**Training Studio app**

List of authors:

Zina Kuzmin

Eliran Levy

# 

# **1.1 Introduction**

We want to build out interpretation of social training app. Currently the most common training apps supply various training sets, highlight the recommended training etc. we want to blend people and connect them via all kind of sport workouts.

## **1.1.1 Purpose of the system**

We want to create a "social network" training app which help users to work together. In the new app people will be able to create and manage training groups, invite other registers users and use it as a main communication form to have the perfect workout in and outdoors.

## **1.1.2 Scope of the system**

System scope:

The app allows the users to create a new training or join to an existing, set up the type, difficulty level and the number of participates.

Beyond the system's scope

The next items will not be included in the system’s scope: app users, Messaging system, Time

# **1.2 Current system**

The new system will bring another kind of aspect to training app. Instead of publishing training sets as an archive app, this app will be active! People will use it all the way to, during and after the training.

# 

# **1.3 Actors & Goals**

|  |  |  |  |
| --- | --- | --- | --- |
| Actor | Major | Description | Goals |
| user | ✔ | The users are the players who are create and consume services from the app. | Manage training and blend with other users. |
| Time | ✗ | The player who's controlling the timing in the system. | Manage scheduled automatic processes. |
| notification |  |  |  |

# 

# **1.4 Functional requirements**

## **1.4.1 Use case diagram**

**We need to create schemas together**

.

## **1.4.2 Use cases**

1. As app user i want to register

2. As app user i want to create a workout activity

3. As app user i want to cancel a workout activity [optional]

4. As app user i want to participate to existing workout activity

5. As app user i want to leave existing workout activity [optional]

6. As app user i want to edit my profile details [app settings - optional]

7. As group creator i want to get a notification when user participates for my workout

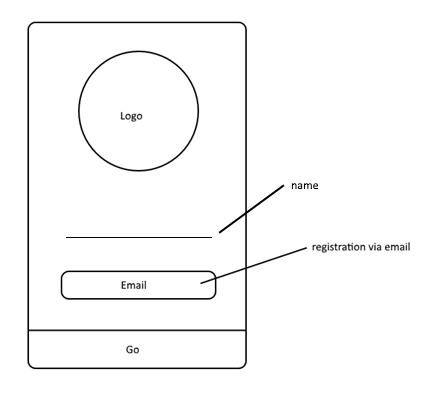
8. As group member i want to get a notification for forthcoming workout activity

9. As group member i want to get a notification for cancelled workout activity [optional , depends on use case 3]

10. As app user i want to see workout statistics [phase 2]

Registration to app

Will provide the user to register for the first time to the app



When enter name and an email address + press on go button, he will get an email to verify his app registration.

[Q]:

1. How we know to display it for the first time only?

Save to sharedPrefrences

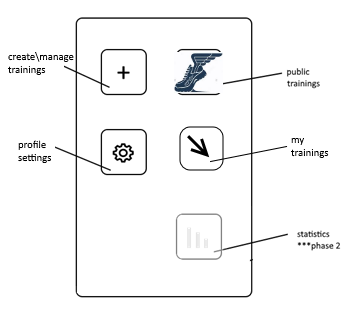
2. How we let him proceed only after registration verified via mail?

Save to sharedPrefrences

3.how we pass the unique "key" after verification? Add field for verification code.

4. What we display if registration made but not verified? Hide name + email and display code field for complete registration. (flag for go button).

Lobby



After registration is verified.

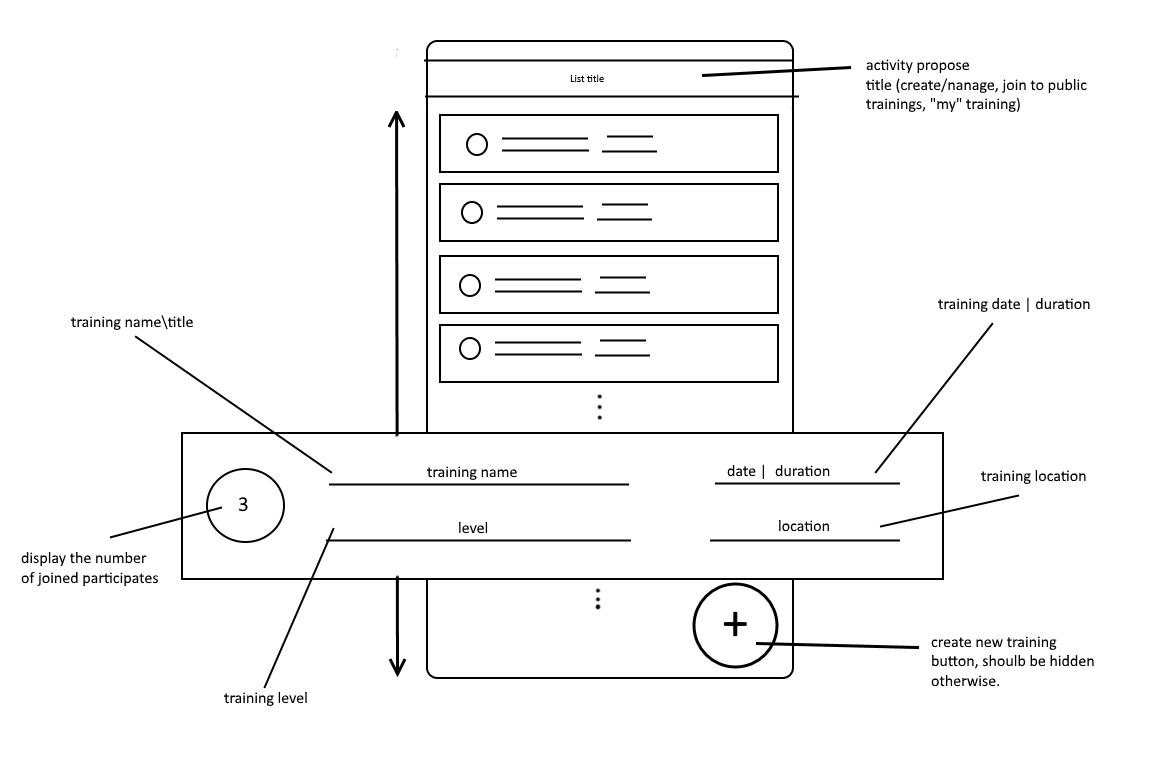
[Q]:

1. Where we want to return after this kind of result? -> toast

**Create/manage new training:**

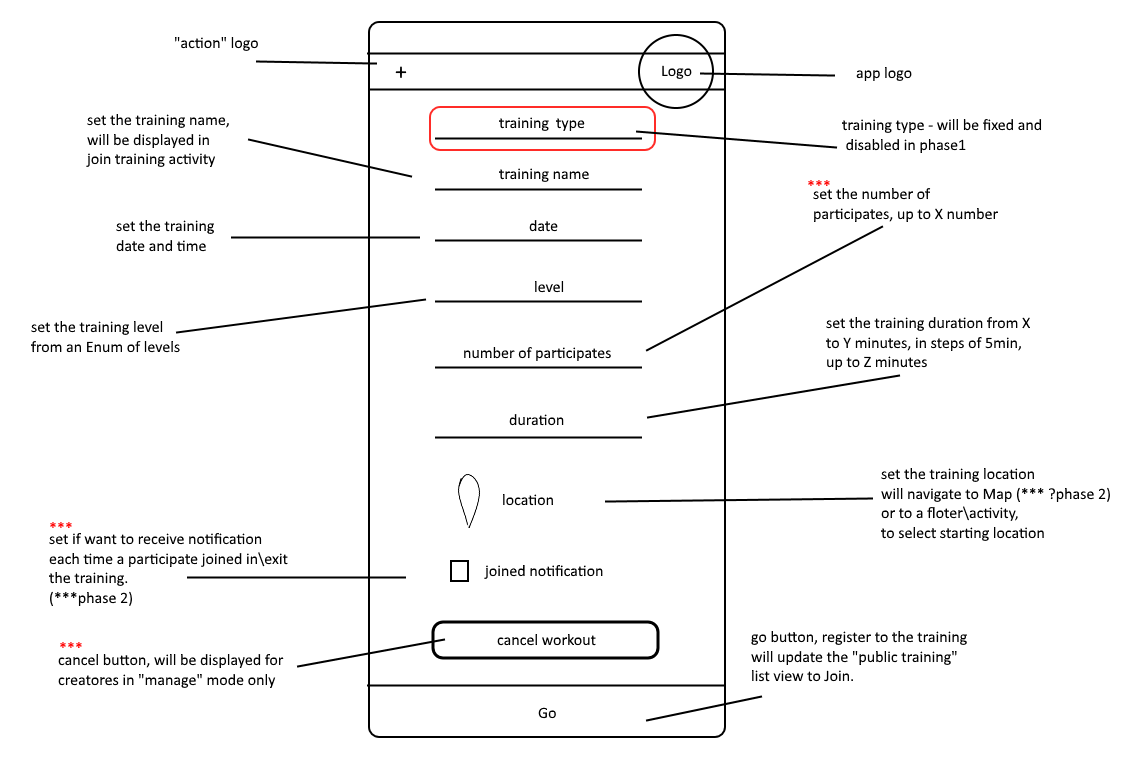
Will navigate from "create/manage" lobby button

Here the user can see his created workouts and add a new one.

****

**Create/manage:**

Will navigate from "create training" button



\*\*\* will be editable in "manage" mode. (the rest will be disabled)

[Q]:

1. Do we need to check if the creator didn’t already create another training on this date?

Check for overlapping all users

4. all actions are sync

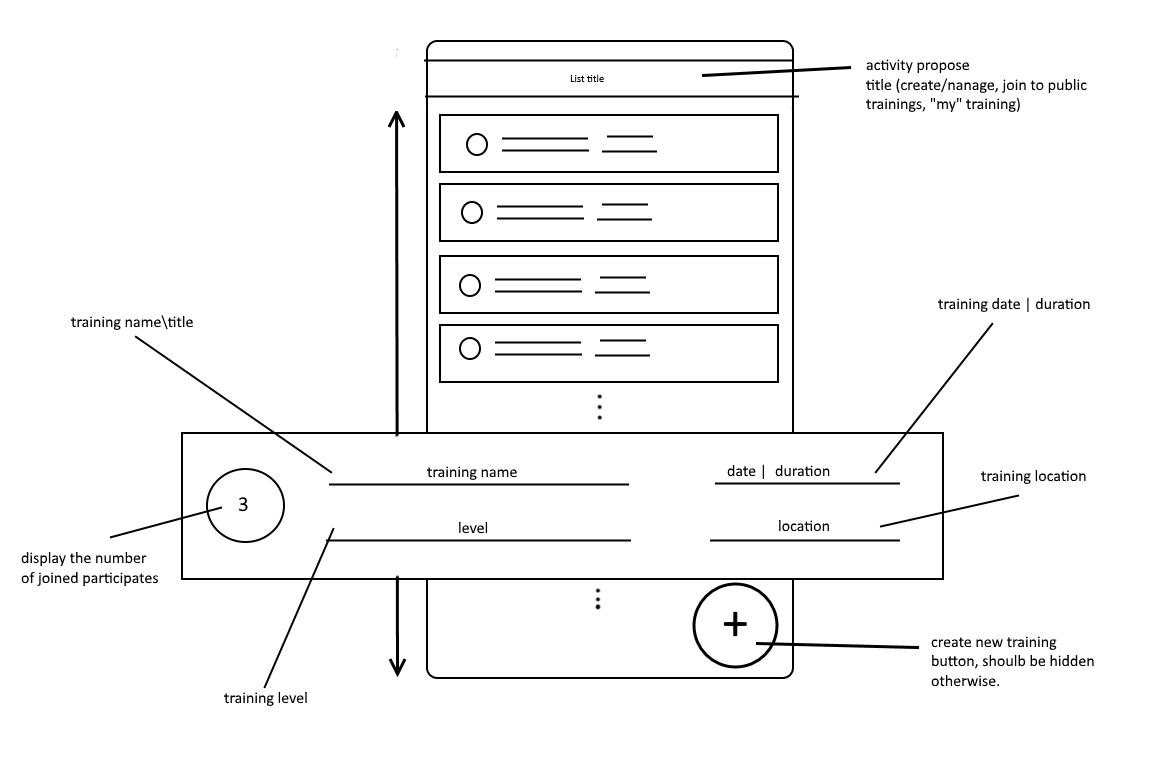
7. location will be insert with free text at first, later we will save the data in map + get/set locations – save to db in a different way?

**List of public training – select training:**

Displaying a list of all available public trainings (which the user did not joined to/created)

Or "my trainings"

\*create button will be hidden.



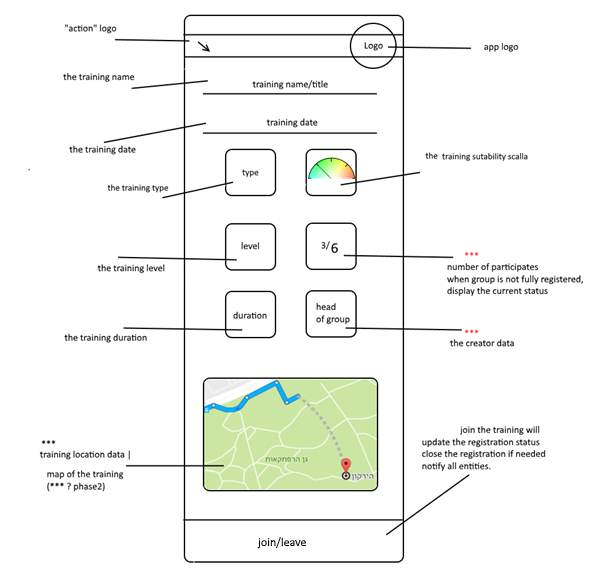
[Q]:

3. "empty state" view – display title.

4. do we want to add training type somehow [\*\*\*phase 2]

**Join to public training**:

Will navigate to "existing training" activity, displaying full data of the training and in public training will allow to join it.



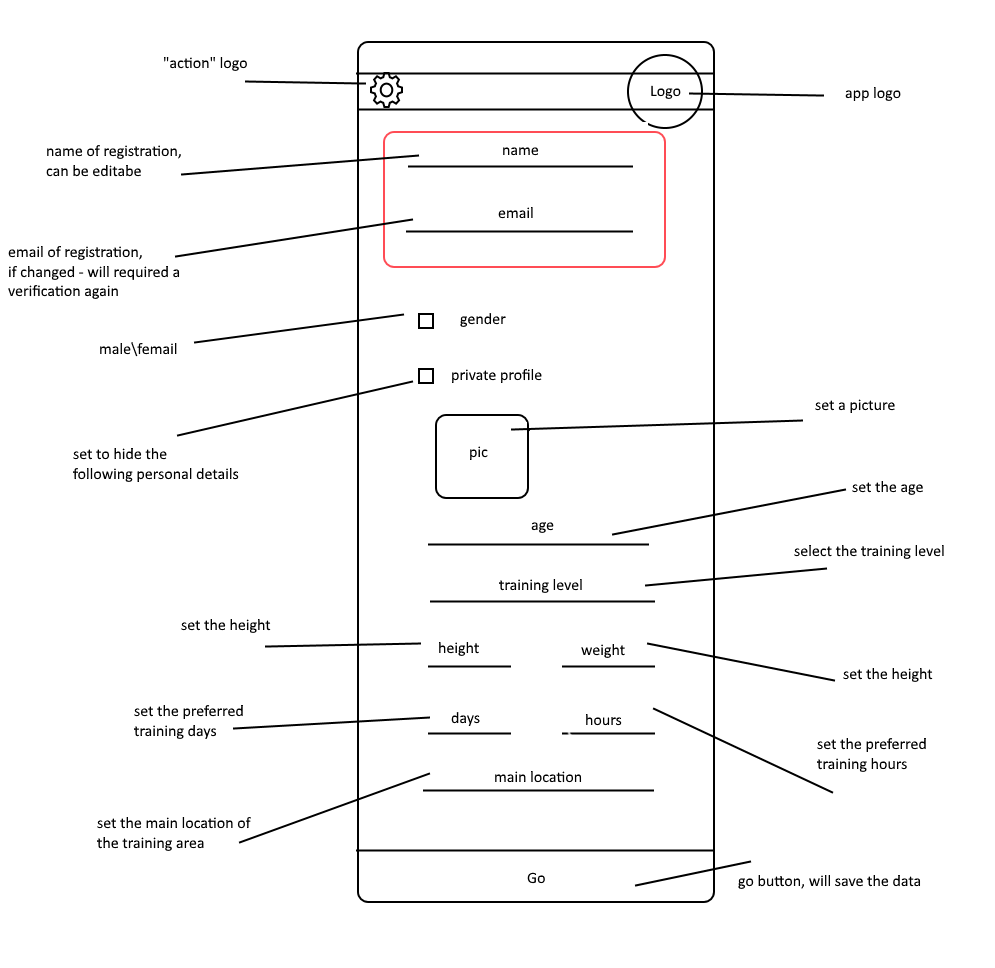
\*\*\* When click on the elements, the user will get details in floaters/activities (phase2)

[Q]:

4. What data we will display in the floaters? Check what data do we want to see as users?

**Profile settings:**

Details about me and my workout preferences.



[Q]

1. Allow picture [phase 2]

2. what about notification : creator – when join, user – when canceled, remainder x time before

**Statistics:**

This will implement on next phase.

1. My progress
2. My best training
3. View training history data (start time, end time, graph: road length, max speed, average speed, gradient data? (use device sensor)

App Dal:

Entities:

baseEntity:

id

Result

Status (enum)

baseEntity[]

Training

creatorId

Type

Name

Date

Time

Duration

Level

Max num of

Current num of

locationId

creatorNotificationFlag

IsClose

isCanceled

user:

name

email

verificationCode

isActive

gender

level

height

weight

mainLocation

picture byte[] – ask eliran

isPrivate

filter: (no need table)

from

to

filterType(enum)

userId

\*sort by date of training – closest data first

(Table) registerToNotification (set in profile activity)

trainingId

userId

notificationType

register(user) : bool

verifyRegistration(user) : user

updateUser(user) : bool

getTrainings(filter object) : trainings[]

createTraining(training object): trainingId

updateTraining(training object): bool

cancelTraining(trainingid): bool

leaveTraining(trainingid, user id): bool

joinTraining(trainingid, user id): bool

# **1.5 Non functional requirements**

Describe user-level requirements that are not related to functionality. This includes reliability, usability, performance, supportability, implementation, interface, operational, packaging, and legal requirements. Specify at least two non functional requirements for each type, that are relevant to your system.

# 

|  |  |
| --- | --- |
| Requirement name | Requirement type (U/R/P/S)[1] |
| Security - Only authorized user will allowed access his trainings in the app, this will required some kind of identification sub system | Reliability |
| Event logging - Every action is documented | Supportability |
| Notification of exceptional events | Supportability |
| The system allows to user to get all information regarding his workouts in graphic form. | Usability |
| Response Time of the system will be in less than 5 seconds | Performance |
| The system has to support 1000 concurrent operations | Performance |

# 

## **Theme:**

Text color: **333333**

Disable text color: **33ffffff**

Hint text color: **555555**

Active button, strips color: **ddFFA330**, no border

Default buttons color: **aaffffff**, border: **ffffff, width="1dp"**

Disable buttons color: **99dddddd**, no border

Icons color: FF6600

Button text size: 17dp, bold

Edit text padding left: 20dp

Hint template: "| <text>" (2 spaces)

Edit text strip: height 3dp

Edit text spacing: 5dp

Buttons spacing from edit text: 20dp

## 

## 

2 System analysis

# **2.1 Dynamic model**

Draw a sequence diagram for each interesting scenario of every use-case. There should be at least one sequence diagram for every use-case. Use the section numbers to refer the diagram to its use-case.

**Use case name:**

Sequence diagram:

# **2.2 Object model**

Draw a class diagram of the problem domain entity objects and their associations and important problem domain attributes. Make sure that all your associations have roles and multiplicities on each side.

# 

3 System design

# **3.1 Current system architecture**

Describe the architecture of the system being replaced. If there is no previous system this section can be replaced by a survey of current architectures for similar systems. The purpose of this section is to make explicit the background information that system architects used, their assumptions, and common issues the new system will address.

# **3.2 Proposed system architecture**

This section documents the new system design model. It is divided into the following subsections:

## **3.2.1 Subsystem decomposition**

This section describes the decomposition into subsystems and the responsibilities of each. This is the main product of system design. Provide a component diagram that describes the major functional subsystems (components) of the system. The diagram must show the major interfaces of each subsystem and how they are connected together.

## **3.2.2 Hardware/software mapping**

This section describes how subsystems are assigned to hardware and off-the-shelf components. It also lists the issues introduced by multiple nodes and software reuse. This section should include a deployment diagram that shows how on which physical platforms the subsystems run.

Estimate how many processors and how much memory you will need in order to satisfy the non functional requirements defined in section 1.5.

## 

## **3.2.3 Persistent data management**

This section describes the persistent data stored by the system and the data management infrastructure required for it. Explain your decision for selecting a particular persistent solution. In particular explain why the solution you have chosen fits the problem better than other options.

## **3.2.4 Access control**

This section describes the user model of the system in terms of an access matrix. Provide an access matrix for the major problem domain objects and the actors of the system.

## **3.2.5 Boundary conditions**

This section describes installation, startup, shutdown, and error behavior of the system.

**3.2.5.1 Use Case Name: Install**

Participating Actors:

Flow of Events (or basic flow activity diagram):

1.

2.

3.

Alternate flows:

# 

# **X. Glossary**

List all the important problem domain concepts that are relevant to your system. Make sure you use the language used by problem domain experts. Briefly explain each concept.

|  |  |
| --- | --- |
| Term | Description |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

[1] Usability/Reliability/Performance/Supportability