize my workout easily. Very helpful, good application!**" (6:50)

Conclusion

The final qualitative user tests helped us to validate our results and correct a few final things that were missed in the first tests. Of course this is not the end for this project, because there are a lot of improvements that can be done in the future as it will be discussed in the following closing chapters. But we got the certainty that our current solution is doing what we expected it to do, in the way that it was supposed to do it.

Next, we will make a discussion of things learned from doing this and we will end with an overall conclusion and description of possible future work.

7. Discussion

Now that we implemented and tested our solution, we can take an overall look and see some more things that we have learned from doing this.

Personally it was a great surprise to see from the beginning how other apps were doing (or not doing) recommendations for weight training and realising that almost none had automated this function. The user had to do a lot of manual work, often could not create a profile of his own, and the recommendations were limited or based only on things as popularity.

What was realised was that social recommendations was not the way to go when someone was serious with fitness. A professional athlete would never take recommendations from his friends, but only from expert coaches or authoritative books and sources. So, the area of domain knowledge had to be explored and techniques like constraint-satisfaction were found to be the most applicable to the situation.

Another thing learned is that the app was never meant to replace completely the personal trainers, because there are elements in personal training by human trainers that we did not even try to simulate. These elements include the continuous monitoring of the actions of the person who trains and the immediate response to these actions without requiring any direct input on the part of the person.

Advanced smart trackers could go more into this area and try to understand what the person is doing while training and perhaps use natural language to give guidance when for example a person executes an exercise wrongly or could give verbal encouragement when a person is achieving his set goals.

The app was intended more for those who do not require continuous monitoring and could train more or less on their own, but they do not know exactly how to go on and what should be their next step. This is where the app recommender can ‘shine’ the most, by giving useful directions when fitness enthusiasts start to get confused and start to rely on popularity.

The app could also be relevant to the trainers themselves who could use it to get fast ideas on workout plans they can propose to their clients during their coaching sessions, plans that would be personalised to each individual.

So I would say that there are a lot of training activities that recommender systems such as this can supplement, even though the human element may never be completely excluded in every case.

For sure there is a lot of room for improvement in such systems, so that they can become highly useful to their users to the point that they would not think of exercising without using something of the sort to log their progress and get guided along. We are all curious to see what the future will bring.

8. Conclusion

We started this project with the question: “How can a recommender-system application support fitness enthusiasts by producing automated and personalised weight-training exercise plans based on proven training principles?”

Additionally, we set some sub-questions like: Which recommender techniques can be used?, What type of domain information about fitness training will be needed?, How can the application be personalised?, Which recommendation algorithms are applicable?, How can the application be designed, implemented and tested?

In terms of recommender systems, we did an analysis of the various popular recommendation techniques and found that the Constraint-based systems are very applicable to this case as well as systems for doing Sequential Recommendations of items.

In terms of domain knowledge about weight training, we found out that we need to know about exercise principles: how many exercises need to be performed on which muscle groups and how often, what is a correct sequence that these exercises should have, also what equipment can be employed and what is the effect of these exercises on the muscle groups. We answered these questions by distilling the relevant information from various professional domain sources.

In regards to how the application can be personalised, we proposed 5 ways this can be done (of course there can be more) but we found out that these 5 ways already gave us a lot of flexibility and they were happily accepted by the users that tested it.

On the applicable recommendation algorithms, we found that Constraint satisfaction algorithms, Similarity heuristics and Cosine-similarity algorithms were the most fitting for our case. With these algorithms we can do the recommendations that we need in an automated way and save a lot of work and pondering by the users.

On how to design, implement and test the application, we conducted moderated and unmoderated user interviews, we collected our requirements, we used UML diagrams and we built various prototypes until we had something that was highly workable as an Android fitness application.

By the end, we had achieved all of our set requirements (with the only exception the extra function to load and save workouts) and we had an app that offered a basic level of automation and flexibility that many existing apps on the market were not able to offer us.

9. Future Work

In the prospect that this project would be further developed in the future, there are some things that would need to be done in order to improve the overall service. We will sort them by the titles of Additional functionality for the users and System capabilities for what improvements the overall system may need.

Additional Functionality:

Creating personal detailed workouts: Users could also have a possibility to create their own workouts from scratch, if that is what they wish, and they could keep track of other metrics like the amount of repetitions and sets they perform for each exercise, the weight that they use, and others.

Storing workout data: In terms of functionality, another thing that would be useful for future work is permitting users to store their exercise plans into the database and retrieve them in the future when they need them again. The user settings are already stored and this can be extended to the user workouts themselves.

Progress graphs: Users could be able to create graphs of various metrics related to their exercises and in this way be able to see their progress over time in achieving their training goals.

History tracking: Historical data of completed workouts and metrics could also be recorded and user achievements could be shared on social media for others to see, if the users so requested.

More detailed exercise information: Additional types of exercises and high definition videos could be added, for example for people working out from home who have no equipment at all, or for people who want to work out with kettle bells, medicine balls and other related activities. Videos could give better explanations on how exactly the exercises should be performed.

System capabilities:

Many User Profiles: Right now one user profile is kept for testing purposes, but this should be extended to hundreds or thousands of user profiles being serviced at the same time as the app scales up. Firestore can store up to 1GB of data for free but after that you need to have a “pay as you go” payment plan which will be adjusted to the exact amount of your customers and their data. Cloud solutions are very flexible in these matters so they would be preferable for scaling.

User Personal Data: Although the Cloud Database provides secure connections, the personal information of users stored in it will need to be further protected per GDPR regulations. That could include the pseudonymisation of user data as well as other actions that would ensure user privacy rights are maintained according to the European regulation, such as the right to be forgotten and others.

Broader User Testing: The app will need to get a lot more testing as more and more functionality is added to it and as more and more users become interested in it and would like to use it on a regular basis. These tests can be conducted on a larger scale with many users using the system in real time, by recording the user behaviour and evaluating user acceptance of the recommendations given.

Motion tracking: There could be a possibility to use sensors in order to track user movements and location as previously mentioned, in order to understand what the user is doing and perhaps correct possible errors or give needed acknowledgements and encouragement while the user is performing exercises.

More recommendation methods: The app could use the history data of its users, additional training principles, more personalisation methods and motion tracking in order to give better recommendations to its users. The more information one can collect about the subject, the more able one will be to use it for the benefit of users who miss this type of expert guidance in their daily fitness activities.

And this concludes the summary of our possible future work and concludes also this Thesis.

10. References

[1] Joe Weider, Book: *Joe Weider's Bodybuilding system*, CA, USA: Weider Health & Fitness, 1988, ISBN 0-945-797-00-1.

[2] Raquel Mc Lish, Bill Reynolds, Book: *Flex Appeal by Raquel*, NY, USA: Warner Books, 1984, ISBN 0-446-381-05-5.

[3] Ami Eisinger, reviewed by Daniel Dubnis, "The Only 7 Gym Machines Worth Using", 2019, available at: https://greatist.com/move/best-gym-machines#lat-pulldown (Accessed May 21, 2020).

[4] Robert Kennedy, ***The Encyclopaedia of Bodybuilding***, CANADA: Robert Kennedy Publishing, 2008, ISBN 1-552-100-51-0, pg. 77.

[5] Jesse Venticinque, "Why Strength-Training will lead the future of fitness", available at: https://www.fitbod.me/blog/2017/12/14/strength-training-will-lead-in-the-future-of-fitness (Accessed May 21, 2020).

[6] "Fitplan: Gym & Home Workouts", App Store Preview, https://apps.apple.com/us/app/fitplan-gym-home-workouts/id1064119547 (Accessed May 21, 2020).

[7] "JEFIT Workout Planner Gym Log", App Store Preview, https://apps.apple.com/us/app/jefit-workout-planner-gym-log/id449810000#?platform=appleWatch (Accessed May 21, 2020).

[8] "GymGoal Pro", App Store Preview, https://apps.apple.com/us/app/gymgoal-pro/id571824492 (Accessed May 21, 2020).

[9] "Fitbod Weight Lifting Workout", App Store Preview, https://apps.apple.com/us/app/fitbod-weight-lifting-workout/id1041517543 (Accessed May 21, 2020).

[10] Alexander Felfernig, Gerhard Friedrich, Dietmar Jannach, and Markus Zanker, "**Constraint-Based Recommender Systems**", in *Recommender Systems Handbook*, NY, USA: Springer, 2015, ISBN 978-1-4899-7636-9, Chapter 5.1.

[11] Markus Schedl, Peter Knees, Brian McFee, Dmitry Bogdanov, and Marius Kaminskas, "**Playlist Generation Algorithms**", in *Recommender Systems Handbook*, NY, USA: Springer, 2015, ISBN 978-1-4899-7636-9, Chapter 13.5.3.

[12] Selye, H. "**Stress and the General Adaptation Syndrome.**" *BMJ*, vol. 1, no. 4667, 17 June 1950, pp. 1383–1392, www.ncbi.nlm.nih.gov/pmc/articles/PMC2038162/pdf/brmedj03603-0003.pdf, 10.1136/bmj.1.4667.1383. (Accessed May 21, 2020).

[13] Fleck, Steven. "**Non-Linear Periodization for General Fitness & Athletes.**" *Journal of Human Kinetics*, vol. 29A, no. Special-Issue, 1 Sept. 2011, pp. 41–45, www.ncbi.nlm.nih.gov/pmc/articles/PMC3588896/, 10.2478/v10078-011-0057-2. (Accessed May 21, 2020).

[14] Felfernig, A., Burke, R.: "**Constraint-based recommender systems: technologies and research issues".** In: *10th International Conference on Electronic Commerce*, ICEC'08, pp. 1–10. ACM, New York, NY, USA (2008).

[15] Felfernig, A., Isak, K., Kruggel, T.: "**Testing Knowledge-based Recommender Applications".** *OEGAI Journal 4*, 12–18 (2007).

[16] Tsang, E.: ***Foundations of Constraint Satisfaction*.** Academic Press, London (1993).

[17] Aucouturier, J.J., Pachet, F.: "**Scaling Up Music Playlist Generation".** In: *Proceedings of the IEEE International Conference on Multimedia and Expo* (ICME 2002), pp. 105–108. Lausanne, Switzerland (2002).

[18] Logan, B.: "**Content-based Playlist Generation: Exploratory Experiments".** In: *Proceedings of the 3rd International Symposium on Music Information Retrieval* (ISMIR 2002), pp. 295–296. Paris, France (2002).

[19] Flexer, A., Schnitzer, D., Gasser, M., Widmer, G.: "**Playlist generation using start and end songs**". In: *ISMIR*, pp. 173–178 (2008).

[20] Fields, B.: "**Contextualize your listening: the playlist as recommendation engine".** Ph.D. thesis, Department of Computing Goldsmiths, University of London (2011).

[21] Knees, P., Pohle, T., Schedl, M., Widmer, G.: "**Combining Audio-based Similarity with Webbased Data to Accelerate Automatic Music Playlist Generation".** In: *Proceedings of the 8th ACMSIGMM International Workshop on Multimedia Information Retrieval* (MIR'06). Santa Barbara, CA, USA (2006).

[22] Pohle, T., Pampalk, E., Widmer, G.: "**Generating Similarity-based Playlists Using Traveling Salesman Algorithms".** In: *Proceedings of the 8th International Conference on Digital Audio Effects* (DAFx-05), pp. 220–225. Madrid, Spain (2005).

[23] Bonnin, G., Jannach, D.: "**Evaluating the quality of playlists based on hand-crafted samples".** In: *14th International Society for Music Information Retrieval Conference*, ISMIR (2013).

[24] Pauws, S., Eggen, B.: "**PATS: Realization and user evaluation of an automatic playlist generator".** In: *Proceedings of the 2nd International Symposium on Music Information Retrieval*, ISMIR (2002).

[25] Francesco Ricci, Lior Rokach, Bracha Shapira, "**Recommender systems: Introduction and challenges.**", in *Recommender Systems Handbook*, NY, USA: Springer, 2015, ISBN 978-1-4899-7636-9, Chapter 1, pp 1-34.

[26] Fabiana Lorenzi and Francesco Ricci. "**Case-Based Recommender Systems: A Unifying View**", pages 89–113. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. ISBN 978-3-540-31655-8. doi: 10.1007/11577935_5. URL

[27] Burke, R.: "**Hybrid web recommender systems**". In: *The Adaptive Web*, pp. 377–408. Springer Berlin / Heidelberg (2007)

[28] Bonnie A Nardi. "The use of scenarios in design". Hewlett-Packard Laboratories, Technical Publications Department, 1992.

[29] David L. Poole, Alan K. Mackworth: "**Artificial Intelligence: Foundations of Computational Agents**", 2nd Edition. Cambridge University Press (2017). ISBN-13: 978-1107195394

[30] Wikipedia, "Markov Chain", https://en.wikipedia.org/wiki/Markov_chain (Accessed May 21, 2020).

[31] Lee Schlenker, "K-NN - Getting to know your nearest neighbors", Medium, available at: https://towardsdatascience.com/k-nn-getting-to-know-your-nearest-neighbors-b60399dc0f32 (Accessed May 21, 2020).

[32] Berkeley Education, "Permutations and combinations", available at: https://math.berkeley.edu/~arash/55/6_3.pdf (Accessed May 21, 2020).

[33] SlideShare, "Precalculus warm up", available at: https://www.slideshare.net/roneick/pre-calculus-warm-up-42114 (Accessed May 21, 2020).

[34] Nielsen Norman Group, "Thinking Aloud: The #1 Usability Tool", available at: https://www.nngroup.com/articles/thinking-aloud-the-1-usability-tool/ (Accessed May 21, 2020).

[35] JustInMind, "All-in-one prototyping tool for web and mobile apps", available at: https://www.justinmind.com (Accessed May 21, 2020).

[36] usability.gov, "Wire-framing", available at: https://www.usability.gov/how-to-and-tools/methods/wireframing.html (Accessed May 21, 2020).

[37] UserZoom, "Usability Testing: Moderated or Unmoderated?", available at: https://info.userzoom.com/rs/293-RDJ-600/images/UserZoom_ebook_Moderated_vs_Unmoderated_02.pdf (Accessed May 21, 2020).

[38] Nielsen Norman Group, "Why you only need to test with 5 users", available at: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/ (Accessed May 21, 2020).

[39] Stellman, Andrew; Greene, Jennifer: *Applied Software Project Management*. O'Reilly Media, (2005), p. 113. ISBN 978-0-596-00948-9.

[40] Haughey, D.: "MOSCOW METHOD", (2014). Retrieved from https://www.projectsmart.co.uk/moscow-method.php (Accessed May 21, 2020).

[41] Asela Gunawardana and Guy Shani, "**Recommender System Properties.**", in *Recommender Systems Handbook*, NY, USA: Springer, 2015, ISBN 978-1-4899-7636-9, Ch. 8.3**,** pp. 281–304.

[42] Wikipedia, “Cosine Similarity”, https://en.wikipedia.org/wiki/Cosine_similarity (Accessed May 21, 2020).

[43] Xavier Amatriain and Josep M. Pujol, “**Similarity measures.**”, in *Recommender Systems Handbook*, NY, USA: Springer, 2015, ISBN 978-1-4899-7636-9, Ch.7.2.1**,** page 230.

[44] The Development House, “Prototyping Model”, available at: http://developmenthouse.blogspot.com (Accessed May 21, 2020).

[45] Saunders, M., Lewis, P., & Thornhill, A.: “Using Secondary Data”, *Research methods for business students*. Harlow: Financial Times Prentice Hall. (2009), Chapter 8.

[46] Saunders, M., Lewis, P., & Thornhill, A.: “Collecting Primary Data Using semi-structured, in-depth and group interviews.”, *Research methods for business students.* Harlow: Financial Times Prentice Hall, (2009), Chapter 10.

[47] Google, “Firebase Authentication”, available at: https://firebase.google.com/docs/auth (Accessed May 21, 2020).

[48] Google, “Cloud Firestore”, available at: https://firebase.google.com/docs/firestore (Accessed May 21, 2020).

[49] Android, “Android Studio”, available at: https://developer.android.com/studio/features (Accessed May 21, 2020).

[50] Braze, “Logcat And JUnit: An Unstoppable Combination For Android Tests”, available at: https://www.braze.com/perspectives/article/logcat-junit-android-tests (Accessed May 21, 2020).

[51] Usabilitybok, “Cognitive Walkthrough”, available at: https://www.usabilitybok.org/cognitive-walkthrough (Accessed May 21, 2020).

[52] Google, “Firebase pricing plans”, available at: https://firebase.google.com/pricing (Accessed May 21, 2020).

[53] Bluepiit, “Demystifying Hybrid Recommender Systems And Their Use Cases”, available at: https://www.bluepiit.com/blog/demystifying-hybrid-recommender-systems-and-their-use-cases/ (Accessed May 21, 2020).

[54] Pablo Castells, Neil J. Hurley, and Saul Vargas, “**Novelty and Diversity in Recommender Systems**”, in *Recommender Systems Handbook*, NY, USA: Springer, 2015, ISBN 978-1-4899-7636-9, Ch.26.

[55] Asela Gunawardana and Guy Shani, “**Evaluating Recommender Systems.**”, in *Recommender Systems Handbook*, NY, USA: Springer, 2015, ISBN 978-1-4899-7636-9, Ch. 8**,** pp. 265–274.

[56] Ian Sommerville, “Design and Implementation”, in *Software Engineering*, 10th edition, England: Pearson Education Limited, 2016.

11. Appendices

A. Exercise Properties Table

This table includes 165 variations of exercises for 9 muscle groups and several other exercise properties used as part of our Knowledge-Base for the Weight Training Domain. The data were collected from Fitness publications: [1], [2], [4].

Exercise	Muscle Group	Men/ Females	Extra muscle groups	Variations of Angle	Equipment variation	Total Variations
Bench Press	Chest (Pecs)	Men/ Females	Triceps/ Shoulders	Flat/Incline/ Decline	Barbell/ Dumbbell/ Smith Machine	9
Chest Bentover Dips	Chest (Pecs)	Men/ Females	Triceps	-	Parallel Bars	1
Pullovers	Chest (Pecs)	Men/ Females	Back	-	Barbell/ Dumbbell/ Cable Machine	3
Cable Flyes	Chest (Pecs)	Men/ Females	-	Upright/Bent Over	Cable Machine	2
Dumbbell Flyes	Chest (Pecs)	Men/ Females	-	Flat/Incline/ Decline	Dumbbell	3
Chest Press	Chest (Pecs)	Men/ Females	Triceps/ Shoulders	-	Special Machine	1
Pec Deck Flyes	Chest (Pecs)	Men/ Females	-	-	Special Machine	1
Punch Press	Chest (Pecs)	Men/ Females	Shoulders	-	Dumbbell/ Cable Machine	2
Pushups	Chest (Pecs)	Men/ Females	Triceps/ Shoulders	-	-	1
Upright Dips	Outside Arms (Triceps)	Men/ Females	Chest	-	Parallel Bars/ Bench	2
Kickbacks	Outside Arms (Triceps)	Men/ Females	-	-	Dumbbell/ Cable Machine (A)	2

Exercise	Muscle Group	Men/ Females	Extra muscle groups	Variations of Angle	Equipment variation	Total Variations
Pushdowns	Outside Arms (Triceps)	Men/ Females	-	-	Cable Machine	1
Close-Grip Bench Press	Outside Arms (Triceps)	Men/ Females	Chest	-	Barbell/ Smith Machine	2
Triceps Extensions	Outside Arms (Triceps)	Men/ Females	-	Upright/Lying	Barbell/ Dumbbell/ Cable Machine/ Special Machine	8
Bentover Triceps Extensions	Outside Arms (Triceps)	Men/ Females	-	-	Cable Machine	1
Bent-over Row	Back (Lats)	Men/ Females	Biceps	-	Barbell/ Dumbbell/ Cable Machine/ Smith Machine/T-Bar Machine	5
Chinups/ Pulldowns	Back (Lats)	Men/ Females	Biceps	Front/Rear	Machine/ Cable Machine	4
Horizontal Pullup	Back (Lats)	Men/ Females	Biceps	-	Special Machine	1
Lying Row	Back (Lats)	Men/ Females	Biceps	-	Dumbbell	1
Seated Row	Back (Lats)	Men/ Females	Biceps	-	Machine/ Cable Machine	2
Straight Arm Pulldowns	Back (Lats)	Men/ Females	Chest	-	Cable Machine	1
Deadlifts	Back (Lats)	Men/ Females	Legs	-	Barbell/ Smith Machine	2
Goodmornings	Back (Lats)	Men/ Females	Legs	-	Barbell/ Smith Machine	2
Hyperextensions	Back (Lats)	Men/ Females	-	-	Special Machine	1
Shrugs	Back (Traps)	Men	-	-	Barbell/ Dumbbell/ Cable Machine	3

Exercise	Muscle Group	Men/ Females	Extra muscle groups	Variations of Angle	Equipment variation	Total Variations
Arm Curls	Inside Arms (Biceps)	Men/ Females	-	Upright/ Incline	Barbell/ Dumbbell/ Cable Machine/ Crossover Machine	8
Concentration Curls	Inside Arms (Biceps)	Men/ Females	-	-	Dumbbell/ Cable Machine	2
Close-Grip Chinups/ Pulldowns	Inside Arms (Biceps)	Men/ Females	Back	-	Machine/ Cable Machine	2
Hammer Curls	Inside Arms (Biceps)	Men/ Females	Forearms	-	Dumbbell/ Cable Machine	2
Reverse Arm Curls	Forearms	Men/ Females	Biceps	-	Barbell/ Dumbbell/ Cable Machine	3
Wrist Curls	Forearms	Men/ Females	-	-	Barbell/ Dumbbell	2
Reverse Wrist Curls	Forearms	Men/ Females	-	-	Barbell/ Dumbbell	2
Behind-Back Wrist Curls	Forearms	Men/ Females	-	-	Barbell	1
Behind-Back Press	Shoulders (Delts)	Men	Triceps	-	Barbell/ Smith Machine	2
Front Raises	Shoulders (Delts)	Men/ Females	-	-	Barbell/ Dumbbell/ Cable Machine	3
Military Press	Shoulders (Delts)	Men/ Females	Triceps	-	Barbell/ Dumbbell/ Cable Machine/ Smith Machine	4
Lateral Raises	Shoulders (Delts)	Men/ Females	-	-	Dumbbell/ Cable Machine/ Special Machine	3
Upright Rows	Shoulders (Delts)	Men/ Females	Back	-	Barbell/ Dumbbell/ Cable Machine	3

Exercise	Muscle Group	Men/ Females	Extra muscle groups	Variations of Angle	Equipment variation	Total Variations
Bentover Raises	Shoulders (Delts)	Men/ Females	-	-	Dumbbell/ Cable Machine	2
Rear Delt Row	Shoulders (Delts)	Men/ Females	Biceps	-	Barbell/ Dumbbell/ Cable Machine/ Smith Machine/ Special Machine	5
Ball crunches	Waist (Abs)	Men/ Females	-	-	Ball	1
Bicycle crunches	Waist (Abs)	Men/ Females	-	-	-	1
Cable crunches	Waist (Abs)	Men/ Females	-	-	Cable Machine	1
Crunches	Waist (Abs)	Men/ Females	-	-	-	1
Crunches with Legs (V-sits)	Waist (Abs)	Men/ Females	-	-	-	1
Vertical Leg Crunches	Waist (Abs)	Men/ Females	-	-	-	1
Leg Raises	Waist (Abs)	Men/ Females	-	Flat/Incline/ Vertical	Bench/ Special Machine	6
Knee Raises	Waist (Abs)	Men/ Females	-	Flat/Incline/ Vertical	Bench/ Special Machine	6
Leg-Hip Raises	Waist (Abs)	Men/ Females	-	-	-	1
Knee-Hip Raises	Waist (Abs)	Men/ Females	-	-	-	1
Plank	Waist (Abs)	Men/ Females	-	-	-	1
Rotating Plank	Waist (Abs)	Men/ Females	-	-	-	1
Seated Twisting/ Side Twist	Waist (Abs)	Men/ Females	-	-	Bar	1
Twisted Crunches	Waist (Abs)	Men/ Females	-	-	-	1

Exercise	Muscle Group	Men/ Females	Extra muscle groups	Variations of Angle	Equipment variation	Total Variations
Side Bend	Waist (Abs)	Men	-	-	Ball/ Dumbbell/ Special Machine	3
Squats	Legs (Quads)	Men/ Females	Back	-	Barbell/ Dumbbell/ Smith Machine	3
Front Squats	Legs (Quads)	Men/ Females	Back	-	Barbell/ Smith Machine	2
Hack Squats	Legs (Quads)	Men/ Females	-	-	Barbell/ Smith Machine/ Special Machine	3
Leg Extensions	Legs (Quads)	Men/ Females	-	-	Special Machine	1
Leg Press	Legs (Quads)	Men/ Females	-	Seated/ Incline/ Vertical	Special Machine	1
Lunges	Legs (Quads)	Men/ Females	-	-	Barbell/ Dumbbell/ Smith Machine	3
Step Up	Legs (Quads)	Men/ Females	-	-	Dumbbell	1
Hip Extensions	Legs (Glutes)	Females	-	-	Cable Machine/ Special Machine	2
Hip Raise	Legs (Glutes)	Females	-	-	Barbell	1
Leg Curls	Legs (Hamstrings)	Men/ Females	-	-	Cable Machine/ Special Machine	2
Stiff Leg Deadlifts	Legs (Hamstrings)	Men/ Females	Back	-	Barbell/ Dumbbell/ Smith Machine	3
Hip Abductions	Legs (Outer Quads)	Females	-	-	Cable Machine/ Special Machine	2

Exercise	Muscle Group	Men/ Females	Extra muscle groups	Variations of Angle	Equipment variation	Total Variations
Hip Adductions	Legs (Inner Quads)	Females	-	-	Cable Machine/ Special Machine	2
Calf Raises	Calves	Men/ Females	-	-	Barbell/ Dumbbell/ Special Machine	3
Calf Presses	Calves	Men/ Females	-	-	Special Machine	1
Seated Calf Raises	Calves	Men/ Females	-	-	Barbell/ Smith Machine/ Special Machine	3
						165

B. Exercise Training Effect Table

This table includes the 70 basic exercises and their training effect on different muscle groups set on a scale from 0 to 2. The data were collected from information in these publications: [1], [2], [4].

MUSCLE GROUPS/ Exercises	CHEST MUSCLE GROUP	TRICEPS MUSCLE GROUP	BACK MUSCLE GROUP	BICEPS MUSCLE GROUP	FOREARM MUSCLE GROUP	SHOULDER MUSCLE GROUP	ABS MUSCLE GROUP	LEGS MUSCLE GROUP	CALF MUSCLE GROUP
Bench Press	2	1	0	0	0	1	0	0	0
Chest Bentover Dips	2	1	0	0	0	0	0	0	0
Pullovers	2	0	1	0	0	0	0	0	0
Cable Flyes	2	0	0	0	0	0	0	0	0
Dumbbell Flyes	2	0	0	0	0	0	0	0	0
Chest Press	2	1	0	0	0	1	0	0	0
Pec Deck Flyes	2	0	0	0	0	0	0	0	0
Punch Press	2	0	0	0	0	1	0	0	0
Upright Dips	1	2	0	0	0	0	0	0	0
Kickbacks	0	2	0	0	0	0	0	0	0
Pushdowns	0	2	0	0	0	0	0	0	0
Close-Grip Bench Press	1	2	0	0	0	0	0	0	0
Triceps Extensions	0	2	0	0	0	0	0	0	0
Bentover Triceps Extensions	0	2	0	0	0	0	0	0	0
Bent-over Row	0	0	2	1	0	0	0	0	0
Chinups/ Pulldowns	0	0	2	1	0	0	0	0	0
Horizontal Pullup	0	0	2	1	0	0	0	0	0

Lying Row	0	0	2	1	0	0	0	0	0
Seated Row	0	0	2	1	0	0	0	0	0
Straight Arm Pulldowns	1	0	2	0	0	0	0	0	0
Deadlifts	0	0	2	0	0	0	0	1	0
Goodmornings	0	0	2	0	0	0	0	1	0
Hyperextensions	0	0	2	0	0	0	0	0	0
Shrugs	0	0	2	0	0	0	0	0	0
Arm Curls	0	0	0	2	0	0	0	0	0
Concentration Curls	0	0	0	2	0	0	0	0	0
Close-Grip Chinups/ Pulldowns	0	0	1	2	0	0	0	0	0
Hammer Curls	0	0	0	2	1	0	0	0	0
Reverse Arm Curls	0	0	0	1	2	0	0	0	0
Wrist Curls	0	0	0	0	2	0	0	0	0
Reverse Wrist Curls	0	0	0	0	2	0	0	0	0
Behind-Back Wrist Curls	0	0	0	0	2	0	0	0	0
Behind-Back Press	0	1	0	0	0	2	0	0	0
Front Raises	0	0	0	0	0	2	0	0	0
Military Press	0	1	0	0	0	2	0	0	0
Lateral Raises	0	0	0	0	0	2	0	0	0
Upright Rows	0	0	1	0	0	2	0	0	0
Bentover Raises	0	0	0	0	0	2	0	0	0
Rear Delt Row	0	0	0	1	0	2	0	0	0
Ball crunches	0	0	0	0	0	0	2	0	0

Bicycle crunches	0	0	0	0	0	0	2	0	0
Cable crunches	0	0	0	0	0	0	2	0	0
Crunches	0	0	0	0	0	0	2	0	0
Crunches with Legs (V-sits)	0	0	0	0	0	0	2	0	0
Vertical Leg Crunches	0	0	0	0	0	0	2	0	0
Leg Raises	0	0	0	0	0	0	2	0	0
Knee Raises	0	0	0	0	0	0	2	0	0
Leg-Hip Raises	0	0	0	0	0	0	2	0	0
Knee-Hip Raises	0	0	0	0	0	0	2	0	0
Plank	0	0	0	0	0	0	2	0	0
Rotating Plank	0	0	0	0	0	0	2	0	0
Seated Twisting/ Side Twist	0	0	0	0	0	0	2	0	0
Twisted Crunches	0	0	0	0	0	0	2	0	0
Side Bend	0	0	0	0	0	0	2	0	0
Squats	0	0	1	0	0	0	0	2	0
Front Squats	0	0	1	0	0	0	0	2	0
Hack Squats	0	0	0	0	0	0	0	2	0
Leg Extensions	0	0	0	0	0	0	0	2	0
Leg Press	0	0	0	0	0	0	0	2	0
Lunges	0	0	0	0	0	0	0	2	0
Step Up	0	0	0	0	0	0	0	2	0
Hip Extensions	0	0	0	0	0	0	0	2	0
Hip Raise	0	0	0	0	0	0	0	2	0
Leg Curls	0	0	0	0	0	0	0	2	0

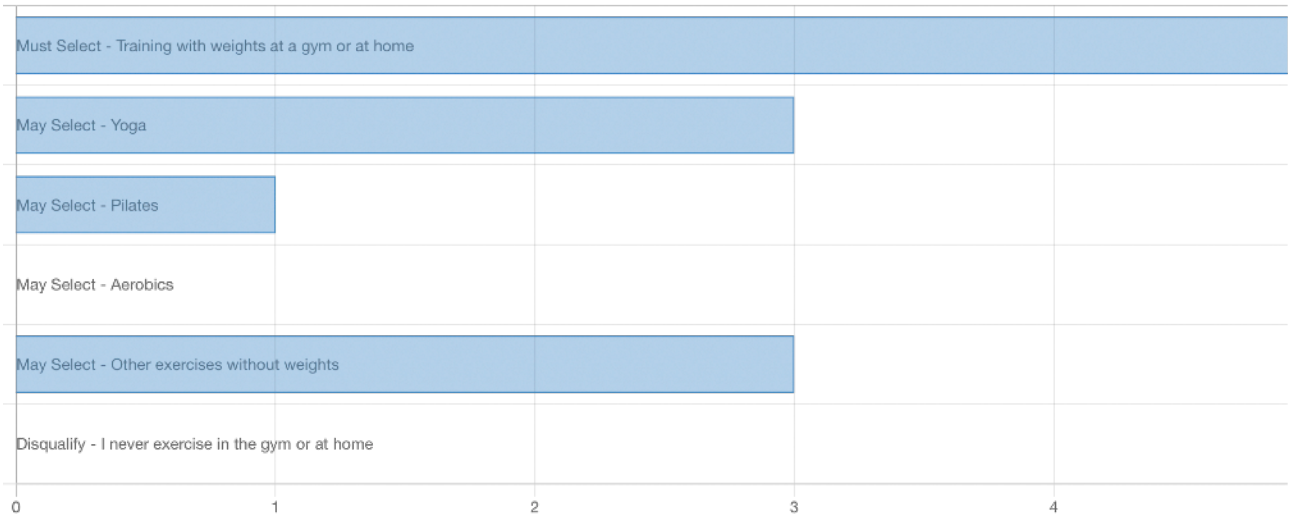
Stiff Leg Deadlifts	0	0	1	0	0	0	0	2	0
Hip Abductions	0	0	0	0	0	0	0	2	0
Hip Adductions	0	0	0	0	0	0	0	2	0
Calf Raises	0	0	0	0	0	0	0	0	2
Calf Presses	0	0	0	0	0	0	0	0	2
Seated Calf Raises	0	0	0	0	0	0	0	0	2

C. User Testing Analysis Summary

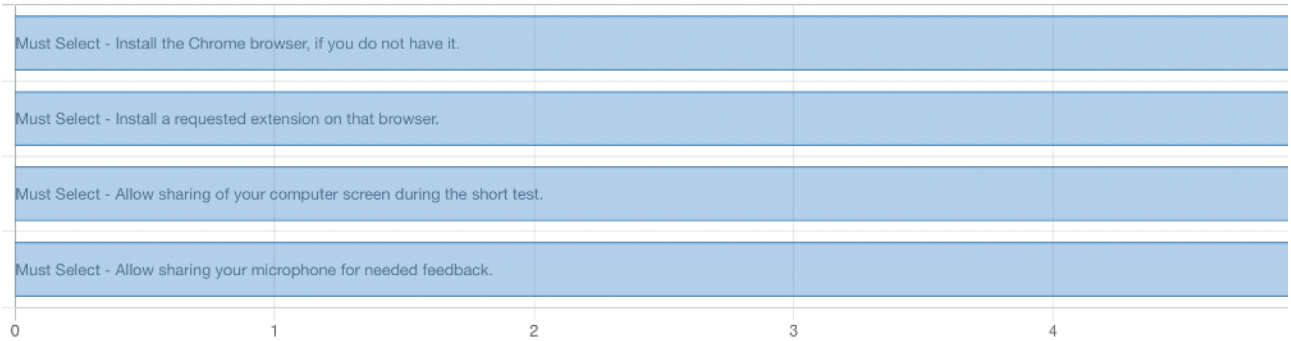
What follows is the results we got from “Validately” after conducting our 5 unmoderated interviews for User Testing. It includes the results from the Screener Questions, the Pre-study Questions, the Tasks and Task Questions, and also the Follow-up Questions. The full PDF file together with the 5 recorded videos of the unmoderated interviews can be found in the attached software folder.

Screener questions

Question 1: Which of these activities have you done in the past?

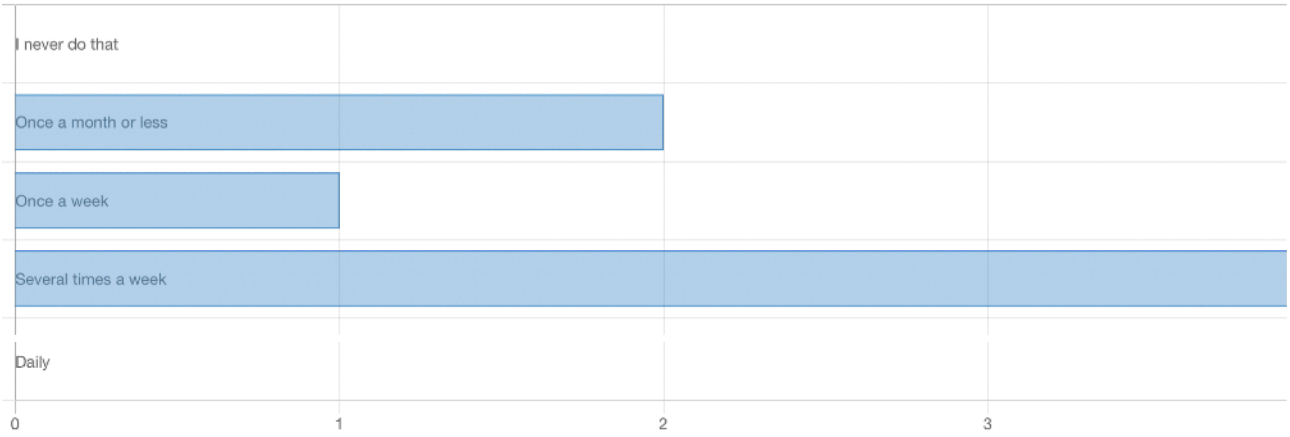


Question 2: Are you willing to do the following NECESSARY short steps (if requested) for conducting this test?



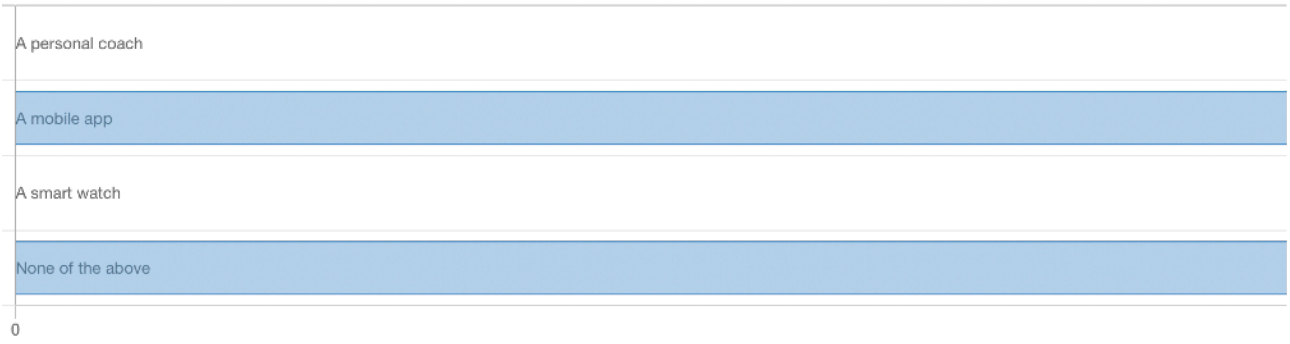
Pre-study questions

Question 1: How often do you do sports or exercise?



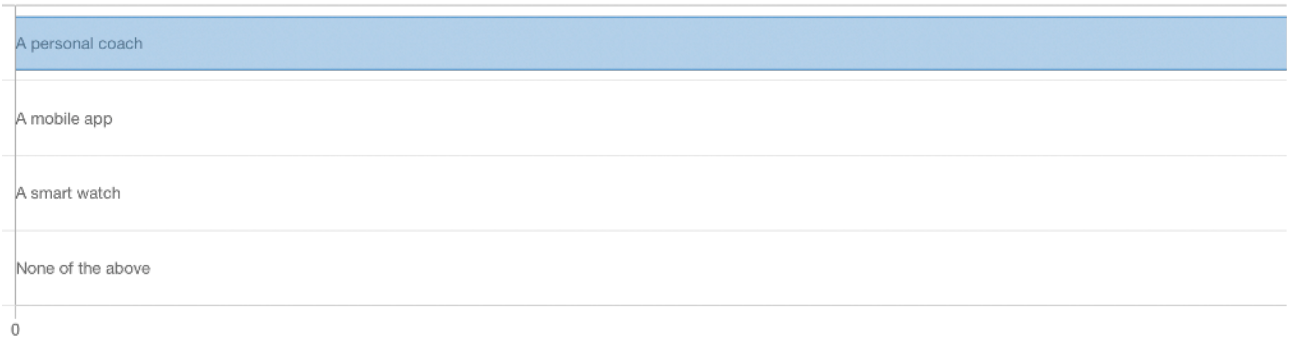
If answer is: Once a month or less

Question 1: When doing sports or exercise, which of these do you use to help you?



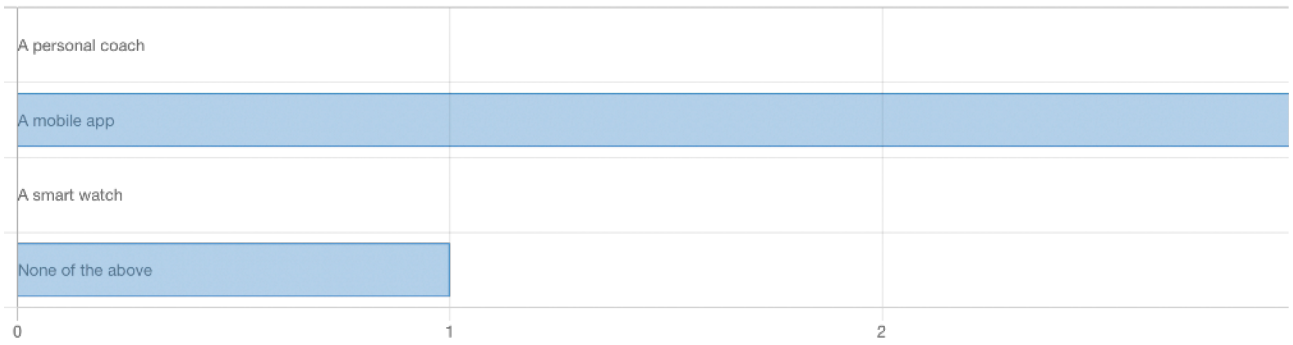
If answer is: Once a week

Question 2: When doing sports or exercise, which of these do you use to help you?

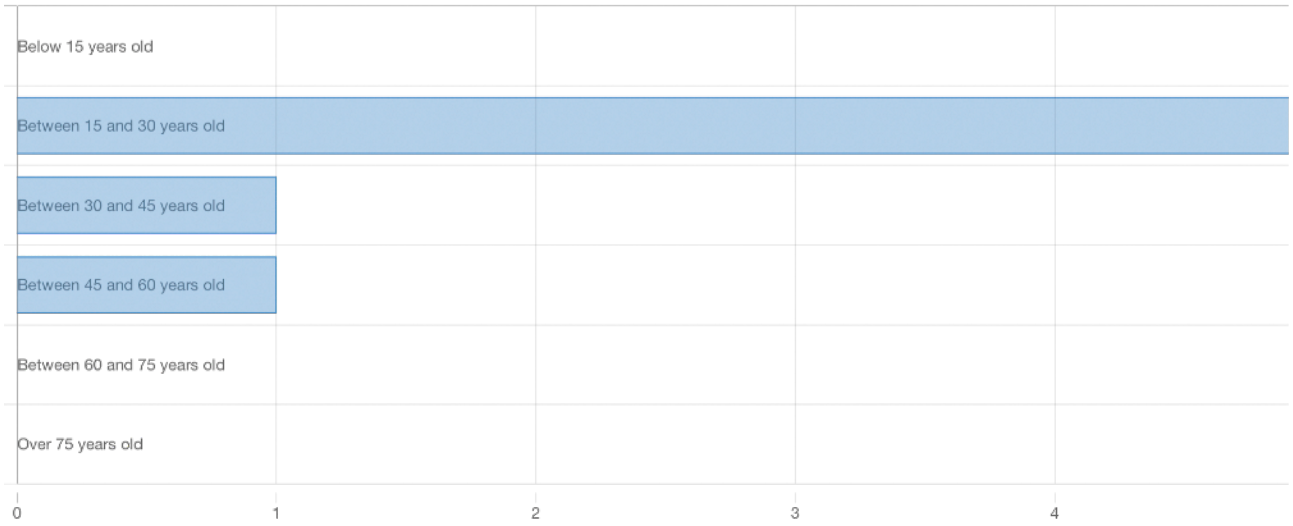


If answer is: Several times a week

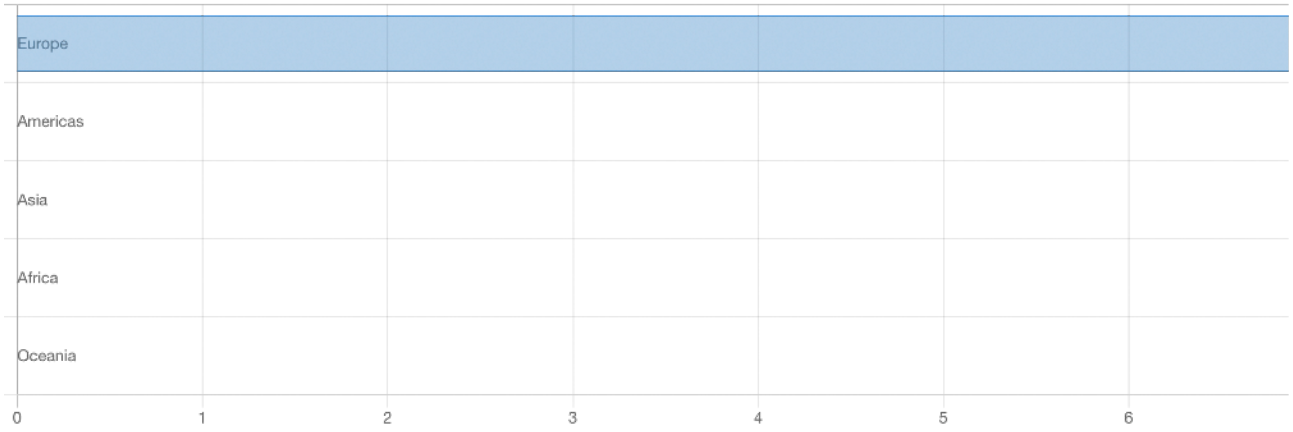
Question 3: When doing sports or exercise, which of these do you use to help you?



Question 2: What is your age bracket?

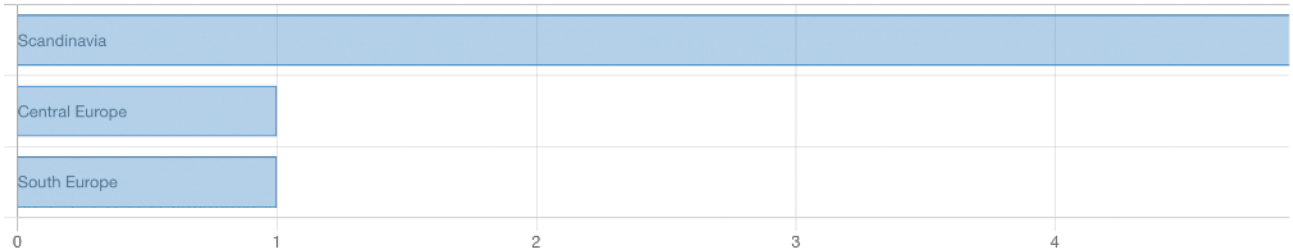


Question 3: Where do you live?



If answer is: Europe

Question 1: Which part of Europe?



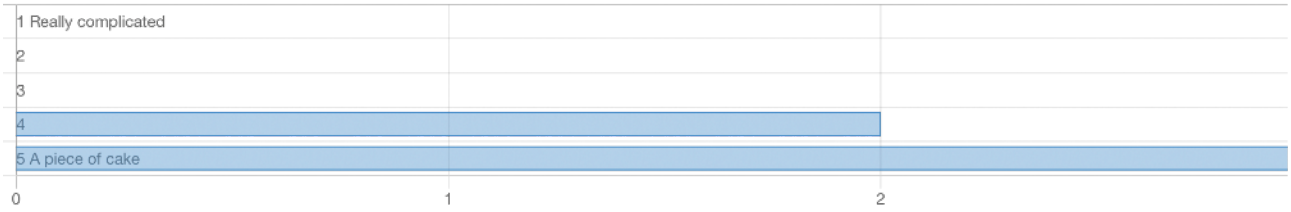
Task 1 Analysis

Task 1: Note: You are supposed to speak or "think aloud" during this test, so that any confusions about the design can be recorded. On the background you see a crude prototype of the "Log-In" screen of our application. (By using the -/+ buttons on the top, you be able to adjust the size of that prototype to fit your screen properly, if needed.) 1) Your first task will be to type-in an email and password (use a fake one) and press the button to "Log-In". When you have done it, you will just see the "Settings" screen and your first Task will be complete! To mark your Task as complete, you need to press the green "Show Task / Hide Task" button on the to right corner and you will see another button with which you can mark your first task as "Complete".

00:01:43 AVG TIME **5** PASSED **0** FAILED **0** UNGRADED

Task 1 Questions

Question 1: In a rate from 1 to 5, how easy did you find performing this task?



Task 2 Analysis

Task 2: You should be looking now at the "Settings" screen of our prototype. 2) Your task is to set that the exercises that will be recommended are intended for BOTH men and women (Unisex). Remember to speak your thoughts while performing the task and then press again the "Show Task" and "Complete" buttons, when you are done.

00:00:47 AVG TIME **3** PASSED **2** FAILED **0** UNGRADED

Task 2 Questions

Question 1: Were there any confusions while performing this task?

Respondent 2585357

None

Respondent 2593619

Nope, it's the top choice

Respondent 2594568

I found it confusing, i'm not sure that I were at the correct location. I couldn't find the button "Complete" and "show task" as described in the given task.

Respondent 2596050

no

Respondent 2597585

no

Task 3 Analysis

Task 3: 3) Your next Task is to set what equipment are available for weight training. You are supposed to mark ONLY these equipment as available: "Barbells", "Dumbbells", "Benches" and "Cable Machines". When you are done, mark the task as “Complete”.

00:00:14 AVG TIME **3** PASSED **2** FAILED **0** UNGRADED

Task 3 Questions

Question 1: Were there any difficulties in performing this Task, or do you have any suggestions?

Respondent 2585357

None

Respondent 2593619

No

Respondent 2594568

No difficulties.

Respondent 2596050

no

Respondent 2597585

no

Task 4 Analysis

Task 4: 4) The next Task is to choose the type of Training Program. For the purpose of this Test, you are supposed to choose a "4-days split routine" program, which takes 4 days to complete. Keep thinking aloud (speaking your thoughts) and again when you a done mark the Task as "Complete".

00:00:16 AVG TIME **4** PASSED **1** FAILED **0** UNGRADED

Task 4 Questions

Question 1: Did you encounter any difficulties or do you have any suggestions to this?

Respondent 2585357

None

Respondent 2593619

no

Respondent 2594568

Not really, but I had not sure what a split routine means, maybe add a small info box about it.

Respondent 2596050

no

Respondent 2597585

no

Task 5 Analysis

Task 5: 5) The next Task is to Set if there are any exercises that you cannot do for some reason. For this Test you need to choose only the exercise called: "Squats". When done, mark the Task "Complete".

00:00:17 AVG TIME **3** PASSED **2** FAILED **0** UNGRADED

Task 5 Questions

Question 1: Any difficulties or suggestions to mention about this Task?

Respondent 2585357

None

Respondent 2593619

Why are the options not alphabetized?

Respondent 2594568

You have a long list and it is hard to find the specific exercise, when it looks like it is in a random. Maybe list it after first alphabet.

Respondent 2596050

no

Respondent 2597585

no

Task 6 Analysis

Task 6: 6) The last Task on the Settings screen is to choose how many exercises should be performed per muscle group. There are muscle groups listed. You are supposed to change only 2 default numbers: The exercises for the "Back" muscles need to change from 4 to 5 and the exercises for the "Biceps" muscles need to change from 2 to 3. When done, mark as "Complete".

00:00:14 AVG TIME **4** PASSED **1** FAILED **0** UNGRADED

Task 6 Questions

Question 1: Any difficulties or suggestions to this Task?

Respondent 2585357

None

Respondent 2593619

no

Respondent 2594568

No it was easy, could be fun to add a random button?

Respondent 2596050

no

Respondent 2597585

no.

Task 7 Analysis

Task 7: 7) Now you need to Press the Recommendations Menu-button in order to go to the "Recommendations" screen. Here you will see the first day of your Recommended Exercise Workout Program. It will include a simple list of 8 names of exercises, group together based on their muscle groups. When you are done with this Task, mark "Complete".

00:00:21 AVG TIME **5** PASSED **0** FAILED **0** UNGRADED

Task 8 Analysis

Task 8: 8) Now we suppose that you performed the exercises of the first day and the second day of your program. Your Task is to see the Recommendations of exercises for the third day (3rd) of your program. When you are done, mark the Task as "Complete".

00:00:11 AVG TIME **5** PASSED **0** FAILED **0** UNGRADED

Task 8 Questions

Question 1: Did you encounter any difficulties in performing this task? Any suggestions you may have?

Respondent 2585357

None

Respondent 2593619

no

Respondent 2594568

I think what I did was correct, not totally sure.

Respondent 2596050

no

Respondent 2597585

no.perfect

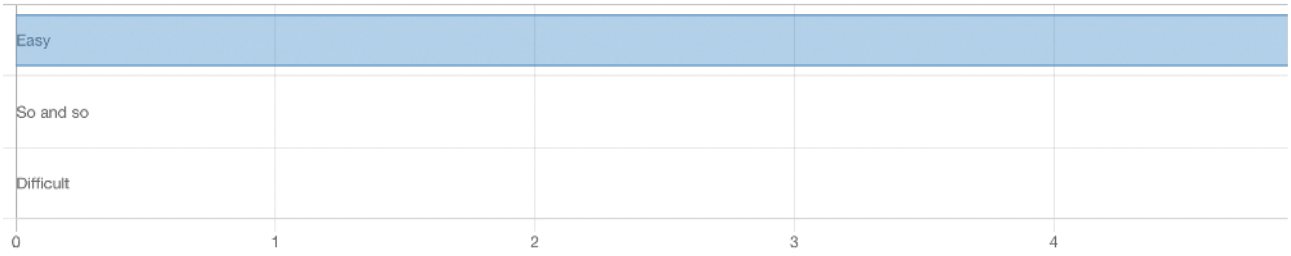
Task 9 Analysis

Task 9: 9) Now we suppose that you have finished all the 5 days of your exercise program and you are going to repeat the same program again from Day 1. Your Task is to continue getting recommendations for the next days of the program until you arrive again at the same Recommendation for Day 1. When you are done with the Task, mark as "Complete".

00:00:12 AVG TIME **4** PASSED **1** FAILED **0** UNGRADED

Task 9 Questions

Question 1: How difficult was it to perform this Task?



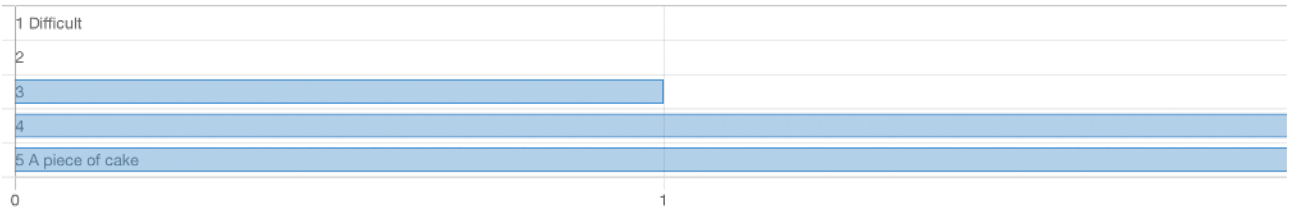
Task 10 Analysis

Task 10: 10) Now we suppose that you have performed this exercise program for several weeks and you need to get the next one Your Task is to request and get a recommendation for the next Program to do. When done, mark the Task as "Complete".

00:00:34 AVG TIME **4** PASSED & **1** FAILED ' **0** UNGRADED

Task 10 Questions

Question 1: In a scale from 1 to 5, how easy was it to perform this Task?



Task 11 Analysis

Task 11: 11) Ok. Now let's say that you have arrived to the second day of your new program and you want to see the recommendation. When you do that you realise that you are not satisfied with some exercise of the 2nd day and you want to change it. Your double Task is: a) to see the 2nd Day of Recommendations and then b) request editing of the recommendation fo one of the 8 exercises, and receive a similar recommendation (not the same) for that 2nd day of exercises. When you are done, a you have received new recommendations for that Day, press "Complete".

00:01:58 AVG TIME **3** PASSED **2** FAILED **0** UNGRADED

Task 11 Questions

Question 1: Were there any difficulties in performing that Task? Do you have any suggestions for it?

Respondent 2585357

Yes. It does not say how one changes the work out drill.

Respondent 2593619

I should have a list of alternatives instead of it just changing the exercise on its own. Even if there only exists one alternative, I should still get a list with one item in it, so that I understand that something is changing and that I have agency over that change

Respondent 2594568

If what I did was correct, I find the icon at the exercise a little confusing, since I suppose the "pencil" icon stands for editing, I was expecting a pop-up with editing options. If I just wanted to delete it, I would think trash icon or cross. Maybe if it was a toggle (toggling between exercises) button, it should look different

Respondent 2596050

There should be a button name change the exercise and then may be a comparison side by side what exercises are changed.

Respondent 2597585

Yes, I don't know how to change my recommendations.

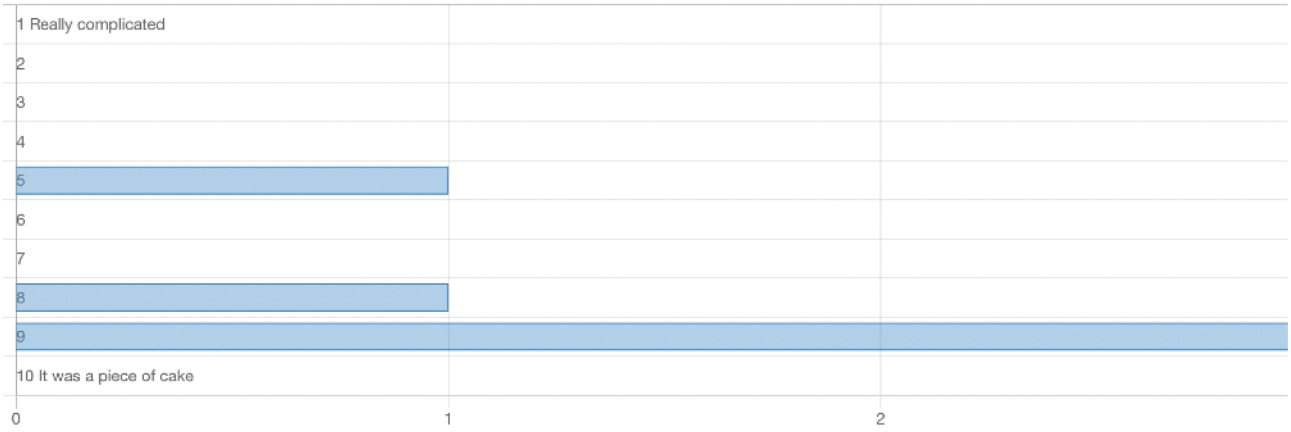
Task 12 Analysis

Task 12: 12) Finally your last Task is to Log-Out for the day and be directed back to the Log-In screen. When done, press “Complete”.

00:00:12 AVG TIME **5** PASSED **0** FAILED **0** UNGRADED

Follow up questions

Question 1: In a rate from 1 to 10, how easy was it to use the prototype?



Question 2: The prototype gave the opportunity for you to make some personalised Settings. Were they what you preferred, or would you prefer different settings - and if so which?

Respondent 2585357

They were what I preferred

Respondent 2593619

I'm not sure

Respondent 2594568

Mentioned during the test, it would be fun to instead of being really specific on doing exercises for the biceps, forearm etc. Maybe just do a low, medium, high, since i'm not picky with my workouts. During the drop-down menus, it is hard to notice that the empty space ones are the one, we you don't select any, maybe write "nothing or none selected"

Respondent 2596050

that is fine for me

Respondent 2597585

Yes, I could select them in the way preferred.

Question 3: How did you find the presentation of the Recommendations? Are there any things you would like changed on that?

Respondent 2585357

No

Respondent 2593619

Pretty barebones, but I understand that this is an early prototype, so I shouldn't speak about fidelity or the graphical elements. It is a bit weird that when I want to change a setting I'm not given a list, but instead the app changes the exercise on its own.

Respondent 2594568

Maybe the icons/bullet point, I found them weird, since when I clicked on them, I expected some editing menu, maybe change it to something which more represents the function better.

Respondent 2596050

the visual design can be more attractive

Respondent 2597585

No, they were quite good.

Question 4: Thanks a lot for your contribution! Last question: do you have any other comments or suggestions about the main features or design?

Respondent 2585357

No

Respondent 2593619

I don't

Respondent 2594568

Depending on who you are targeting, maybe including some visual on how to do the exercises, or perhaps some tips on how to make them harder.

Respondent 2596050

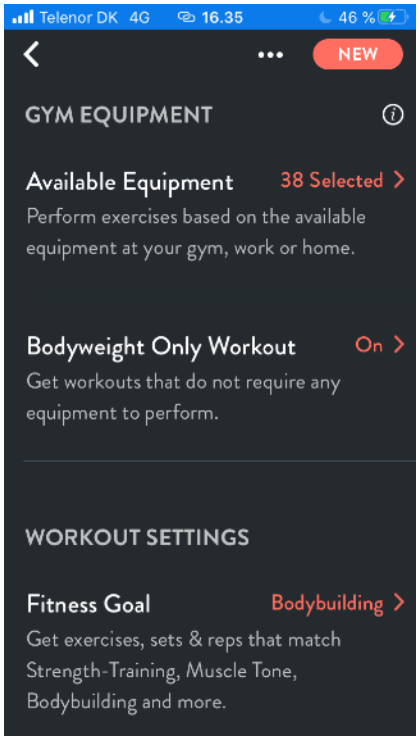
No, this is quite nice.

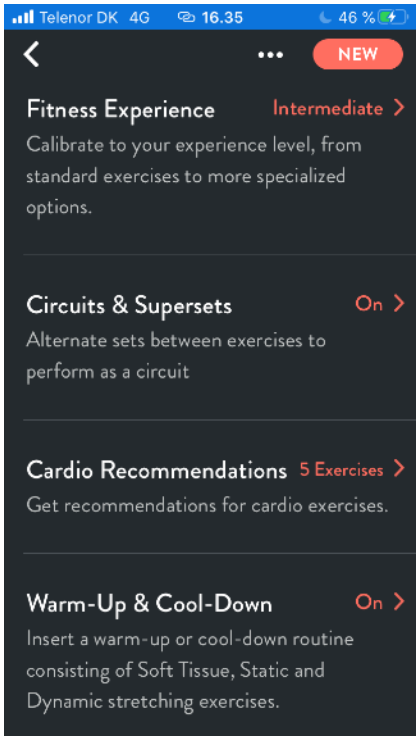
Respondent 2597585

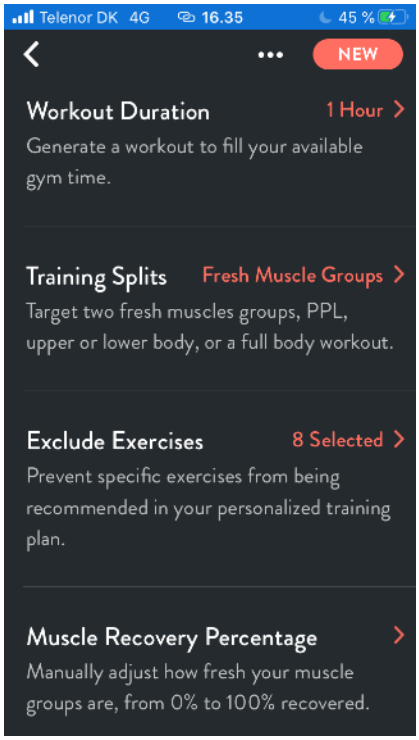
No.

D. Personalisation in the Fitbod App

These screenshots show the variety of personalisation options offered in the State-of-the-Art Fitbod app for fitness training. They include choices for available equipment, fitness goals and fitness experience, choices for stretching and cardio, workout duration, exclusion of exercises, state of muscle recovery and others…







E. Java Code for Filtering and Recommendations

This is an excerpt from our Java code which includes the algorithms for filtering exercises and also for creating automated exercise programs. The full code is included in a software folder that accompanies the report.

```
295         // F I L T E R I N G   P E R   U S E R   C O N S T R A I N T S
296
297         if (UserProfileSettings.getEquipment() != null) {  // filtering unavailable equipment
298             Log.d("Tag", "Filtering Exercises with unavailable equipment from: " +
                        PgmRecommendation.getFilteredExerciseList().size());
299             ArrayList<Exercise> temp = new ArrayList<>();
300             for (Exercise ex : PgmRecommendation.getFilteredExerciseList()) {
301                 if (UserProfileSettings.getEquipment().contains(ex.getEquipment()))
                            temp.add(ex);
302             }
303             PgmRecommendation.removeFilteredExercises(temp);
304             Log.d("Tag", "Filtered size: " +
                        PgmRecommendation.getFilteredExerciseList().size());
305         }
306
307         if (UserProfileSettings.getExcludedExerciseIDs() != null) {  // filtering excluded
308             Log.d("Tag", "Filtering Exercises that can't be done from: " +
                        PgmRecommendation.getFilteredExerciseList().size());
309             ArrayList<Exercise> temp = new ArrayList<>();
310             for (Exercise ex : PgmRecommendation.getFilteredExerciseList()) {
311                 if (UserProfileSettings.getExcludedExerciseIDs().contains(ex.getExerciseId()))
                            temp.add(ex);
312             }
313             PgmRecommendation.removeFilteredExercises(temp);
314             Log.d("Tag", "Filtered size: " +
                        PgmRecommendation.getFilteredExerciseList().size());
315         }
316
317         if (UserProfileSettings.getMaleFemale().equals("Only for Males")) { // filtering F
318             Log.d("Tag", "Filtering Exercises for Women from: " +
                        PgmRecommendation.getFilteredExerciseList().size());
319             ArrayList<Exercise> temp = new ArrayList<>();
320             for (Exercise ex : PgmRecommendation.getFilteredExerciseList()) {
321                 if (ex.getMaleFemale().equals("Only for Females")) temp.add(ex);
322             }
323             PgmRecommendation.removeFilteredExercises(temp);
324             Log.d("Tag", "Filtered size: " +
                        PgmRecommendation.getFilteredExerciseList().size());
325         }
326         if (UserProfileSettings.getMaleFemale().equals("Only for Females")) { //filtering M
327             Log.d("Tag", "Filtering Exercises for Men from: " +
                        PgmRecommendation.getFilteredExerciseList().size());
328             ArrayList<Exercise> temp = new ArrayList<>();
329             for (Exercise ex : PgmRecommendation.getFilteredExerciseList()) {
330                 if (ex.getMaleFemale().equals("Only for Males")) temp.add(ex);
331             }
332             PgmRecommendation.removeFilteredExercises(temp);
333             Log.d("Tag", "Filtered size: " +
                        PgmRecommendation.getFilteredExerciseList().size());
334         }
335
336         // MAX DAILY TRAINING
337         int maxDailyTraining = (int) ceil((double) MuscleGroups.getMaxTotalTraining()/
                    UserProfileSettings.getTrainingDays());
338         Log.d("Tag", "Max Day-Training: " + maxDailyTraining);
339
340         // S H U F F L I N G   E X E R C I S E S
341
342         Collections.shuffle(PgmRecommendation.getFilteredExerciseList());  // Randomizing list
343
344         // R E C O M M E N D A T I O N   A L G O R I T H M
345
```

```
346         Log.d("Tag", "Training Days: " + UserProfileSettings.getTrainingDays());
347         // Calculate recommendations for each Day of the Workout Program
348         for (int day=1; day <= UserProfileSettings.getTrainingDays(); day++) {
349
350             Log.d("Tag", "Day: " + day);
351             long totDayExercises = 0;    // initialise amount of recommended exercises for the
    day
352
353             while (2*totDayExercises < maxDailyTraining) {
354
355                 // Sorting the BIG muscles, by "least Total Training" and by "most MaxTraining"
356                 Collections.sort(MuscleGroups.getBigMuscleGroups(),
                                 new Comparator<MuscleGroup>() {
357                     @Override
358                     public int compare(MuscleGroup m1, MuscleGroup m2) {
359                         return ComparisonChain.start().
                             compare(m1.getTotalTraining(),m2.getTotalTraining()).
                             compare(m2.getMaxTraining(),m1.getMaxTraining()).result();
360                     }
361                 });
362                 StringBuilder bigGroups = new StringBuilder();
363                 for (int bigMuscle=0; bigMuscle < MuscleGroups.getBigMuscleGroups().size();
                         bigMuscle++)
364                     bigGroups.append(MuscleGroups.getBigMuscleGroup(bigMuscle).getName()+" "+
                         MuscleGroups.getBigMuscleGroup(bigMuscle).getTotalTraining()+" - ");
365                 Log.d("Tag", "Sorting Big Muscles: " + bigGroups.toString());
366                 Log.d("Tag", "Expected Training: " + (2*totDayExercises +
                         MuscleGroups.getBigMuscleGroup(0).getMaxTraining()));
367                 Log.d("Tag", "Max Day-Training: " + maxDailyTraining);
368                 if (MuscleGroups.getBigMuscleGroup(0).getTrainingDay() == 0)
                         seedMuscle = MuscleGroups.getBigMuscleGroup(0).getName();
369
370                 if ((2*totDayExercises + MuscleGroups.getBigMuscleGroup(0).getMaxTraining()) >
                         maxDailyTraining |
371                        MuscleGroups.getBigMuscleGroup(0).getTrainingDay() > 0) {
372
373                     // Sorting Small muscles, by "least Total Training" & by"least MaxTraining"
374                     Collections.sort(MuscleGroups.getSmallMuscleGroups(),
                             new Comparator<MuscleGroup>() {
375                         @Override
376                         public int compare(MuscleGroup m1, MuscleGroup m2) {
377                             return ComparisonChain.start().
                                 compare(m1.getTotalTraining(), m2.getTotalTraining()).
                                 compare(m1.getMaxTraining(), m2.getMaxTraining()).result();
378                         }
379                     });
380                     StringBuilder smallGroups = new StringBuilder();
381                     for (int smallMuscle=0; smallMuscle <
                             MuscleGroups.getSmallMuscleGroups().size(); smallMuscle++)
                             smallGroups.append(MuscleGroups.getSmallMuscleGroup(smallMuscle)
                             .getName()+ " "+MuscleGroups.getSmallMuscleGroup(smallMuscle)
                             .getTotalTraining()+" - ");
382                     Log.d("Tag", "Sorting Small Muscles: " + smallGroups.toString());
383                     Log.d("Tag", "Expected Training: " + (2*totDayExercises +
                             MuscleGroups.getSmallMuscleGroup(0).getMaxTraining()));
384                     Log.d("Tag", "Max Day-Training: " + maxDailyTraining);
385                     seedMuscle = MuscleGroups.getSmallMuscleGroup(0).getName();
386                     if ((2*totDayExercises + MuscleGroups.getSmallMuscleGroup(0)
                             .getMaxTraining()) > maxDailyTraining) break;
387                 }
388
389                 Log.d("Tag", "Seed Muscle: " + seedMuscle + " day: " +
                         MuscleGroups.getBigMuscleGroup(0).getTrainingDay());
390                 long totSeedMuscleExercises = 0;  // initialise recommended exercises for seed
391
392                 // Find recommendable exercises for Seed Muscle
393                 for (Exercise ex : PgmRecommendation.getFilteredExerciseList()) {
                         // Use the filtered list to find exercise
394                     if (ex.getPrimaryMuscleGroup().equals(seedMuscle)) {
                             // if exercise is primarily for the seed muscle, recommend!
395                         totDayExercises ++;  // increment total exercises
396                         totSeedMuscleExercises ++;
```

```
397                         PgmRecommendation.addPgmExercise(ex); // Add exercise to Program
398                         toBeRemoved.add(ex);
                             // Keep track of exeries to be removed from filtered list
399                         if (UserProfileSettings.getTrainingDays()<3) {
                             // Add exercise to the correct Day based on Program Type
400                             if (PgmRecommendation.getPgmRecommendation().size() % 2 != 0)
                             PgmRecommendation.addDayExercise(ex, day);
401                             else PgmRecommendation.addDayExercise(ex, day+2);
402                         }
403                         else PgmRecommendation.addDayExercise(ex, day);
404                         Log.d("Tag", "Seed muscle exercises: " + totSeedMuscleExercises
                             + " - Day exercises: " + totDayExercises);
405
406                         if (!Arrays.asList(MuscleGroups.getSmallMuscleNames())
                             .contains(seedMuscle)) {
                             // if big muscle, add up the total training from the exercise
407                             for (MuscleGroup bigMuscle: MuscleGroups.getBigMuscleGroups()) {
408                                 bigMuscle.setTotalTraining(bigMuscle.getTotalTraining() +
                                     (long) ex.getTrainingByMuscle().get(bigMuscle.getName()));
409                             }
410                             // if the seed muscle training reaches or exceeds the set maximum
                                 for the big muscle - stop recommending for that muscle
411                             if (2*totSeedMuscleExercises >= (long)
                                 UserProfileSettings.getTrainingByMuscle().get(seedMuscle)) {
412                                 MuscleGroups.getBigMuscleGroups().get(0).setTrainingDay(day);
413                                 Log.d("Tag", "Breaking big muscle: " + seedMuscle + " day: "
                                     + MuscleGroups.getBigMuscleGroups().get(0).getTrainingDay());
414                                 break;
415                             }
416                         } else {    // if small muscle, add up the training from the exercise
417                             MuscleGroups.getSmallMuscleGroups().get(0).
                                 setTotalTraining(MuscleGroups.getSmallMuscleGroups().
                                 get(0).getTotalTraining()
418                                  + (long) ex.getTrainingByMuscle().get(seedMuscle));
419                             Log.d("Tag", "Breaking small muscle");
420                             break;
421                         }
422                     }
423                 }
424                 // Remove recommended exercises from filtered exercises
425                 for (Exercise ex: toBeRemoved) PgmRecommendation.removeFilteredExercise(ex);
426                 toBeRemoved.clear();
427             }
428         }
429     }
430
```

F. Java Code for Sorting exercises by Cosine Similarity

This is an excerpt from our Java code which describes the functions used for finding similar exercises to replace recommended exercises, by using the cosine similarity metric. Full code can be found in the software folder accompanying the report.

```
208     private void findReplacements(Exercise exercise) {
209         PgmRecommendation.getReplaceRecommendation().clear();
210         PgmRecommendation.getReplaceRecommendation().add(exercise);
211         Log.d("Tag", "Potential exercises: " +
                    PgmRecommendation.getFilteredExerciseList().size());
212         for (Exercise filtered : PgmRecommendation.getFilteredExerciseList()) {
213             if (filtered.getPrimaryMuscleGroup().equals(exercise.getPrimaryMuscleGroup())) {
                      //if exercise trains mainly the original muscle, add it to the list
214                 PgmRecommendation.getReplaceRecommendation().add(filtered);
215             }
216         }
217         for (Exercise recommended : PgmRecommendation.getReplaceRecommendation()) {
                      // calculating cosine similarity for recommendations
218             recommended.setCosineSimilarity
                      (cosineSimilarity(exercise.getTrainingByMuscleVector(),
                      recommended.getTrainingByMuscleVector()));
219         } // sorting by descending cosine similarity metric
220         Collections.sort(PgmRecommendation.getReplaceRecommendation(),
                                  new Comparator<Exercise>() {
221             @Override
222             public int compare(Exercise e1, Exercise e2) {
223                 return ComparisonChain.start().compare
                              (e2.getCosineSimilarity(),
                               e1.getCosineSimilarity()).result();
224             }
225         });
226         for (Exercise recommended : PgmRecommendation.getReplaceRecommendation()) {
227             Log.d("Tag", "Vector : " + Arrays.toString(recommended.
                                          getTrainingByMuscleVector()));
228             Log.d("Tag", "Cosine Sim: " +cosineSimilarity(exercise.getTrainingByMuscleVector(),
                                          recommended.getTrainingByMuscleVector()));
229         }
230     }
231
232     private static double cosineSimilarity(long[] vectorA, long[] vectorB) {
233         double dotProduct = 0.0;
234         double normA = 0.0;
235         double normB = 0.0;
236         for (int i = 0; i < vectorA.length; i++) {
237             dotProduct += vectorA[i] * vectorB[i];
238             normA += Math.pow(vectorA[i], 2);
239             normB += Math.pow(vectorB[i], 2);
240         }
241         return dotProduct / (Math.sqrt(normA) * Math.sqrt(normB));
242     }
```