# Deliverable 1: Software Requirements Specification

Johanna Bell
Luca Bosch
Niklas Maier
Sebastian Schwarz
Simon Vogelbacher
Yasmin Hoffman

May 28, 2020

Advised by Till Niese

# Contents

# List of Figures

Part I

# Software Requirements Specification

# 1 Introduction

## 1.1 Intention of system

The goal of this project is to track the mood of individuals and groups (couples, co-workers, family members, etc.) to evaluate them from a scientific point of view. The final aim of the developed software will be to collect data about how individuals' moods are affected by the presence and moods of other individuals in a group (emotional contagion).

## 1.2 Extensiveness of system

The project consists of an android application for the users whose mood will be monitored, an interactive website for the client who will be able to access the data that the users enter, and those two systems will be connected via a web server. The functionalities of the app should be to provide an easy way for the users to track their mood and stress level, an overview and visualization of past activity and the ability to share their data with companions. The system should remind users regularly by making use of notifications and provide some basic settings like setting a profile picture or username. The application furthermore needs to collect the users data and upload it to a server.
The web application is only intended to be used by the client to download and manage the data from all participants and change settings that will then be synchronized with the mobile application (for instance the experimental cycle). Optional features, that are not necessary to successfully complete the project are additional visualization features for the website as well as the option that allows the client to change the frequency and content of questions for the individual users.

## 1.3 Goals and success criteria of project

- May 22nd, 2020 (Deliverable 1)
  Requirement Analysis

- June 12th, 2020 (Deliverable 2a)
  Design Document: Software Architecture

- July 3rd, 2020 (Deliverable 2b)
  Design Document: Detailed Design

- July 31st, 2020 (Deliverable 3)
  Implementation

The project is successfully completed if it passes the specified customer acceptance tests.

## 1.4 Definitions, acronyms and abbreviations

- Mood
  this refers to how good or bad a user is feeling. Mood will be measured on a five point scale from "very bad" to "very good".

- Companion
  a user that has sent or been sent a friendship request, that was accepted. User can see the mood changes of their companions after the end of an experimental cycle. It will be noted by the app if a companion is nearby while answering mood questions. Through this companion feature the phenomenon of emotional contagion will be explored.

- Special Situations
  Events that take place in the daily life of a person (related to work, family, health, etc.) and are associated with with a negative or positive

- Stress/Relaxation level
  The level of stress/relaxation that a user is experiencing. Measured on a five point scale from "very stressed" to "very relaxed'

- Emotional Contagion
  Happening when people's emotions and behaviours influence the emotions/behaviours of other people.

# 2 Existing System

The client has no existing system, which can be used as code base to upgrade or change for our mood tracking system, however there are a few mood tracker apps available, which the client likes and that can be used as inspiration. However none of them offer the ability to see or research how an individual's emotions are influenced by the people around them. The client stated that she liked the design and stucture of the app called 'Daylio' a lot, therefore we could base parts of our work on it, though we try to avoid copying large parts as is. The process of recording and visualizing information and the design could be good inspiration, whereas the collection of data in an experiment is not part of the functionality.

# 3 Proposed System

## 3.1 Overview

The software which has to be developed is a mood tracker in form of an android application, which is intended to capture the feelings of the users in order to evaluate them later from a scientific point of view. For this purpose, the feelings of several users are to be recorded in order to investigate the influence of the feelings of each user on the other users. In this way, insights shall be gained into how the feelings of the users are influenced by other people (and their feelings).

## 3.2 Functional Requirements

### 3.2.1 Required Functions

**/FR 10/**
Users are able to register to the system by providing a valid e-mail address and a password

**/FR 11/**
Users can login with the created account

**/FR 12/**
Users can delete their account

**/FR 13/**
Users can change their password

**/FR 14/**
Users must fill-in a "one-time consent form" when first signing into the app

**/FR 15/**
Users can send "unique user codes" to other people who have the app in order to add them as couple, friend, family, coworker, roommate, other

**/FR 16/**
Users can invite other people as couple, friend, family, coworker, roommate, other who don't have the app via e-mail

**/FR 17/**
Users have to declare their relationship (couple, friend, family, coworker, roommate, other) when adding a friend via "unique user code" or via email

**/FR 18/**
Users can optionally choose a nickname and profile picture

**/FR 19/**
Users can change their relationship with companions up to one time each

**/FR 20/**
The system should display a calendar on the start page when opened voluntarily (without

clicking notification)

**/FR 21/**
Users can click on days in the calendar to visualize the mood and stress level, companions and special situations they recorded on that day

**/FR 22/**
Users can click on a button with the calendar where they can enter their current mood, stress level, companions and special situations (which needs to have a timestamp of the recorded time)

**/FR 30/**
The app must remind the user randomly, multiple times a day to record their current mood, stress level, companions and special situations

**/FR 31/**
The app must align with the users current status for the notification settings

**/FR 40/**
Users can record mood ratings, stress/relaxtion level ratings, which companions they met and special situations they're experiencing throughout the day

**/FR 41/**
The app has to display the mood recording screen when the user opens the app following a notification

**/FR 42/**
Users have to enter a mood rating inside the mood recording screen

**/FR 43/**
The app has to display a stress/relaxation screen after the mood recording where the user has to declare a level of stress on a five-point scale

**/FR 44/**
The app has to display the companions screen after the stress / relaxation screen

**/FR 45/**
User must add relationship labels (couple, friend, family, coworker, roommate, other) in the companion screen

**/FR 46/**
The app has to display a special situations recording screen after the companions recording screen in case of the first recording of the day

**/FR 47/**
The app has to display the special situations recording screen every time when the user voluntarily records his mood

**/FR 48/**
Users can choose from a preset of special situations, adding positive or negative connotations

**/FR 50/**
The system can calculate the distance between the user and all their contacts through GPS

**/FR 60/**
The recorded mood data needs to be stored locally on the user's android device first

**/FR 61/**
The mood data needs to have a time stamp, and information about if it was recorded voluntarily or by notification

**/FR 62/**
The locally stored recorded mood data should be synchronized once a day (after the last reminder)

**/FR 70/**
The visualization for mood and stress changes within a day is accessible from the landing screen

**/FR 71/**
The visualization for mood and stress corresponds to a choosable time window

**/FR 72/**
The visualization should have a x-axis which represents time

**/FR 73/**
The visualization should have a y-axis which represents mood/stress level

**/FR 74/**
The visualization tool should allow filters that can be turned off and on

**/FR 75/**
The visualization tool should have a filter to show negative or positive events of the user

**/FR 76/**
The visualization tool should have a filter to display meetings with other people

**/FR 77/**
The visualization tool should have a filter to display mood or stress levels

**/FR 78/**
The visualization tool allows to visualize data of up to 5 companions, given the selected time window does not cover the current experiment cycle

**/FR 80/**
The app should have a help page to explain the different functionalities of the app to the user

**/FR 90/**
The administrator can log-in with given credentials to the admin website

**/FR 91/**
The administrator can change the settings which are stored in the server (including administrator password, notification timing parameters, length of experimental cycle, consent form text)

**/FR 92/**
The administrator has the option to change if and how many times a day the users get notifications

**/FR 93/**
The administrator has the option to change the experiment cycle duration in days

**/FR 94/**
The administrator has the option to change the one time consent form

**/FR 95/**
The system should store a history of the changed consent form texts

**/FR 96/**
Users have to be signed to a version of the consent form text which they accepted

**/FR 100/**
The administrator can change the settings of the firebase in the web interface

**/FR 101/**
The administrator can download recorded user data of single or all users through the admin website

**/FR 102/**
The data for download can be selected in range of dates

**/FR 103/**
The data can be downloaded all in once (all available data)

**/FR 104/**
The administrator can archive recorded user data in the web interface

**/FR 110/**
The downloaded data is contained in a single CSV file with one row per user data

**/FR 111/**
The CSV file consists of rows which don't contain user names but consent form identifiers on which the users agreed

**/FR 112/**
The CSV file consists of rows which contain questionnaire answers, relationships established (with user ID and timestamp)

**/FR 113/**
The CSV file consists of rows which contain data entered along notification and enter timestamps and companions detected by GPS.

**/FR 114/**
The CSV file row only consists of a flag "voluntary" and indicator value -1 when the user omits a notification.

### 3.2.2 Desired Functions

**/DF 10/**

Users can filter the visualization for relationships, variables of the questionnaire, personality, emotion contagion score, special situations, time of the day or a combination of several filters

**/DF 20/**

The administrator can visualize the users data in the website using different filters

**/DF 30/**

The administrator can change, edit or erase the in-app questions which are displayed to the users, to trigger which kind of information users log.

**/DF 40/**
The admin can remove questions, add new questions, and select the order in which they appear

**/DF 50/**
The administrator can change if all questions appear on every notification or just once a day or in special situations

**/DF 60/**
The administrator can edit/add/erase questionnaires that need to be answered by the users upon the first log-in to the app

**/DF 61/**
The admin can select if the user has to sign the questionnaires at sign up or at a specific point in time

**/DF 70/**
The data transmission from the apps to the server is encrypted

## 3.3 Non-functional Requirements

### 3.3.1 Reliability

- The notifications and information logging should be extremely reliable and robust while the user is offline, without battery, etc.

### 3.3.2 Usability

- The information logging and visualization tool should be intuitive and easy to use for the user
- The application should be self explanatory and accessible to everyone
- The application should provide the user with a visually pleasing experience
- The application should provide a quick and intuitive navigation to avoid annoyance

- The user should personally benefit from using the app in order to get motivated to track his moods more regularly
- The user should be able to gain knowledge from the provided visualizations

### 3.3.3 Efficiency

- The app should run smoothly and quickly, to avoid burdening the users

### 3.3.4 Performance

- The system should support at least 100 users at the same time

## 3.4 System models

### 3.4.1 Scenarios

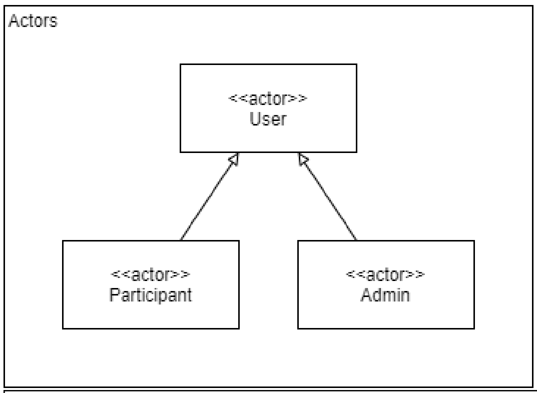
#### 3.4.1.1 General sample scenario

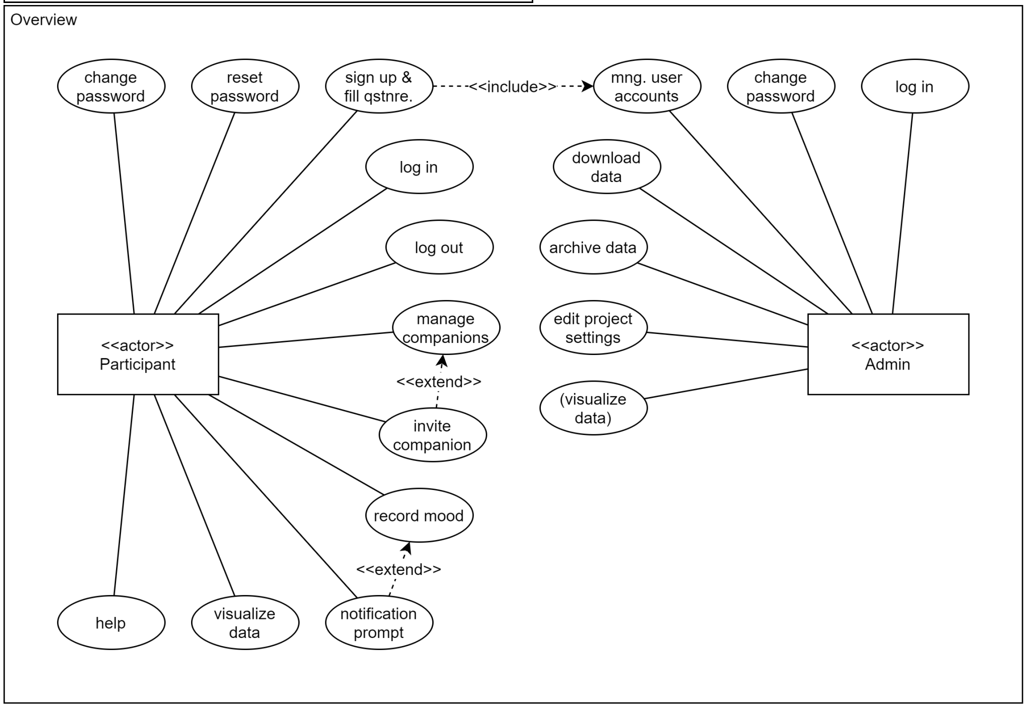
Actors: Admin, 3 participants (friends)

Events:
Participant A downloads the app and creates an account.
Participant A then invites participants B and C via Email through the app, who download it and create an account each.
Participant A shares their unique user code with both B and C, who each add A as a friend to their companions.
Over a couple of days, all participants fill the mood prompt whenever they receive notifications.
After two weeks, the admin downloads the recorded data from the server.
The admin analyses the gathered data to examine how the participants' moods develop depending on another.

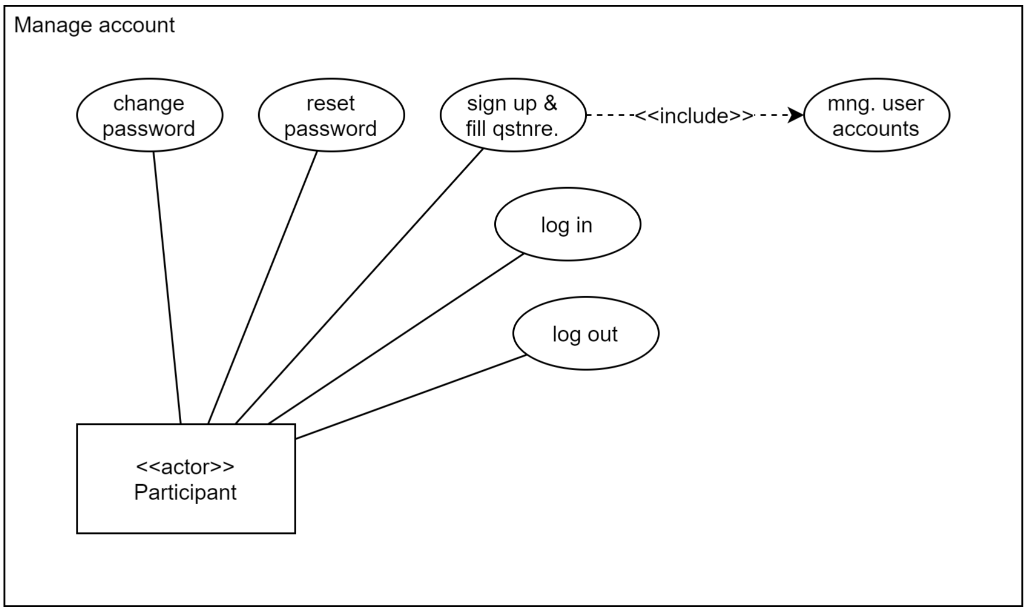
## 3.4.2 Use cases

### 3.4.2.1 Use cases overview





#### 3.4.2.2 Participant account management



| | |
|---|---|
| **Use case name** | Sign up |
| **Actors** | Participant |
| **Starting condition** | The app is installed on the participant's phone and the participant has yet to create an account. |
| **Quality requirements** | The interface is easy to understand and use.<br>The operations run seamlessly and quickly. |
| **Events** | 1. The participant opens the app for the first time and is prompted to log in or create an account.<br>2. The participant chooses to create an account and is presented a consent form.<br>3. Upon accepting, the participant is prompted for account details.<br>4. Upon filling in, the participant is asked to set an optional nickname and profile picture.<br>5. Upon completion, the participant is presented questionnaires they have to fill in.<br>6. Upon finishing, the app synchronizes data with the server and the participant is logged in. |
| **Completion condition** | The registration is successful and the participant is now logged in. |

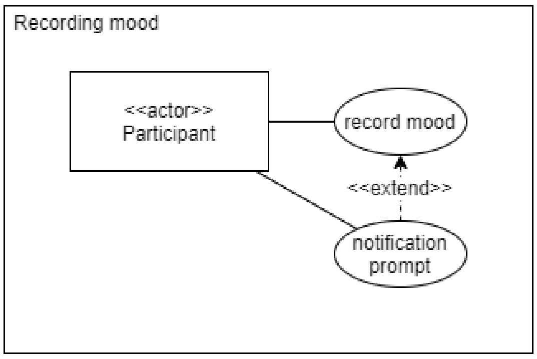
| **Use case name** | Log in |
|---|---|
| **Actors** | Participant |
| **Starting condition** | The participant has an account but is not logged in.<br>The app is installed on their phone. |
| **Quality requirements** | The interface is easy to understand and use.<br>The operations run seamlessly and quickly. |
| **Events** | 1. The participant opens the app and is prompted to log in or create an account.<br>2. The participant chooses to log in and is presented a log in form.<br>3. Upon filling in, the app compares provided data with server data.<br>4a. If data is correct, the participant is now logged in.<br>4b. If data is incorrect, the participant is told so and they have to repeat the process. |
| **Completion condition** | Logging in is successful. |

| **Use case name** | Reset password |
|---|---|
| **Actors** | Participant |
| **Starting condition** | The participant has an account but is not logged in.<br>The app is installed on their phone. |
| **Quality requirements** | The interface is easy to understand and use.<br>The operations run seamlessly and quickly. |
| **Events** | 1. The participant opens the app and is prompted to log in or create an account.<br>2. The participant chooses to log in and is presented a log in form.<br>3. The participant uses the "reset password" option on the login screen and is prompted to enter their email address.<br>4. Upon entering, the app requests the server to send a password reset email to the entered address, given it's registered. |
| **Completion condition** | The provided email address is valid and a reset link is sent. |

| **Use case name** | Log out |
|---|---|
| **Actors** | Participant |
| **Starting condition** | The participant has an account and is logged in. |
| **Quality requirements** | The interface is easy to understand and use.<br>The operations run seamlessly and quickly. |
| **Events** | 1. The participant clicks the log out button inside the app.<br>2. The app removes the saved account data and the participant is logged out. |
| **Completion condition** | The participant has logged out successfully. |

| **Use case name** | Change password |
|---|---|
| **Actors** | Participant |
| **Starting condition** | The participant has an account and is logged in. |
| **Quality requirements** | The interface is easy to understand and use.<br>The operations run seamlessly and quickly. |
| **Events** | 1. The participant clicks the change password button inside the app.<br>2. The participant is presented a password changing form and fills in the fields.<br>3a. If the current password is entered correctly, the password changes are reflected on the server.<br>3b. If the current password is incorrect, the participant is told so and has to start over. |
| **Completion condition** | The password is changed and the new password is saved on the server. |

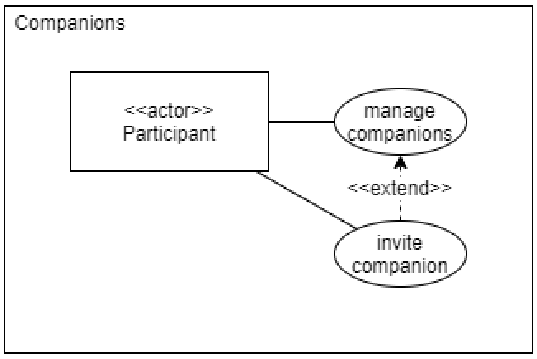
#### 3.4.2.3 Recording mood



| | |
|---|---|
| **Use case name** | Record mood |
| **Actors** | Participant |
| **Starting condition** | The participant has an account and is logged in. |
| **Quality requirements** | The interface is easy to understand and use.<br>The operations run seamlessly and quickly. |
| **Events** | 1. The participant clicks the record mood button either inside the app or on a notification.<br>2. The participant is presented a flow of questions:<br>2.1. The participant is asked to enter their mood on a scale of 1-5.<br>2.2. The participant is asked to enter their relaxation on a scale of 1-5.<br>2.3 The participant is asked to select all near companions.<br>2.4. The participant is asked to select any special situations and state their positivity.<br>3. The app fires a GPS location collection for the participant and all their companions.<br>4. The app saves the recorded data locally with timestamp and voluntarity flag. |
| **Completion condition** | The recorded data is saved on the device. |

| **Use case name** | Notification prompt |
|---|---|
| **Actors** | Participant |
| **Starting condition** | The participant has an account and is logged in.<br>The app is installed on their phone. |
| **Quality requirements** | The notification is well visible. |
| **Events** | 1. The participant receives a notification prompting them to record their mood.<br>2a.1. The participant accepts the prompt and is taken to the *Record mood* case.<br>2a.2. If this is not the first notification of the day, the situations screen in the Record mood use case (2.4.) is skipped.<br>2b. The participant ignores the notification. The app saves an entry of a missed notification with timestamp. |
| **Completion condition** | The recorded data is saved on the device. |

#### 3.4.2.4 Companions



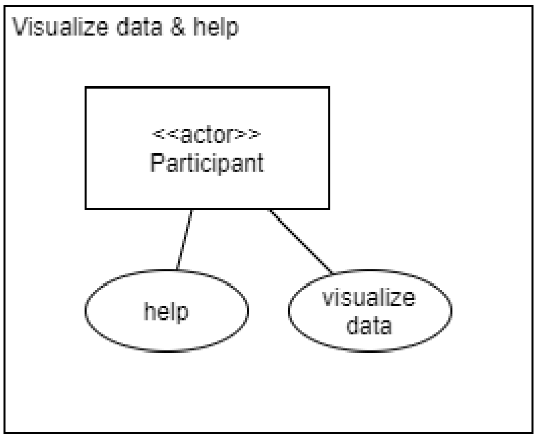
| | |
|---|---|
| **Use case name** | Adding companion |
| **Actors** | Participant |
| **Starting condition** | The participant has an account and is logged in.<br>The participant knows the unique user code of their companion. |
| **Quality requirements** | The interface is easy to understand and use.<br>The operations run seamlessly and quickly. |
| **Events** | 1. The participant clicks the add companion button inside the app.<br>2. The participant is prompted for unique user code and relationship.<br>3. Upon filling in, the companion is added to their list and the data is saved on the server. |
| **Completion condition** | The companion data is saved on the server. |

| | |
|---|---|
| **Use case name** | Inviting companion |
| **Actors** | Participant |
| **Starting condition** | The participant has an account and is logged in. |
| **Quality requirements** | The interface is easy to understand and use.<br>The operations run seamlessly and quickly. |
| **Events** | 1. The participant clicks the invite companion button inside the app.<br>2. The participant is asked for the relationship to the invited person.<br>3. Upon selecting, the participant has the option to send a message with generated links via email. |
| **Completion condition** | The email is sent. |

| | |
|---|---|
| **Use case name** | Changing relationship |
| **Actors** | Participant |
| **Starting condition** | The participant has an account and is logged in.<br>The participant has not changed the relationship with specified companion before. This is only possible once. |
| **Quality requirements** | The interface is easy to understand and use.<br>The operations run seamlessly and quickly. |
| **Events** | 1. The participant clicks the change relationship button on a companion inside the app.<br>2. The participant is prompted for the new relationship status.<br>3. Upon selecting, the changed data is saved on the server. |
| **Completion condition** | The changed data is saved on the server. |

| | |
|---|---|
| **Use case name** | Removing companion |
| **Actors** | Participant |
| **Starting condition** | The participant has an account and is logged in. |
| **Quality requirements** | The interface is easy to understand and use.<br>The operations run seamlessly and quickly. |
| **Events** | 1. The participant clicks the remove companion button on a companion inside the app.<br>2. The changed data is saved on the server. |
| **Completion condition** | The changed data is saved on the server. |

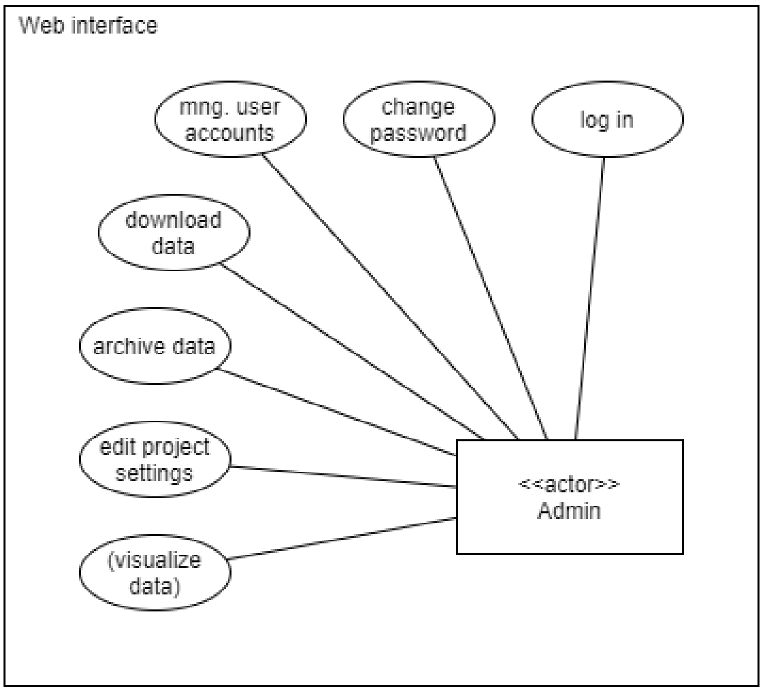
#### 3.4.2.5 Visualize data and help



| **Use case name** | Visualize data |
|---|---|
| **Actors** | Participant |
| **Starting condition** | The participant has an account and is logged in. |
| **Quality requirements** | The interface is easy to understand and use.<br>The visualization is rich and clear. |
| **Events** | 1. The participant clicks the visualize data button inside the app.<br>2. A visualization graph is shown, containing information about mood, relaxation, companions and situations.<br>3.1. The participant can choose a time frame to visualize.<br>3.2. The participant can toggle which information to display.<br>3.3. The participant can pick up to 5 companions whose mood and relaxation will be included in the visualization. Companion data can only be visualized for completed experiment cycles. |
| **Completion condition** | The graph is displayed properly. |

| **Use case name** | Help page |
|---|---|
| **Actors** | Participant |
| **Starting condition** | The participant has an account and is logged in. |
| **Quality requirements** | The interface is easy to understand and use.<br>The help page is informative. |
| **Events** | 1. The participant clicks the help button inside the app.<br>2. A help page containing information about the app and its usage is presented. |
| **Completion condition** | The help page is displayed properly. |

#### 3.4.2.6 Web interface



| **Use case name** | Admin log in |
|---|---|
| **Actors** | Admin |
| **Starting condition** | The admin has a functioning web browser and knows the address of the web interface. |
| **Quality requirements** | The interface is easy to understand and use.<br>The operations run seamlessly and quickly. |
| **Events** | 1. The admin loads the web interface and is presented a log in form.<br>2. Upon filling in, the interface compares provided data with server data.<br>3a. If data is correct, the admin is now logged in.<br>3b. If data is incorrect, the admin is told so and they have to repeat the process. |
| **Completion condition** | Logging in is successful. |

| **Use case name** | Change admin password |
|---|---|
| **Actors** | Admin |
| **Starting condition** | The admin is logged in to the web interface. |
| **Quality requirements** | The interface is easy to understand and use.<br>The operations run seamlessly and quickly. |
| **Events** | 1. The admin clicks the change password button on the web interface.<br>2. The admin is presented a password changing form and fills in the fields.<br>3a. If the current password is entered correctly, the password changes are reflected on the server.<br>3b. If the current password is incorrect, the admin is told so and has to start over. |
| **Completion condition** | The password is changed and the new password is saved onthe server. |

| **Use case name** | Project settings |
|---|---|
| **Actors** | Admin |
| **Starting condition** | The admin is logged in to the web interface. |
| **Quality requirements** | The interface is easy to understand and use.<br>The operations run seamlessly and quickly. |
| **Events** | 1. The admin clicks the edit project settings button on the web interface.<br>2. The admin can edit notification frequency and synchrony, length of experimental cycle and consent form text.<br>3. Changes are saved on the server. |
| **Completion condition** | Changes are saved on the server. |

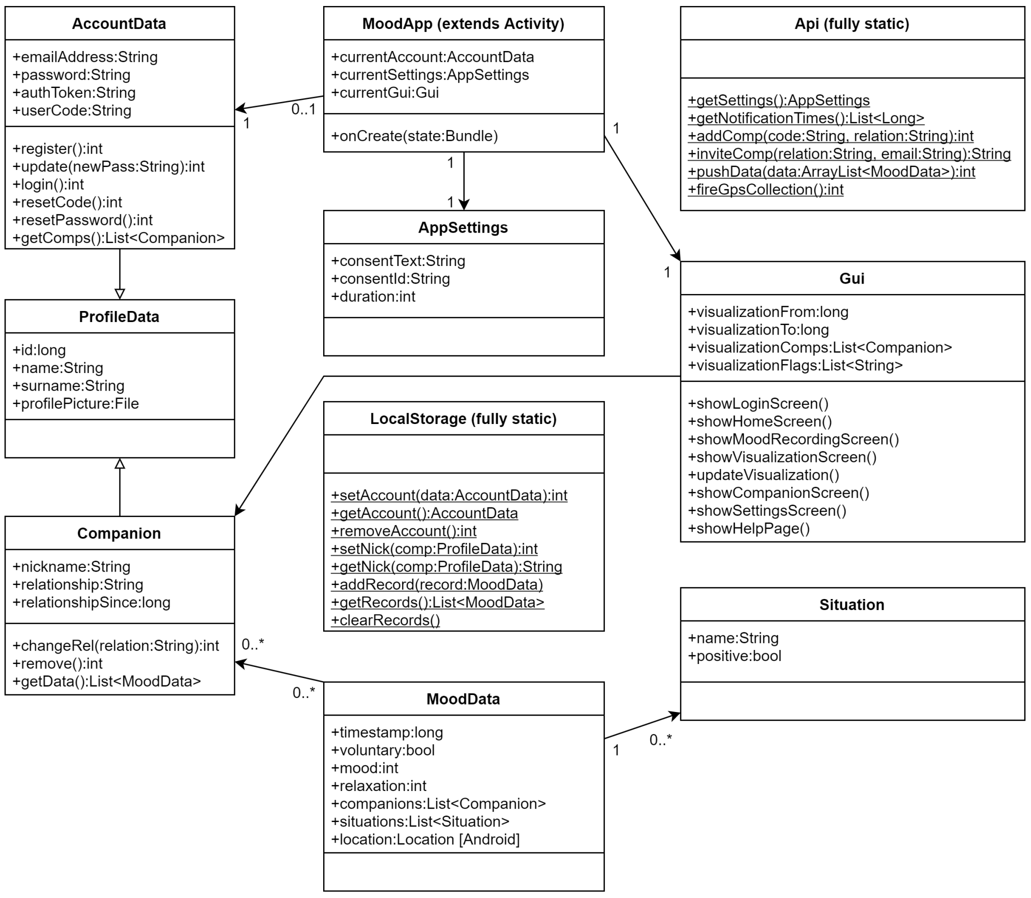
| **Use case name** | Delete user account |
|---|---|
| **Actors** | Admin |
| **Starting condition** | The admin is logged in to the web interface. |
| **Quality requirements** | The interface is easy to understand and use.<br>The operations run seamlessly and quickly. |
| **Events** | 1. The admin received a request to delete a user account.<br>2. The admin chooses to remove a user account from within the web interface.<br>3. The admin is prompted for either the user id or email address of the account to be deleted.<br>4a. An account corresponding to the entered value is found and deleted.<br>4b. No account is found for the entered value. The admin is told so. |
| **Completion condition** | All account data for the specified account is deleted from the server. |

| | |
|---|---|
| **Use case name** | Download data |
| **Actors** | Admin |
| **Starting condition** | The admin is logged in to the web interface. |
| **Quality requirements** | The interface is easy to understand and use.<br>The operations run seamlessly and quickly. |
| **Events** | 1. The admin clicks the download data button on the web interface.<br>2. The admin can specify a time window for data to be included.<br>3. A single CSV file containing all data for the specified time window with one line per user is generated and a download is prompted. |
| **Completion condition** | A CSV file is generated and downloaded. |

| | |
|---|---|
| **Use case name** | Archive data |
| **Actors** | Admin |
| **Starting condition** | The admin is logged in to the web interface. |
| **Quality requirements** | The interface is easy to understand and use.<br>The operations run seamlessly and quickly. |
| **Events** | 1. The admin clicks the archive data button on the web interface.<br>2. The admin can specify a time window for data to be archived.<br>3. The specified time window is marked as archived on the server.<br>4. This change is visible in the web interface in the future. |
| **Completion condition** | The changes are saved on the server. |

| | |
|---|---|
| **Use case name** | Web visualize data |
| **Actors** | Admin |
| **Starting condition** | The admin is logged in to the web interface. |
| **Quality requirements** | The interface is easy to understand and use.<br>The visualization is rich and clear. |
| **Events** | 1. The admin clicks the visualize data button on the web interface.<br>2. The admin can choose a time frame to visualize.<br>3. The admin can pick a user whose information to visualize.<br>4. A visualization graph is shown, containing information about mood, relaxation, companions and situations. |
| **Completion condition** | The graph is displayed properly. |

### 3.4.3 Class model


AccountData
+emailAddress:String
+password:String
+authToken:String
+userCode:String
+register():int
+update(newPass:String):int
+login():int
+resetCode():int
+resetPassword():int
+getComps():List<Companion>
MoodApp (extends Activity)
+currentAccount:AccountData
+currentSettings:AppSettings
+currentGui:Gui
+onCreate(state:Bundle)
Api (fully static)
+getSettings():AppSettings
+getNotificationTimes():List<Long>
+addComp(code:String, relation:String):int
+inviteComp(relation:String, email:String):String
+pushData(data:ArrayList<MoodData>):int
+fireGpsCollection():int
AppSettings
+consentText:String
+consentId:String
+duration:int
Gui
+visualizationFrom:long
+visualizationTo:long
+visualizationComps:List<Companion>
+visualizationFlags:List<String>
+showLoginScreen()
+showHomeScreen()
+showMoodRecordingScreen()
+showVisualizationScreen()
+updateVisualization()
+showCompanionScreen()
+showSettingsScreen()
+showHelpPage()
ProfileData
+id:long
+name:String
+surname:String
+profilePicture:File
LocalStorage (fully static)
+setAccount(data:AccountData):int
+getAccount():AccountData
+removeAccount():int
+setNick(comp:ProfileData):int
+getNick(comp:ProfileData):String
+addRecord(record:MoodData)
+getRecords():List<MoodData>
+clearRecords()
Companion
+nickname:String
+relationship:String
+relationshipSince:long
+changeRel(relation:String):int
+remove():int
+getData():List<MoodData>
Situation
+name:String
+positive:bool
MoodData
+timestamp:long
+voluntary:bool
+mood:int
+relaxation:int
+companions:List<Companion>
+situations:List<Situation>
+location:Location [Android]
0..1
1
1
1
1
1
0..*
0..*
1
0..*

### 3.4.4 UI prototypes

Figure 1: UI prototype 1

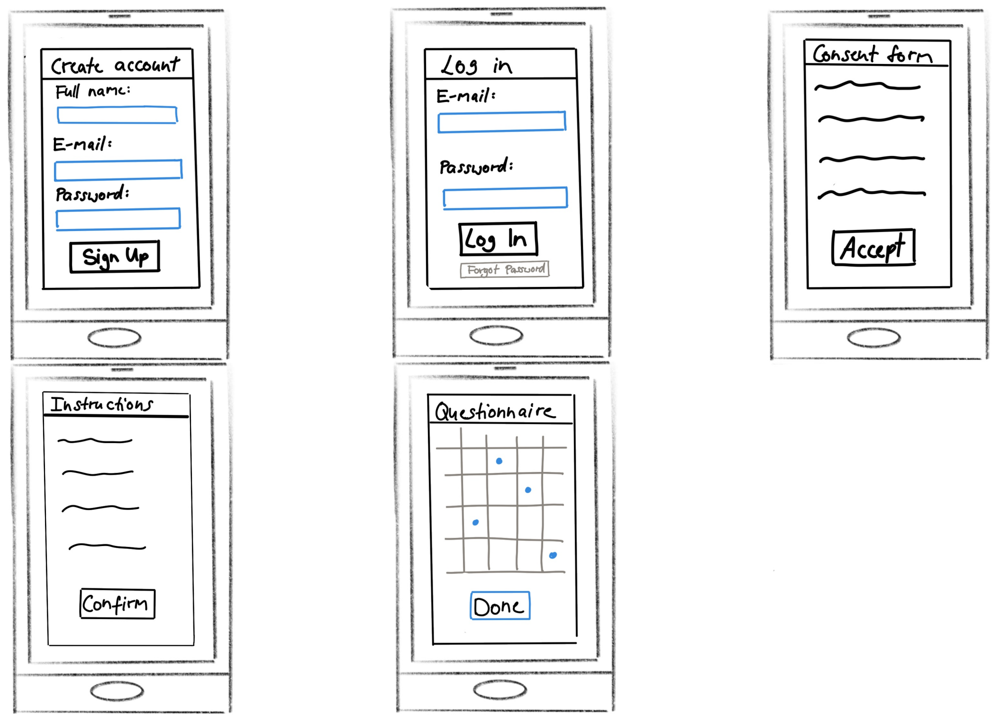


Figure 2: UI prototype 2

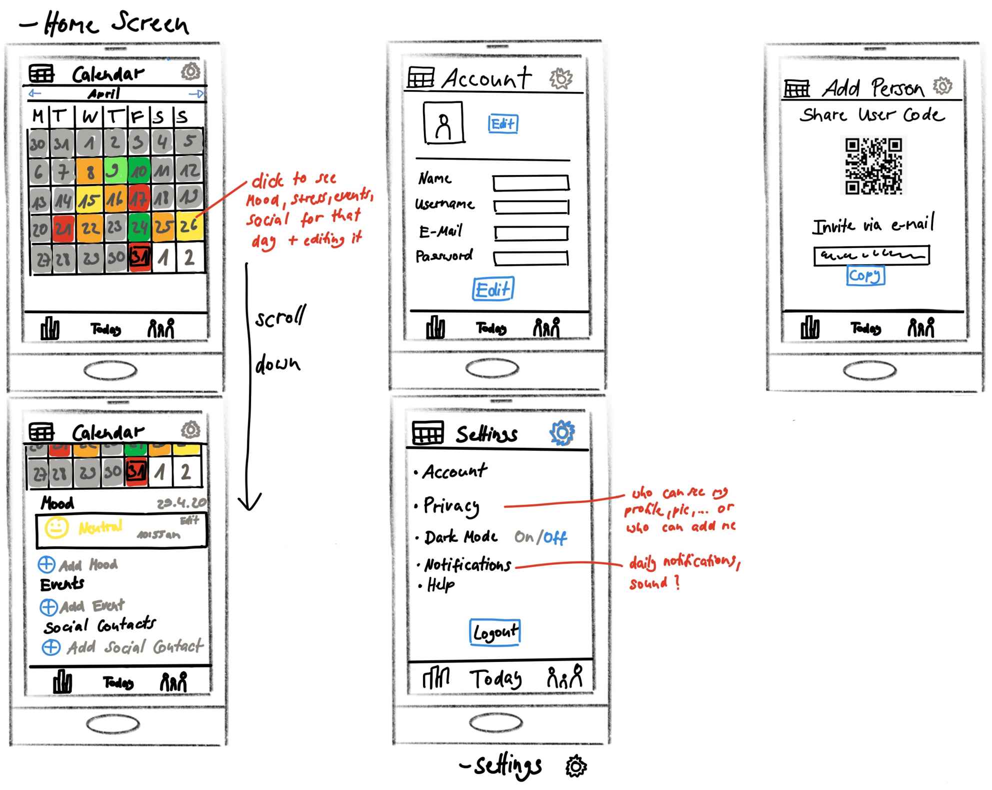


Figure 3: UI prototype 3

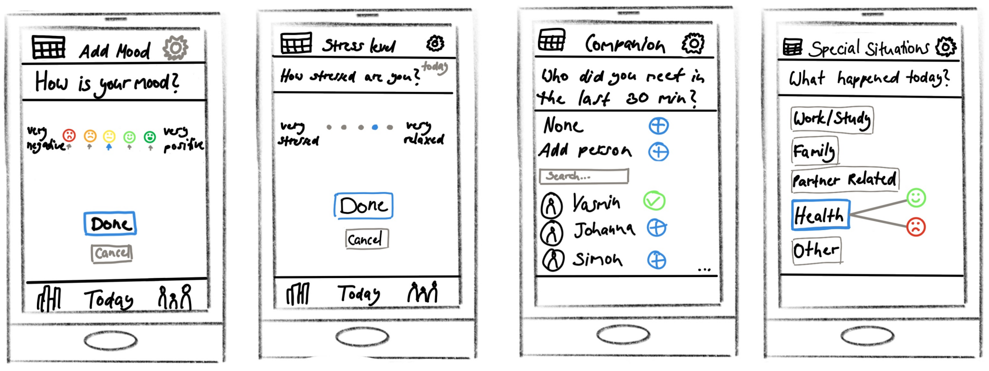
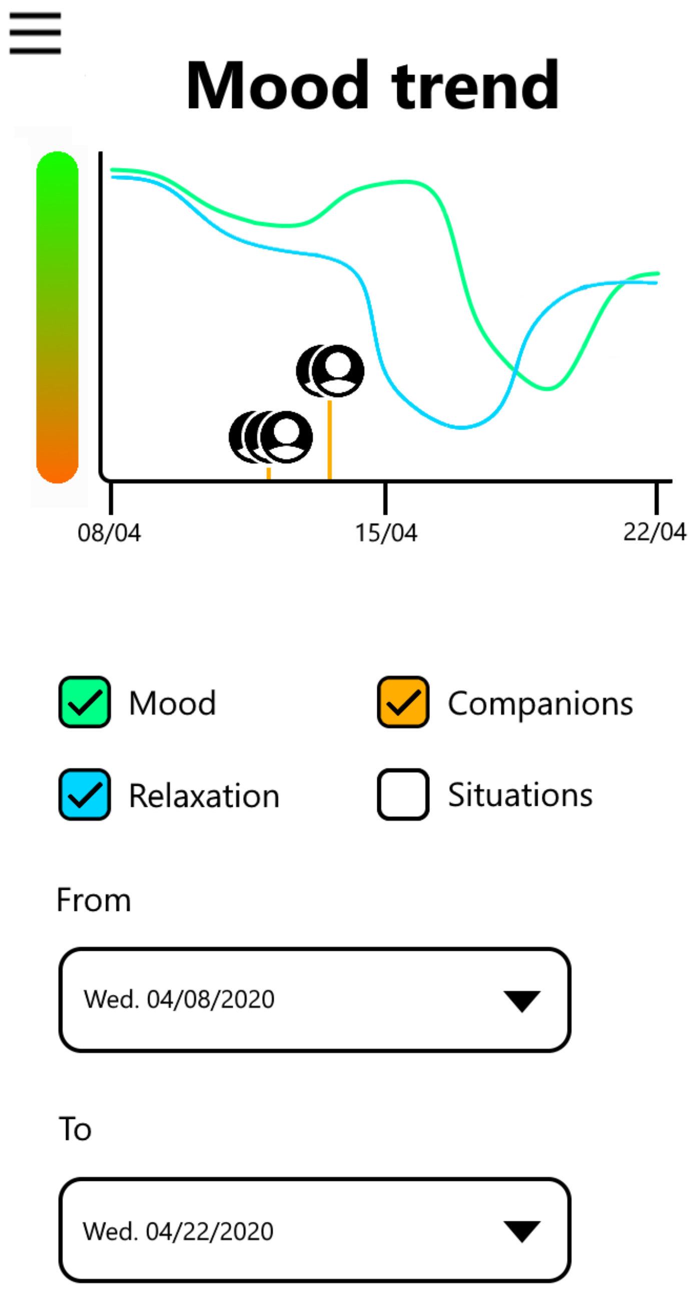


Figure 4: UI prototype 4



Mood trend

✓ Mood
✓ Companions
✓ Relaxation
☐ Situations

From

Wed. 04/08/2020 ▼

To

Wed. 04/22/2020 ▼

Figure 5: Web UI prototype 1

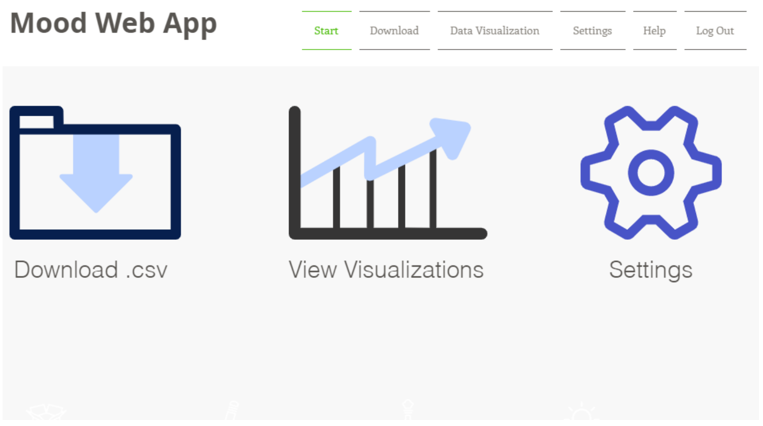


Figure 6: Web UI prototype 2

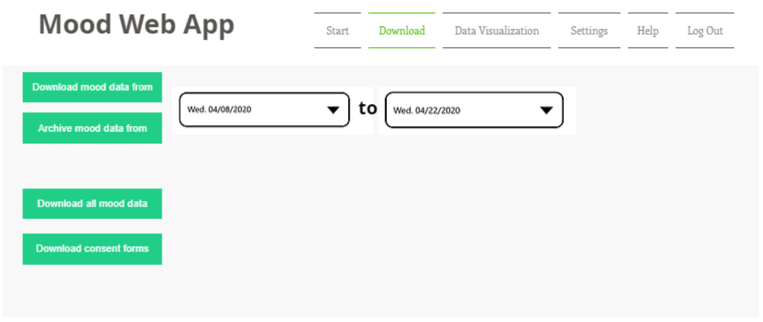
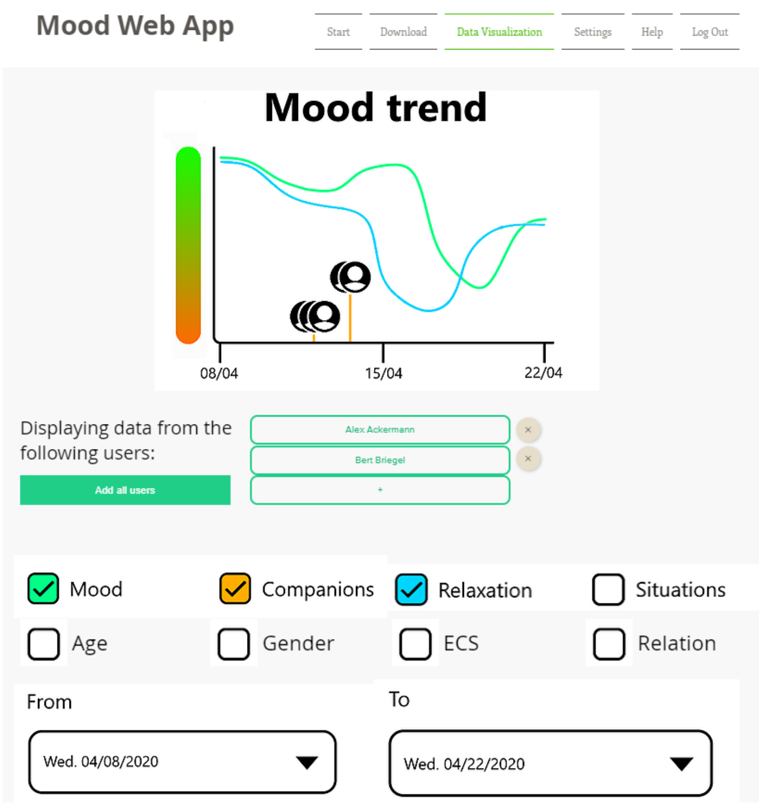


Figure 7: Web UI prototype 3



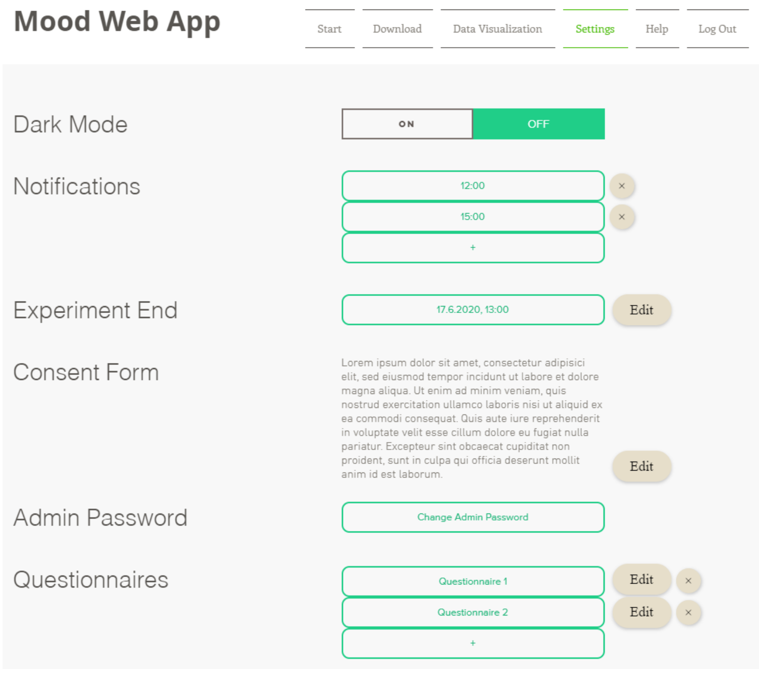
Displaying data from the following users:

Alex Ackermann ×
Bert Briegel ×

[✓ Mood] [✓ Companions] [✓ Relaxation] Situations
Age Gender ECS Relation

From: Wed. 04/08/2020
To: Wed. 04/22/2020

Figure 8: Web UI prototype 4



Dark Mode: ON [✓ OFF]

Notifications: 12:00 ×, 15:00 ×

Experiment End: 17.6.2020, 13:00

Consent Form: Lorem ipsum dolor sit amet, consectetur adipisici elit, sed eiusmod tempor incidunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquid ex ea commodi consequat. Quis aute iure reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint obcaecat cupiditat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Admin Password

Questionnaires: Questionnaire 1 ×, Questionnaire 2 ×

# 4 Glossary

**Admin/Administrator**
The person who will run the experiment and is able to access the data generated by the application and change its settings.

**Intuitive**
By saying the application should be intuitive, we mean that the user should be able to understand how to use all of its functionalities without having to read an instruction manual beforehand.

**Mood data**
Mood data refers to the data input by the user, including mood level, relaxation level, near companions and special situations.

**GPS**
*Global Positioning System* is used to determine the users position.

**one-time consent form**
The one time constent form can be edited by the admin and has to be accepted once by every users before he can start use the app.

**CSV**
*Comma-Separated Values.* The format of the table which the admin can download in which the data will be stored in.

**Experiment cycle**
After establishing a relation with a companion, the *experiment cycle* specifies the time frame which has to elapse before the corresponsive participants can view each other's data for said time frame.

**Participant**
With participant, we refer to the people who will be partaking in the experiment, the users of our application.

**qstnre.**
*Questionnaire*

**mng.**
*Manage*

**Logged in/out**
A user can only log in with his credentials, this assures that only people that are supposed to participate in the experiment are able to do so. A user can log out at any time, but will not be able to use the application.

**Voluntarity flag**
Indicates whether the user entered the mood afte receiving a notification prompting them to record their mood or on their own.

**User ID**
Every user will have a unique *User ID*, usually a string of letters and numbers which allows

a program to identify him easily, even if there are multiple users with the same names, etc.

# Part II

# Tests

# 5 Client acceptance tests

## 5.1 Introduction

For the execution of the test we will use 2 android phones (AP1, AP2) and one laptop (L1). AP1 and AP2 will be used to test the functions of the users in the application and L1 will be used to test the web interface.

### 5.1.1 Hardware and Test-Software

The different hardware on which we will run the tests is capable to support the demand of our system. The operating system of L1 is Windows 10 with the latest version of Google Chrome and Firefox browser installed where we will run our web interface of the application. AP1 and AP2 are running on Android 8+ with our application installed.

### 5.1.2 Test data

The application will be tested with the standard start configuration without predefined data.

### 5.1.3 Personnel required

All members of the group will take part in the acceptance tests, as specified in the according test suite.

## 5.2 Acceptance criteria

### 5.2.1 Criteria for acceptance / rejection

Acceptable criteria: The test case returns the predefined result for that specific test.
Unacceptable criteria: The test case returns an unexpected result or error for that specific test.

### 5.2.2 Interruptions

Unacceptable criteria that are not caused by the system.

### 5.2.3 Continuation of the test

The test case should be restarted and tested again.

### 5.2.4 Testcases Component 1

#### 5.2.4.1 Testsuite: Android application user registration and login

#### 5.2.4.2 Test Case C1-1

| | |
|---|---|
| *Input* | The user inputs the info to create a new user and clicks the button to sign-in |
| *Operation* | Create a new user |
| *Output* | User account is registered in DB |

#### 5.2.4.3 Test Case C1-2

| | |
|---|---|
| *Input* | username, password |
| *Operation* | Log in to account |
| *Output* | User is logged into the application |

#### 5.2.4.4 Test Case C1-3

| | |
|---|---|
| *Input* | password, new password |
| *Operation* | Change password |
| *Output* | The password of the user is changed |

#### 5.2.4.5 Test Case C1-4

| | |
|---|---|
| *Input* | email adress |
| *Operation* | Reset password |
| *Output* | The password of the user is changed and the new password is sent to the email adress given by the user |

#### 5.2.4.6 Test Case C1-5

| | |
|---|---|
| *Input* | logout request |
| *Operation* | Log out |
| *Output* | The currently logged in user is logged out of the system |

#### 5.2.4.7 Test Case C1-6

| | |
|---|---|
| *Input* | delete user request |
| *Operation* | Delete user |
| *Output* | A selected user is deleted from the system |

### 5.2.5 Testcases Component 2

#### 5.2.5.1 Testsuite: Android application mood recording

#### 5.2.5.2 Test Case C2-1

| | |
|---|---|
| *Input* | mood on a scale of 1-5, relaxation on a scale of 1-5, companions that are near, special situations |
| *Operation* | Mood recording |
| *Output* | The mood of the user is recorded and locally stored on the android device |

#### 5.2.5.3 Test Case C2-2

| | |
|---|---|
| *Input* | accept notification |
| *Operation* | Notification prompt |
| *Output* | The user is directed to the mood recording after clicking on the notification |

### 5.2.6 Testcases Component 3

#### 5.2.6.1 Testsuite: Android application companion/friend management

#### 5.2.6.2 Test Case C3-1

| | |
|---|---|
| *Input* | unique user code |
| *Operation* | Adding companion |
| *Output* | The companion is added to the friend list of the user |

#### 5.2.6.3 Test Case C3-2

| | |
|---|---|
| *Input* | relationship to the invited person |
| *Operation* | Inviting companion |
| *Output* | An invitation email is send to the invited user |

#### 5.2.6.4 Test Case C3-3

| | |
|---|---|
| *Input* | new relationship status |
| *Operation* | Changing relationship status to companion |
| *Output* | The current relationship status to the friend is changed |

#### 5.2.6.5 Test Case C3-4

| | |
|---|---|
| *Input* | removing companion operation |
| *Operation* | Remove companion from companions list |
| *Output* | The companion which has to be removed is removed from the companions list |

### 5.2.7 Testcases Component 4

#### 5.2.7.1 Testsuite: Android application data visualization

#### 5.2.7.2 Test Case C4-1

| | |
|---|---|
| *Input* | time frame, data to visualize, up to 5 companions whose mood and relaxation will be included in the visualization. |
| *Operation* | Visualize mood data |
| *Output* | A visualization with the user defined parameters |

### 5.2.8 Testcases Component 5

#### 5.2.8.1 Testsuite: Android application help page

#### 5.2.8.2 Test Case C5-1

| | |
|---|---|
| *Input* | help page request |
| *Operation* | Show help page |
| *Output* | The help page is prompted to the user |

### 5.2.9 Testcases Component 6

#### 5.2.9.1 Testsuite: Web interface

#### 5.2.9.2 Test Case C6-1

| | |
|---|---|
| *Input* | admin, admin-password |
| *Operation* | Log in to Web interface |
| *Output* | Admin is logged into the Web interface |

#### 5.2.9.3 Test Case C6-2

| | |
|---|---|
| *Input* | admin-password, new admin-password |
| *Operation* | Change admin-password |
| *Output* | The password of the admin is changed |

#### 5.2.9.4 Test Case C6-3

| | |
|---|---|
| *Input* | change settings request |
| *Operation* | Change project settings |
| *Output* | The changes made by the administrator are saved on the server |

#### 5.2.9.5 Test Case C6-4

| | |
|---|---|
| *Input* | delete user request |
| *Operation* | Delete user |
| *Output* | All account data for the specified account is deleted from the server |

#### 5.2.9.6 Test Case C6-5

| | |
|---|---|
| *Input* | download data request |
| *Operation* | Download data |
| *Output* | A csv file is generated and downloaded |

#### 5.2.9.7 Test Case C6-6

| | |
|---|---|
| *Input* | archive data request |
| *Operation* | Archive data |
| *Output* | Changes are saved on the server |

#### 5.2.9.8 Test Case C6-7

| | |
|---|---|
| *Input* | visualize data request |
| *Operation* | Visualize data |
| *Output* | The data is visualized in a graph |

# 6 System Tests

## 6.1 Introduction

### 6.1.1 Purpose of the tests

The main purpose of the system tests is to test the whole system, and thus discover all possible errors that may occur.

### 6.1.2 Test coverage area

The complete system will be tested.

## 6.2 Test environment

### 6.2.1 Introduction

The test environment is based on the systems we will use to create the system.

### 6.2.2 Hardware and Test-Software

The hardware on which we are going to run the system tests is capable to support the demand of the system. We will use Android 8+ for the mobile app and modern browsers (Chrome, Firefox) for the web interface.

### 6.2.3 Test-Files

The program will be tested with a test database. We will use several accounts for different test sections.

### 6.2.4 Manpower requirements

All members of the group will take part in the system tests. We will, separately and together, test the different aspects of the system as specified by the according test suites.

## 6.3 Acceptance criteria

### 6.3.1 Acceptable and unacceptable criteria

Acceptable criteria: The test case returns the specified result for that case. Unacceptable criteria: The test case returns an error, due to a function or component which has not returned the specified result.

### 6.3.2 Interrupts

Unacceptable criteria that are not caused by the system

### 6.3.3 Resumption of test

The test section should be executed over again.

## 6.4 Test section 1: User accounts

### 6.4.1 Introduction

Test section 1 from the system tests is going to validate if the registration, login and logout work correctly.

### 6.4.2 Test suites

#### 6.4.2.1 Test suite: Registration tests

#### 6.4.2.2 Test Case TG1-1-1

| | |
|---|---|
| *Entry conditions* | User has app installed and is not logged in |
| *Input* | Valid account information |
| *Operation* | Registration |
| *Output* | Account created on server |
| *Exit conditions* | Successfully created account |

#### 6.4.2.3 Test Case TG1-1-2

| | |
|---|---|
| *Entry conditions* | User has app installed and is not logged in |
| *Input* | Account information with invalid email |
| *Operation* | Registration |
| *Output* | Error message ”Invalid email address” |
| *Exit conditions* | Account is not created |

#### 6.4.2.4 Test suite: Log in tests

#### 6.4.2.5 Test Case TG1-2-1

| | |
|---|---|
| *Entry conditions* | User has app installed and is not logged in |
| *Input* | Valid account credentials |
| *Operation* | Log in |
| *Output* | Account is saved on device |
| *Exit conditions* | User is logged in |

#### 6.4.2.6 Test Case TG1-2-2

| | |
|---|---|
| *Entry conditions* | User has app installed and is not logged in |
| *Input* | Invalid account credentials |
| *Operation* | Log in |
| *Output* | Error message ”Invalid credentials” |
| *Exit conditions* | User is not logged in |

#### 6.4.2.7 Test Case TG1-2-3

| | |
|---|---|
| *Entry conditions* | User has app installed and is not logged in |
| *Input* | Valid email address |
| *Operation* | Reset password |
| *Output* | Message ”If this account exists, an email was sent” |
| *Exit conditions* | Email with reset link was sent |

#### 6.4.2.8 Test Case TG1-2-4

| | |
|---|---|
| *Entry conditions* | User has app installed and is not logged in |
| *Input* | Invalid email address |
| *Operation* | Reset password |
| *Output* | Message ”If this account exists, an email was sent” |
| *Exit conditions* | No email was sent |

#### 6.4.2.9 Test suite: Log out tests

#### 6.4.2.10 Test Case TG1-3-1

| | |
|---|---|
| *Entry conditions* | User has app installed and is logged in |
| *Input* | - |
| *Operation* | Log out |
| *Output* | Account is removed from the device |
| *Exit conditions* | User is logged out |

#### 6.4.2.11 Test suite: Password changing tests

#### 6.4.2.12 Test Case TG1-4-1

| | |
|---|---|
| *Entry conditions* | User has app installed and is logged in |
| *Input* | Correct old and new password |
| *Operation* | Change password |
| *Output* | Changes are reflected on the server |
| *Exit conditions* | Password is changed |

#### 6.4.2.13 Test Case TG1-4-2

| | |
|---|---|
| *Entry conditions* | User has app installed and is logged in |
| *Input* | Wrong old and new password |
| *Operation* | Change password |
| *Output* | Error message ”Wrong current password” |
| *Exit conditions* | No changes are done |

## 6.5 Test section 2: Recording mood

### 6.5.1 Introduction

Test section 2 from the system tests is going to validate if recording mood both voluntarily and by notifications including all features work correctly.

### 6.5.2 Test suites

#### 6.5.2.1 Test suite: Record mood tests

#### 6.5.2.2 Test Case TG2-1-1

| | |
|---|---|
| *Entry conditions* | User has app installed and is logged in |
| *Input* | Mood, relaxation, near companions, special situations |
| *Operation* | Record mood with special situations (voluntary) |
| *Output* | Data is saved on the device |
| *Exit conditions* | Data is successfully saved |

#### 6.5.2.3 Test Case TG2-1-2

| | |
|---|---|
| *Entry conditions* | User has app installed and is logged in |
| *Input* | Mood, relaxation, near companions |
| *Operation* | Record mood without special situations (only by notification) |
| *Output* | Data is saved on the device |
| *Exit conditions* | Data is successfully saved |

#### 6.5.2.4 Test suite: Notification tests

#### 6.5.2.5 Test Case TG2-2-1

| | |
|---|---|
| *Entry conditions* | User has app installed and is logged in |
| *Input* | First notification prompt of the day |
| *Operation* | Accept notification |
| *Output* | Record mood screen |
| *Exit conditions* | User is prompted to record mood with special situations |

#### 6.5.2.6 Test Case TG2-2-1

| | |
|---|---|
| *Entry conditions* | User has app installed and is logged in |
| *Input* | Not first notification prompt of the day |
| *Operation* | Accept notification |
| *Output* | Record mood screen |
| *Exit conditions* | User is prompted to record mood without special situations |

#### 6.5.2.7 Test Case TG2-2-3

| | |
|---|---|
| *Entry conditions* | User has app installed and is logged in |
| *Input* | Notification prompt |
| *Operation* | Miss notification |
| *Output* | Data with miss flag is saved on the device upon next notification |
| *Exit conditions* | Data is successfully saved |

## 6.6 Test section 3: Companions

### 6.6.1 Introduction

Test section 3 from the system tests is going to validate if adding, removing, changing and inviting companions via Email work correctly.

### 6.6.2 Test suites

#### 6.6.2.1 Test suite: Adding and inviting companion tests

#### 6.6.2.2 Test Case TG3-1-1

| | |
|---|---|
| *Entry conditions* | User has app installed and is logged in, user knows their companion's unique code |
| *Input* | Relation, valid unique code |
| *Operation* | Add companion by code |
| *Output* | Data is saved on the server |
| *Exit conditions* | Companion is successfully added |

#### 6.6.2.3 Test Case TG3-1-2

| | |
|---|---|
| *Entry conditions* | User has app installed and is logged in, user knows thier companion's unique code |
| *Input* | Relation, invalid unique code |
| *Operation* | Add companion by code |
| *Output* | Error message "Invalid user code" |
| *Exit conditions* | No changes are done |

#### 6.6.2.4 Test Case TG3-1-3

| | |
|---|---|
| *Entry conditions* | User has app installed and is logged in |
| *Input* | Relation, valid email |
| *Operation* | Invite by email |
| *Output* | Message "An invite email was sent" |
| *Exit conditions* | Email was sent |

#### 6.6.2.5 Test Case TG3-1-4

| | |
|---|---|
| *Entry conditions* | User has app installed and is logged in |
| *Input* | Relation, invalid email |
| *Operation* | Invite by email |
| *Output* | Error message "Invalid email address" |
| *Exit conditions* | No email was sent |

#### 6.6.2.6 Test suite: Changing and removing companion tests

#### 6.6.2.7 Test Case TG3-2-1

| | |
|---|---|
| *Entry conditions* | User has app installed and is logged in, user has not changed the relation before |
| *Input* | Companion, new relation |
| *Operation* | Change relation |
| *Output* | Changes are saved on the server |
| *Exit conditions* | Relation is successfully changed |

#### 6.6.2.8 Test Case TG3-2-2

| | |
|---|---|
| *Entry conditions* | User has app installed and is logged in, user has changed the relation before |
| *Input* | Companion, new relation |
| *Operation* | Change relation |
| *Output* | Error message "Relation with this user cannot be changed again" |
| *Exit conditions* | No changes are done |

#### 6.6.2.9 Test Case TG3-2-3

| | |
|---|---|
| *Entry conditions* | User has app installed and is logged in, user has companion on their list |
| *Input* | Companion |
| *Operation* | Remove companion |
| *Output* | Data is saved on the server |
| *Exit conditions* | Companion is successfully removed |

## 6.7 Test section 4: Visualize data

### 6.7.1 Introduction

Test section 4 from the system tests is going to validate if the dynamic data visualization works correctly.

### 6.7.2 Test suites

#### 6.7.2.1 Test suite: Visualize own data tests

#### 6.7.2.2 Test Case TG4-1-1

| | |
|---|---|
| *Entry conditions* | User has app installed and is logged in, no companions are selected |
| *Input* | New time window |
| *Operation* | Change time window |
| *Output* | Visualization graph is updated |
| *Exit conditions* | New time window is visualized |

#### 6.7.2.3 Test Case TG4-1-2

| | |
|---|---|
| *Entry conditions* | User has app installed and is logged in |
| *Input* | Select data (mood, relaxation, companions, situations) |
| *Operation* | Change visualized data |
| *Output* | Visualization graph is updated |
| *Exit conditions* | Changes are visualized |

#### 6.7.2.4 Test suite: Visualize companion data tests

#### 6.7.2.5 Test Case TG4-2-1

| | |
|---|---|
| *Entry conditions* | User has app installed and is logged in, time window does not cover current experiment cycle |
| *Input* | Select companion |
| *Operation* | Visualize companion data |
| *Output* | Visualization graph is updated |
| *Exit conditions* | Companion data is visualized |

#### 6.7.2.6 Test Case TG4-2-2

| | |
|---|---|
| *Entry conditions* | User has app installed and is logged in, time window covers current experiment cycle |
| *Input* | Select companion |
| *Operation* | Visualize companion data |
| *Output* | Error message ”Time window covers current experiment cycle with this companion” |
| *Exit conditions* | No changes are done |

#### 6.7.2.7 Test Case TG4-2-3

| | |
|---|---|
| *Entry conditions* | User has app installed and is logged in, time window does not cover current experiment cycle |
| *Input* | New time window |
| *Operation* | Change time window |
| *Output* | Visualization graph is updated |
| *Exit conditions* | New time window is visualized |

#### 6.7.2.8 Test Case TG4-2-4

| | |
|---|---|
| *Entry conditions* | User has app installed and is logged in, time window covers current experiment cycle |
| *Input* | New time window |
| *Operation* | Change time window |
| *Output* | Error message ”Time window covers current experiment cycle with selected companions” |
| *Exit conditions* | No changes are done |

## 6.8 Test section 5: Web interface

### 6.8.1 Introduction

Test section 5 from the system tests is going to validate if the web interface works correctly.

### 6.8.2 Test suites

#### 6.8.2.1 Test suite: Admin log in tests

#### 6.8.2.2 Test Case TG5-1-1

| | |
|---|---|
| *Entry conditions* | Admin has access to the web interface |
| *Input* | Valid admin account credentials |
| *Operation* | Admin log in |
| *Output* | Admin account data is saved in cache |
| *Exit conditions* | Admin is logged in |

#### 6.8.2.3 Test Case TG5-1-2

| | |
|---|---|
| *Entry conditions* | Admin has access to the web interface |
| *Input* | Invalid admin account credentials |
| *Operation* | Admin log in |
| *Output* | Error message ”Invalid credentials” |
| *Exit conditions* | Admin is not logged in |

#### 6.8.2.4 Test suite: Change admin password tests

#### 6.8.2.5 Test Case TG5-2-1

| | |
|---|---|
| *Entry conditions* | Admin is logged in to the web interface |
| *Input* | Correct old and new password |
| *Operation* | Change admin password |
| *Output* | Changes are saved on the server |
| *Exit conditions* | Admin password is changed |

#### 6.8.2.6 Test Case TG5-2-2

| | |
|---|---|
| *Entry conditions* | Admin is logged in to the web interface |
| *Input* | Wrong old and new password |
| *Operation* | Change admin password |
| *Output* | Error message ”Wrong current password” |
| *Exit conditions* | No changes are done |

#### 6.8.2.7 Test suite: Project settings tests

#### 6.8.2.8 Test Case TG5-3-1

| | |
|---|---|
| *Entry conditions* | Admin is logged in to the web interface |
| *Input* | New consent text |
| *Operation* | Change consent text |
| *Output* | Changes are saved on the server |
| *Exit conditions* | Consent text is changed |

#### 6.8.2.9 Test Case TG5-3-2

| | |
|---|---|
| *Entry conditions* | Admin is logged in to the web interface |
| *Input* | New frequency options |
| *Operation* | Change notification frequency |
| *Output* | Changes are saved on the server |
| *Exit conditions* | Notification frequency is changed |

#### 6.8.2.10 Test Case TG5-3-3

| | |
|---|---|
| *Entry conditions* | Admin is logged in to the web interface |
| *Input* | Check / uncheck notification synchrony |
| *Operation* | Change notification synchrony |
| *Output* | Changes are saved on the server |
| *Exit conditions* | Notification synchrony is changed |

#### 6.8.2.11 Test Case TG5-3-4

| | |
|---|---|
| *Entry conditions* | Admin is logged in to the web interface |
| *Input* | New cycle duration in days |
| *Operation* | Change experiment cycle |
| *Output* | Changes are saved on the server |
| *Exit conditions* | Experiment cycle duration is changed |

#### 6.8.2.12 Test suite: Manage user accounts tests

#### 6.8.2.13 Test Case TG5-4-1

| | |
|---|---|
| *Entry conditions* | Admin is logged in to the web interface |
| *Input* | User credentials (id, email) |
| *Operation* | Delete user account |
| *Output* | All user data is removed from the server |
| *Exit conditions* | User account is deleted |

#### 6.8.2.14 Test suite: Download and archive data tests

#### 6.8.2.15 Test Case TG5-5-1

| | |
|---|---|
| *Entry conditions* | Admin is logged in to the web interface |
| *Input* | - |
| *Operation* | Download all data |
| *Output* | CSV file with all user data |
| *Exit conditions* | Data is downloaded |

#### 6.8.2.16 Test Case TG5-5-2

| | |
|---|---|
| *Entry conditions* | Admin is logged in to the web interface |
| *Input* | Time window |
| *Operation* | Download data by time window |
| *Output* | CSV file with user data for time window |
| *Exit conditions* | Data is downloaded |

#### 6.8.2.17 Test Case TG5-5-3

| | |
|---|---|
| *Entry conditions* | Admin is logged in to the web interface |
| *Input* | Time window |
| *Operation* | Archive data by time window |
| *Output* | Data is flagged as archived on the server |
| *Exit conditions* | Data is marked as archived |

#### 6.8.2.18 Test suite: Web data visualization tests

#### 6.8.2.19 Test Case TG5-6-1

| | |
|---|---|
| *Entry conditions* | Admin is logged in to the web interface |
| *Input* | Time window |
| *Operation* | Change time window |
| *Output* | Visualization graph is updated |
| *Exit conditions* | New time window is visualized |

#### 6.8.2.20 Test Case TG5-6-2

| | |
|---|---|
| *Entry conditions* | Admin is logged in to the web interface |
| *Input* | Select data (mood, relaxation, companions, situations) |
| *Operation* | Change visualized data |
| *Output* | Visualization graph is updated |
| *Exit conditions* | Changes are visualized |

#### 6.8.2.21 Test Case TG5-6-3

| | |
|---|---|
| *Entry conditions* | Admin is logged in to the web interface |
| *Input* | User (id, email) |
| *Operation* | Change user to visualize |
| *Output* | Visualization graph is updated |
| *Exit conditions* | New user data is visualized |

Part III

# Documentation of the Teamwork

## 7 General organization

- Project Manager - Niklas Maier
- Customer Relationship Officer - Johanna Bell
- Documentation Lead - Luca Bosch
- Technical Lead - Sebastian Schwarz
- Repository Lead - Simon Vogelbacher
- Quality Assurance Manager - Yasmin Hoffmann

## 8 Individual tasks of the team members

### 8.1 Johanna Bell

- Sections 3.1, 3.2, 3.3, 5.1, 5.2
- Requirements and Client acceptance tests

### 8.2 Luca Bosch

- Sections 1.3, 7.1-7.6, 8.1-8.5
- Introduction and Documentation of the teamwork

### 8.3 Niklas Maier

- Sections 1.1, 1.2, 1.3, 1.4, 6.1
- Introduction and System tests

### 8.4 Sebastian Schwarz

- Sections 3.4, 6
- Use cases, system models and system tests

## 8.5 Simon Vogelbacher

- Sections 3.1, 3.2, 3.3, 5.1, 5.2
- Requirements and Client acceptance tests

## 8.6 Yasmin Hoffman

- Sections 1.1, 1.2, 1.3, 1.4, 6.1
- Introduction and System tests

# 9 Protocols

## 9.1 Protocol from April 27th

**Date:** 27.04.20 **Time:** 16:00-17:00

- Discussed Topics
  - In which form are we going to hold regular online meetings in the future?
  - Who is going to create the protocols of these meetings?
  - What are the roles of each team member and who is going to be the project manager?
  - How do we want to call our team?
  - Who is going to schedule a meeting with our advisor?
  - Where are we going to gather questions for said meeting?
- Conclusions
  - Future meetings will be held on a Discord channel.
  - Everyone can contribute to the protocol, the responsibility for it will be taken by the Documentation Lead
  - Roles were distributed.
  - Meeting with advisor booked by the Project Manager for 13:00-13:30 on Tue. 28/04/20.
  - Qustions for the meeting will be gathered in a Discord channel.
  - We have decided on the team name 'Angry Nerds'.

## 9.2 Protocol from May 1st

**Date:** 01.05.20 **Time:** 11:00-13:00

- Discussed Topics
  - We talked about the meeting with our advisor and discussed the new information gained by getting answers to our questions
  - We discussed the design choices regarding the application layout
  - We discussed how the steps of mood selection should take place.

- Next meeting Monday 16:00.

- Conclusions
  - We decided what to do this week
  - We gathered some more questions
  - We are going to use a bottom bar
  - We decided which elements to include in each page of the app and how the sliders, switches and other interactive buttons are going to look like. For now there will be a home screen, where you add moods, companion screen, where you add friends and relatives, settings and stats screen, where you can visualize past entries.

## 9.3 Protocol from May 7th

**Date:** 07.05.20 **Time:** 17:00-18:45

- Discussed Topics
  - We discussed our solutions to the requirement catalogue
  - Meeting with the client timeslot on friday
  - We discussed how we are going to talk to the client
- Conclusions
  - The customer relations officer will introduce our group to the client
  - We merged our solutions from the requirement catalogues, removed duplicates and refactored the text
  - We are flexible regarding the meeting on friday

## 9.4 Protocol from May 11th

**Date:** 07.05.20 **Time:** 15:00-15:20

- Discussed Topics
  - Current progress and what we are going to do next
  - We agreed that Sebastians work on the System Requirements (3.4) was to our liking
- Conclusions

    - We will meet again on Friday, 12:00 pm so we can get a look at more finished work and put it together

## 9.5 Protocol from May 15th

**Date:** 07.05.20 **Time:** 15:00-15:20

- Discussed Topics
    - We discussed work our team did this week
    - We created the work protocol for this week online
    - We discussed errors occurring in the LaTeX-Template for D1
- Conclusions
    - We will ask our advisor about how we can fix the errors
    - Next meeting 19.5. 17:00