Project Fingerpaint

URD-0.4

User Requirements Document

Authors:
Tessa Belder (0739377)
Lasse Blaauwbroek (0749928)
Thom Castermans (0739808)
Roel van Happen (0751614)
Benjamin van der Hoeven (0758975)
Femke Jansen (0741948)
Hugo Snel (0657700)

Junior Management:
Simon Burg
Areti Paziourou
Luc de Smet

Senior Management:
Mark van den Brand, MF 7.096
Lou Somers, MF 7.145

Technical Advisor:
Ion Barosan, MF 7.082

Customer:
Patrick Anderson, GEM-Z 4.147

May 3, 2013

Abstract

This document describes the User Requirements of FINGERPAINT and is based on discussions with the client; the requirements described in this document are conform to the client's wishes. This project is part of the Software Engineering Project (2IP35) and is one of the assignments at Eindhoven University of Technology. The User Requirements Document (URD) is based on the ESA standard for software development, as set by the European Space Agency (ESA) [1]. This document lists what the FINGERPAINT application should be capable of, and how and in what environment it should function.

Contents

Document Status Sheet

General

Document title:	User Requirements Document
Identification:	URD0.4
Author:	Tessa Belder, Roel van Happen, Benjamin van der Hoeven, Femke Jansen, Hugo Snel
Document status:	Initial

Document history

Version	*Date*	*Author*	*Reason of change*
0.1	24-Apr-2013	Tessa Belder Roel van Happen Benjamin van der Hoeven Femke Jansen Hugo Snel	Initial version.
0.2	26-Apr-2013	Tessa Belder Roel van Happen Benjamin van der Hoeven Femke Jansen Hugo Snel	Revised version as prompted by the client meeting on 25-Apr
0.3	2-May-2013	Tessa Belder Roel van Happen Benjamin van der Hoeven Femke Jansen Hugo Snel	Revised version following from feedback of team managers and technical advisor.
0.4	3-May-2013	Tessa Belder Roel van Happen Benjamin van der Hoeven Femke Jansen Hugo Snel	Revised version following from internal feedback.

Document Change Records since previous issue

General

Date:	3-May-2013
Document title:	User Requirements Document
Identification:	URD0.4

Changes

For each of the document changes listed here, we will refer to the pages and paragraphs in the previous version of this document, to clarify what exactly has been changed.

Page	*Paragraph*	*Reason to change*
-	Abstract	Replaced 'how it should function and in what environment it should function' to 'how and in what environment it should function'.
5	1	Added an introductionary sentence.
5	1.3	Added definitions, and made a seperate list with abbreviations.
8	1.3.1	Added Mixing Protocol, concentration Distribution, and Mixing run to definition list.
8	2.1	Reworded the first paragraph to remove ambiguity.
8	2.1	Removed implementation detail: constraints and a vector.
8	2.2	Added general description to section head.
8	2.2.1	Reworded first paragraph to clarify that the client device does not do simulation and to remove some ambiguity.
9	2.2.1	Clarified the second constraint to explain what the characteristics mean.
9	2.2.1	Extended the paragraph for the third constraint by giving an example protocol.
9	2.2.1	Reworded last paragraph to clarify how and where the user can draw the fluids.
9	2.2.2	Clarified that the application will also probably run on desktop PCs, but this is not actively supported.
9	2.2.2	Clarified that multiple runs can be compared, not just previous runs and the current.

Page	*Paragraph*	*Reason to change*
9	2.2.2	Clarified what is meant by intermediate results.
9	2.2.2	Listed .svg as an example file format instead of a requirement, so it can be changed in the future if we desire.
9	2.2.2	Clarified that the result is also visualised on the client device.
9	2.3	Reworded the paragraph to make it easier to read.
9	2.3	Specified what we will save.
9	2.4	Added description of the target audience.
9	2.4	Added that it is possible to compare mixers using their performance results.
9	2.5	Clarified when intermediate results are visualised.
11	3	Added reference to Appendix A.
11	3	Changed formulation of sentence 'Any requirements following from further request will be added here'.
11-15	3.1/3.2	Grouped requirements into smaller blocks.
11-12	3.1	Split CPR03 in CPR3 and CPR4.
14-16	3.1	Changed formulation of CPR5, CPR11, CPR17, CPR18, CPR19 and CPR25 - CPR31.
13	3.1	Added CPR22.
16	3.1	Added CPR23 and CPR24.
14	3.2	Split CNR02 in CNR2 and CNR3. Split CNR03 in CNR4, CNR5 and CNR6.
14	3.2	Changed priority of CNR6 to won't have, because Opera isn't compatible with the testing environment.
14	3.2	Changed CNR10, CNR11 and CNR12 to 'average waiting time', instead of 'waiting time'.
15	3.2	Changed formulation of CNR13.
17	3.1	Rephrased requirements to clarify that images/animations are exported, also removed the corresponding delete requirements.
-	Appendix	Added alternatives to all use cases, where applicable.
-	Appendix	Improved the readability. Inserted "the" a lot. The old version had more of a mechanical summation rather than fluent text.
-	Appendix	Changed preconditions of use cases to includes at step 1.
-	Appendix	Added 'safe mixing run', 'remove initial concentration distribution', 'safe graph of multiple runs', 'remove mixing protocol', 'load saved mixing protocol' and 'load predefined mixing protocol' use cases.
-	Appendix	Replaced included preconditions of all 'Load' and 'Remove' use-cases by alternatives for when the history is empty.
-	Appendix	Updated format, inserted UCD, changed order of use cases to match UCD.
19	Appendix	"Define an initial distribution"; Inserted intermediate steps of saving file (choose location/name). Added 'at the chosen location' for safe step at system side.

Page	*Paragraph*	*Reason to change*
21	Appendix	"Define mixing protocol(2)"; Inserted intermediate steps of saving file (choose location/name). Added 'at the chosen location' for safe step at system side. Changed reference from *Define mixing protocol (0)* to *A.10* (also fixing the erroneous (0) (should be (1))). Inserted steps from A.9 (user can do paint actions at intermediate steps)
19	Appendix	"Define an initial distribution"; Inserted intermediate steps of saving file (choose location/name). Added 'at the chosen location' for safe step at system side.
21	Appendix	"Define mixing protocol(2)"; Inserted intermediate steps of saving file (choose location/name). Added 'at the chosen location' for safe step at system side. Changed reference from *Define mixing protocol (0)* to *A.10* (also fixing the erroneous (0) (should be (1))). Inserted steps from A.9 (user can do paint actions at intermediate steps)
21	Appendix	"Define mixing protocol(2)"; Inserted intermediate steps of saving file (choose location/name). Added 'at the chosen location' for safe step at system side. Changed reference from *Define mixing protocol (1)* to *A.10*.

Chapter 1

Introduction

This chapter lists general information about this document.

1.1 Purpose

This document contains the requirements for FINGERPAINT. These requirements are a negotiated agreement between prof.dr.ir. P.D. Anderson and Group FINGERPAINT. The listed requirements will be implemented in the FINGERPAINT application according to their priorities. The requirements with the *must have* priority (see chapter 3) will be assured and if time allows, other requirements might also be implemented. Any changes to these requirements require the full consent of both parties.

1.2 Scope

FINGERPAINT is an application designed and developed by Group FINGERPAINT for prof. dr. ir. P.D. Anderson. The application provides a cross-platform tool to visualise fluid mixing. Users can define the initial concentration distribution, as well as manipulate the mixing protocol. The resulting fluid distribution can be stored and analysed by the user for comparison purposes.

1.3 List of definitions and abbreviations

1.3.1 Definitions

Client Prof.dr.ir. P.D. Anderson.

Firefox A web browser developed by Mozilla.

Google Chrome A web browser developed by Google.

Internet Explorer A web browser developed by Microsoft.

iOS A mobile operating system developed by Apple.

iOS Safari A web browser developed by Apple designed for devices running iOS.

Opera A web browser developed by Opera Software.

Safari A web browser developed by Apple.

iPhone A line of smartphones developed by Apple.

Concentration Distribution The distribution of the fluids in the mixer.

Mixing Protocol Sequence of mixer wall movements with their step(D).

Step(D) Parameter of the mixing protocol that indicates the amount of time the wall should move.

#steps Parameter of the mixing protocol that indicates how many times the protocol is applied.

Mixing Performance List of numbers representing the fluid segregation throughout the mixing protocol.

Mixing Run Combination of an initial concentration distribution, and a mixing protocol.

1.3.2 Abbreviations

2IP35	The Software Engineering Project
CM	Configuration Manager
CPR	Capability Requirement
CNR	Constraint Requirement
PC	Personal Computer
TU/e	Eindhoven University of Technology
SEP	Software Engineering Project
SR	Software Requirements
SRD	Software Requirements Document
URD	This document, the User Requirements Document

1.4 List of references

[1] ESA, *ESA Software Engineering Standards*. ESA, March 1995.

[2] COLEY consulting, "Moscow prioritisation." `http://www.coleyconsulting.co.uk/moscow.htm`. [Online; accessed 24-April-2013].

1.5 Overview

The remaining chapters describe the user requirements in more detail. Chapter 2 gives a general description of

- The relation to other systems (2.1)
- The main capabilities (2.2)

- Constraint information and justification (2.3)
- User characteristics (2.4)
- The operational environment (2.5)
- Assumptions and dependencies (2.6)

Chapter 3 gives a detailed list of the system's capability requirements in section 3.1, and a list of the constraint requirements is given in section 3.2.
In the appendices all use cases related to this project are described. Appendix A contains the use case diagram, which shows the relations between the different use cases. Appendix B gives detailed descriptions of all individual use cases.

Chapter 2

General description

This chapter describes general aspects of the application to be created as requested by the client.

2.1 Product perspective

The aim of this project is to deliver an application that allows the user to visualise the mixing of fluids. To accomplish this, the user should be able to specify initial parameters, such as the shape of the mixer, in an intuitive and attractive user interface. These parameters should be set on a client device, which will send them to a server (owned by our Client) to compute the displacement of the fluids. When this server has computed the resulting fluid concentration distribution, this distribution is sent back to the client device, where it can be visualised.

A similar project was initiated around eleven years ago. The result of this project was a MATLAB implementation that achieved a similar goal as our project. However, its user interface is outdated by now, and it is impossible to comfortably use this solution on a mobile device. Part of this original solution was a FORTRAN implementation for the displacement of the fluids. This implementation is still available, and we are to use it as a black box which can compute the resulting concentration distribution given some constraints.

2.2 General capabilities

This chapter describes the general properties our application should have. Firstly, the mixing constraints are described, detailing the parameters the user should set in order to be able to compute the displacement of fluids. Secondly, we describe additional capabilities of the application, such as storage and portability.

2.2.1 Mixing constraints

As mentioned before, the application should be able to visualise the mixing of fluids. There are a number of constraints to be specified on the client device. Of these constraints, the first is the geometry of the mixer. There are four kinds of geometries in total: rectangle, square, circle, and *Journal Bearing*. We will start by implementing support for rectangular geometries, and we will implement more geometries if time permits.

The second constraint concerns the characteristics of the mixer. These influence the flow of the fluids. Stored in the server are multiple pre-computed matrices which each specify one set of mixer characteristics. Whenever these characteristics are changed, a different matrix must be used. Each geometry has its own set of possible characteristics and hence its own set of matrices.

The third constraint concerns parameters applicable to the mixer and the mixing protocol. Available parameters are determined by the type of mixer defined in the second constraint. For example, a rectangular mixing geometry has two walls that can be moved. The step parameter D indicates the amount of time the wall should move, and is fixed for the entire protocol. Possible values for D are 4, 2, 1, 0.5, 0.25 and 0.1, which can be combined to create other values. For example, a value of 6.2 can be achieved by setting $D = 4 + 2 + 0.1 + 0.1$. The user can specify whether a complete protocol should be executed a number of times, or to only execute one step at a time. For a protocol for a rectangular geometry, the user can choose to move each of the two walls for D time units at a time. The user can move the top wall to the right (T), the bottom wall to the left (B), or one of those walls to the other direction ($-T$ or $-B$).

The fourth constraint is the initial concentration distribution of the fluids. This can be specified either by drawing on a canvas that has the shape of the selected mixer, or by loading an existing initial distribution. To draw on the canvas, the user should first select one of two possible colors, namely black or white. After this the user can tap on or drag over the screen, to indicate where the selected fluid should be placed. It is not possible to use any other method than the free-form drawing described here.

2.2.2 Additional capabilities

The main user interface is intended to be run on a mobile device, such as an iPhone. As we are creating a cross-platform solution, the application should also run on desktop PCs. However, we are not actively supporting such devices. On the chosen client device, the constraints mentioned in Section 2.2.1 should be specified. When the initial parameters have been set, the computation of the new concentration distribution is offloaded to the Client's server. After each iteration of the mixing protocol, the result – along with a metric indicating the performance of the mixer – is sent back to the client device to be shown to the user via a two-dimensional image of the fluid concentration distribution.

A history of past simulations is stored on the device to compare multiple runs. The result of runs should be exportable to commonly-used vector graphics files with support for alpha channels (to realise transparency). An additional capability might involve saving of entire runs – hence including intermediate results – as an animation.

After each result is received and visualised by the client device, the user has the choice of continuing with the received concentration distribution, or to reset the concentration distribution to a completely white concentration distribution.

2.3 General constraints

The user interface should be suitable for mobile devices, so it will be easy to share the visualised results with other people, and to quickly try out new ideas for mixers wherever the user may be.

We assume that the server can compute the displacement of fluids reasonably fast, so the visualisation of results can be handled quickly. When the new concentration distribution has been computed, this concentration distribution is sent back to the client device along with a metric to indicate the performance of the mixer.

As we do not want to be locked to one specific type of device, we have chosen to design a cross-platform solution. While this means that desktop PCs should also be able to run the application, we do not actively support such devices. We will concentrate on mobile devices.

As mentioned before, it should be possible to save mixing runs for later reference. For each saved run, we store the initial distribution, the mixer and protocol used, the resulting fluid distribution and the resulting performance metric.

2.4 User characteristics

FINGERPAINT is intended to be used by people who want to quickly try out ideas for mixers and see how well they perform. For this reason the application should be reasonably fast. The user can change the shape and characteristics of the mixer to test a different idea for a mixer, and the initial concentration distribution and the mixing protocol to determine how the mixer will be evaluated. To compare different mixers and their performance, the user can save runs to an image file or an animation, or simply make a plot of the performance levels of mixers.

2.5 Environment description

The main device for the user interface is the mobile device. We are planning to create a cross-platform solution, which means it will be possible to use the application on various kinds of devices. Examples of supported devices are Apple iPhones, Android phones or tablets. The initial concentration distribution of the fluids, the mixing protocol and the shape of the mixer will be specified on such a device. As mentioned before, the user specifies initial parameters on the client device. These parameters are sent to the server, which computes the new concentration distribution. The resulting concentration distribution is sent back to the client device to display (see Figure 2.1).

As mobile devices typically do not have the power (both processing power and battery capacity) to perform intensive computations, the hard work of computing the mixing will be offloaded to a server. The starting parameters described in Section 2.2.1 will be sent to the server, which has an efficient FORTRAN implementation to solve the problem. After each iteration of the mixing protocol, intermediate results are sent back to the client device for visualisation purposes.

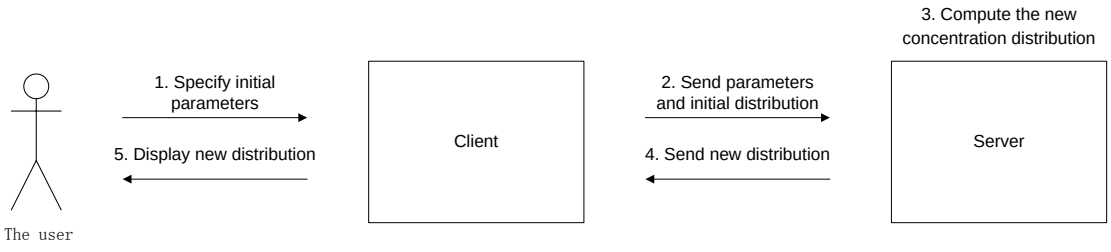

Figure 2.1: Domain model.

2.6 Assumptions and dependencies

This section contains some assumptions for the application to function properly.

- As we are creating a client-server application, we assume that there is an internet connection between the client device and the server.
- As mentioned in the previous section, the application uses the FORTRAN implementation on the server to perform all the calculations. Therefore, we assume this server can compute the new concentration distribution correctly, and it is able to correctly send this concentration distribution back to the client device.

Chapter 3

Specific requirements

This chapter explicitly states all requirements and constraints of the application to be developed. The requirements are based upon the use cases as described in Appendix B. The final product will be delivered conform these requirements. Any requirements following from further requests can only be added after consultance with, and being granted permission by, the technical advisor and senior management.
The requirements are prioritised using the MoSCoW model [2]. This model assigns one out of four priorities to each requirement:

Must have; requirements with this priority are essential for the product, and must be implemented.
Should have; requirements with this priority are not essential for the product to work. However, they are nearly as important as the *must have*'s and are therefore expected to be implemented.
Could have; requirements with this priority are a nice addition to the product, and may be implemented, if time and budget allow this.
Won't have; requirements with this priority will not be implemented in this version of the product, but may be nice to implement in future versions.

3.1 Capability requirements

This section lists all capability requirements for the product. These requirements explicitly state what the system should do.

Requirements regarding the mixer geometry input:

CPR1 The user can select a rectangular mixer geometry	*must have*
CPR2 The user can select a square mixer geometry	*should have*
CPR3 The user can select a circle mixer geometry	*could have*

CPR4	*could have*
The user can select a *Journal Bearing* mixer geometry	

Requirements regarding the initial concentration distribution input:

CPR5	*must have*
The user can define an initial concentration distribution with black and white by drawing on the touchscreen with their finger.	
CPR6	*must have*
The user can reset the current concentration distribution to a completely white concentration distribution.	
CPR7	*should have*
The user can save an initial concentration distribution.	
CPR8	*should have*
The user can select an initial concentration distribution from a list of previously saved distributions.	
CPR9	*could have*
The user can select a predefined initial concentration distribution.	
CPR10	*won't have*
The user can define an initial concentration distribution with more than two different colours.	
CPR11	*won't have*
The user can choose which colours are used for the initial concentration distribution.	

Requirements regarding the mixing protocol input:

CPR12	*must have*
The user can define a mixing protocol for a rectangular geometry as a sequence of movements of the upper and lower walls.	
CPR13	*must have*
The user can define a single movement and step (D) to be executed directly on the current concentration distribution.	
CPR14	*must have*
The user can clear the current settings for the mixing protocol.	
CPR15	*must have*
The user can define a step (D) for each movement from the mixing protocol, to indicate the time that this movement is applied.	
CPR16	*must have*
The user can define how many times the mixing protocol is applied ($\#steps$).	
CPR17	*should have*
The user can save a mixing protocol.	

CPR18	*should have*
The user can select one of the previously saved mixing protocols for the specified geometry as a mixing protocol.	
CPR19	*should have*
The user can define a mixing protocol for a square geometry as a sequence of movements of the upper and lower walls.	
CPR20	*could have*
The user can define a mixing protocol for a circle geometry as a sequence of circle rotations.	
CPR21	*could have*
The user can define a mixing protocol for a 'Journal Bearing' geometry as a sequence of rotations of inner and outer circles.	
CPR22	*could have*
The user can select a predefined mixing protocol for the specified geometry.	

Requirements regarding the application's output:

CPR23	*must have*
The user can execute a mixing run.	
CPR24	*must have*
The user can save the results of an executed mixing run locally on their device.	

Requirements regarding the exportation of results:

CPR25	*must have*
The user can view an image of the end result of a mixing run.	
CPR26	*should have*
The user can export the image from CPR25 locally to their device.	
CPR27	*should have*
The user can view the mixing performance of a mixing run in a graph.	
CPR28	*should have*
The user can export the mixing performance graph locally to their device.	
CPR29	*should have*
Users can view mixing performance results from multiple mixing runs simultaneously in one graph.	
CPR30	*could have*
The user can view an animation of the end result of a mixing run.	
CPR31	*could have*
The user can export the animation from CPR27 locally to their device.	

3.2 Constraint requirements

This section contains all constraint requirements for the application. These requirements state all demands with regard to interfaces, portability, adaptability, availability, security, safety, standards, resources and time scales.

Requirements regarding web browsers:

CNR1	*must have*
The application runs on iOS Safari version 6.0 and higher.	
CNR2	*should have*
The application runs on Firefox version 20 and higher.	
CNR3	*should have*
The application runs on Google Chrome version 26 and higher.	
CNR4	*could have*
The application runs on Internet Explorer version 10 and higher.	
CNR5	*could have*
The application runs on Safari version 6.0 and higher.	
CNR6	*won't have*
The application runs on Opera version 12.1 and higher.	

Requirements regarding operating systems:

CNR7	*must have*
The application runs on devices running on iOS version 6 and higher.	
CNR8	*should have*
The application runs on devices running on Android version 4.0 and higher.	
CNR9	*could have*
The application runs on devices running on Windows 8.	

Requirements regarding performance:

CNR10	*must have*
Average waiting time between submitting input and receiving output is no longer than 5 seconds.	
CNR11	*should have*
Average waiting time between submitting input and receiving output is no longer than 3 seconds.	
CNR12	*could have*
Average waiting time between submitting input and receiving output is no longer than 1 second.	

Requirements regarding extendability:

CNR13 *must have*
The product should be extendable with new mixers.

Appendices

Appendix A

Use case diagram

Figure A.1 of this Appendix contains the use case diagram. All use-cases described in Appendix B are included in this diagram. The following abbreviations are used:

C.D. Concentration Distribution.

M.P. Mixing Protocol.

M.R. Mixing Run.

Furthermore, solid arrows represent the *Preceeds* relationship, and dashed arrows represent the *Uses* relationship.

Figure A.1: Use Case Diagram including all the use cases described in Appendix B.

Appendix B

Use cases

This appendix lists all use cases for the FINGERPAINT application.

B.1 Define a mixing geometry and mixer

Goals: To define a mixing geometry and mixer.
Preconditions: none.
Summary: The user selects the geometry used for the mixing process.
Priority: *Must have* for rectangle, *should have* for square and *could have* for circle and *Journal Bearing*.
Steps:

Actor actions:	FINGERPAINT *response:*
1. Taps the *Start Mixing* button.	2. Displays the mixing interface.
3. Selects a mixing geometry of choice from the mixing interface (rectangle, square, circle or *Journal Bearing*).	4. Displays all mixers associated with the chosen mixing geometry.
5. Selects a mixer of choice.	6. Displays a white initial concentration distribution canvas, conform chosen mixing geometry and mixer.

B.2 Load a previously saved concentration distribution

Goals: To load a previously saved concentration distribution.
Preconditions: none.
Summary: The user loads a previously saved concentration distribution.
Priority: *Should have.*
Steps:

Actor actions:	FINGERPAINT *response:*
1. Actor actions of **B.1**.	2. FINGERPAINT responses of **B.1**.
3. Taps the *Load Saved Distribution* button.	4. Displays the previously saved concentration distributions for the selected geometry.
5. Taps on the concentration distribution of choice	6. Displays the canvas with the selected concentration distribution.

Alternative:

Actor actions:	FINGERPAINT *response:*
	4. Displays a *No Saved Distributions* message.
5. Taps the *OK* button.	6. Displays the mixing interface again.

B.3 Load a predefined concentration distribution

Goals: To load a predefined concentration distribution.
Preconditions: none.
Summary: The user loads a predefined concentration distribution.
Priority: *Could have.*
Steps:

Actor actions:	FINGERPAINT *response:*
1. Actor actions of **B.1**.	2. FINGERPAINT responses of **B.1**.
3. Taps the *Load Predefined Distribution* button.	4. Displays the predefined concentration distributions for the selected geometry.
5. Taps on the predefined concentration distribution of choice	6. Displays the canvas with the selected concentration distribution.

B.4 Define an initial concentration distribution

Goals: To define an initial concentration distribution.
Preconditions: None.
Summary: The user defines the initial concentration distribution
Priority: *Must have.*
Steps:

Actor actions:	FINGERPAINT *response:*
1. Actor actions of **B.1** or **B.2** or **B.3**.	2. FINGERPAINT responses of **B.1** or **B.2** or **B.3**.
3. Selects black or white as color.	4. Gives visual feedback on the selected colour.
5. Moves their finger across the screen to define the initial concentration distribution.	6. Gives real-time visual feedback of the selected area in the selected colour.
7. Repeats step 3 & 5 until satisfied.	8. Repeats step 4 & 6 accordingly.

B.5 Save concentration distribution

Goals: To save a concentration distribution.
Preconditions: None.
Summary: The user saves the concentration distribution
Priority: *Should have.*
Steps:

Actor actions:	FINGERPAINT *response:*
1. Actor actions of **B.4**.	2. FINGERPAINT responses of **B.4**.
3. Taps the *Save Distribution* button.	
4. Chooses a name for the concentration distribution.	
5. Taps the *Save* button.	6. Displays a *Save was succesfull* message.
7. Taps the *OK* button.	8. Displays the mixing interface again.

Alternative S.1:

Actor actions:	FINGERPAINT *response:*
	8. Displays an *Insufficient access rights* message.
9. Taps the *OK* button.	10. Displays the mixing interface again.

Alternative S.2:

Actor actions:	FINGERPAINT *response:*
	8. Displays an *Insufficient memory* message.
9. Taps the *OK* button.	10. Displays the mixing interface again.

Alternative S.3.1:

Actor actions:	FINGERPAINT *response:*
	8. Displays a *Name already in use* message with *Overwrite* and *Cancel* options.
9. Taps the *Overwrite* button.	10. Displays a *Save was succesfull* message.
11. Taps the *OK* button.	12. Displays the mixing interface again.

Alternative S.3.2:

Actor actions:	FINGERPAINT *response:*
	8. Displays a *Name already in use* message with *Overwrite* and *Cancel* options.
9. Taps the *Cancel* button.	10. Displays the mixing interface again.

B.6 Load a previously saved mixing protocol

Goals: To load a previously saved mixing protocol.
Preconditions: *Intermediate Steps* has **not** been ticked.
Summary: The user loads a previously saved mixing protocol.
Priority: *Should have.*
Steps:

Actor actions:	FINGERPAINT *response:*
1. Actor actions of **B.4** or **B.5**.	2. FINGERPAINT responses of **B.4** or **B.5**.
3. Taps the *Load Saved Protocol* button.	4. Displays the previously saved mixing protocols for the selected geometry.
5. Taps on the mixing protocol of choice.	6. Displays the selected mixing protocol.

Alternative:

Actor actions:	FINGERPAINT *response:*
	4. Displays a *No Saved Protocols* message.
5. Taps the *OK* button.	6. Displays the mixing interface again.

B.7 Load a predefined mixing protocol

Goals: To load a predefined mixing protocol.
Preconditions: none.
Summary: The user loads a predefined mixing protocol.
Priority: *Could have.*
Steps:

Actor actions:	FINGERPAINT *response:*
1. Actor actions of **B.4** or **B.5**.	2. FINGERPAINT responses of **B.4** or **B.5**.
3. Taps the *Load Predefined Distribution* button.	4. Displays the predefined mixing protocols for the selected geometry.
5. Taps on the predefined mixing protocol of choice	6. Displays the selected mixing protocol.

Alternative:

Actor actions:	FINGERPAINT *response:*
	4. Displays a *No Saved Protocols* message.
5. Taps the *OK* button.	6. Displays the mixing interface again.

B.8 Define mixing protocol (1)

Goals: To define the mixing protocol
Preconditions: None.
Summary: The user defines the mixing protocol
Priority: *Must have.*
Steps:

Actor actions:	FINGERPAINT *response:*
1. Actor actions of **B.4** or **B.5** or **B.6** or **B.7**.	2. FINGERPAINT responses of **B.4** or **B.5** or **B.6** or **B.7**.
3. Selects a value for step (D).	4. Gives visual feedback on the step (D) that has been selected
5. Swipes his/her finger across the screen to indicate the movement of the geometry.	6. Displays which movement has been selected.
7.The user does or does not tick the *Intermediate Steps* tickbox.	8. Displays result in checkbox.
9. Repeats step 3, 5 & 7 until satisfactory parameters have been selected	10. Repeats step 4, 6 & 8 accordingly. The most recent parameter value is applied.

Remark: Steps 3-4 , 5-6 and 7-8 can also be executed in reverse order.

B.9 Define mixing protocol (2)

Goals: To define the mixing protocol
Preconditions: *Intermediate Steps* has been ticked.
Summary: The user defines the mixing protocol
Priority: *Must have.*
Steps:

Actor actions:	FINGERPAINT *response:*
1. Actor actions of **B.8**.	2. FINGERPAINT responses of **B.8**.
3. Taps the *Mix Now* button	4. Disables the *Intermediate Steps* checkbox (it can not be edited anymore). Adds the movement to the protocol-log. Computes the result of applying the given movement to the distribution and displays it on the canvas.
5. Repeats steps at **B.4** (only 3, 5 & 7), **B.8** (only 3, 5, 7 & 9) and (3) until satisfied.	6. Repeats steps at **B.4** (only 4, 6 & 8), **B.8** (only 4, 6, 8 & 10) and (4).

Alternative:

Actor actions:	FINGERPAINT *response:*
	4. Adds the movement to the protocol-log. Fails to connect to server. Displays a *Connection Failed* message.
5. Taps the *OK* button.	6. Displays the mixing interface again.

B.10 Define mixing protocol (3)

Goals: To define the mixing protocol
Preconditions: *Intermediate Steps* has **not** been ticked.
Summary: The user defines the mixing protocol
Priority: Must have.
Steps:

Actor actions:	FINGERPAINT *response:*
1. Actor actions of **B.8**.	2. FINGERPAINT responses of **B.8**.
3. Taps the *Add To Protocol* button	4. Disables the *Intermediate Steps* checkbox (it can not be edited anymore). Adds the selected movement to protocol-log.
5. Repeats steps at **B.8** (3, 5, 7 & 9) and (3) until satisfied	6. Repeats steps at **B.8** (4, 6, 8 & 10) and (4).
7. Taps on the *Mix Now* button	8. Computes the result of applying all movements in the protocol-log to the initial concentration distribution, and displays it on the canvas.

Alternative:

Actor actions:	FINGERPAINT *response:*
	8. Fails to connect to server. Displays a *Connection Failed* message.
5. Taps the *OK* button.	6. Displays the mixing interface again.

B.11 Save mixing run

Goals: To save a mixing run.
Preconditions: None.
Summary: The user saves the mixing run.
Priority: *Should have.*
Steps:

Actor actions:	FINGERPAINT *response:*
1. Actor actions of **B.9** or **B.10**.	2. FINGERPAINT responses of **B.9** or **B.10**.
3. Taps the *Save Run* button.	
4. Chooses a name for the run.	
5. Taps the *Save* button.	6. Displays a *Save was succesfull* message.
7. Taps the *OK* button.	8. Displays the mixing interface again.

Alternatives: Alternatives **S.1, S.2, S.3.1 and S.3.2** are applicable here.

B.12 Save mixing protocol

Goals: To save a mixing protocol.
Preconditions: None.
Summary: The user saves the mixing protocol.
Priority: Should have.
Steps:

Actor actions:	FINGERPAINT *response:*
1. Actor actions of **B.9** or **B.10**.	2. FINGERPAINT responses of **B.9** or **B.10**.
3. Taps the *Save Protocol* button.	
4. Chooses a name for the protocol.	
5. Taps the *Save* button.	6. Displays a *Save was succesfull* message.
7. Taps the *OK* button.	8. Displays the mixing interface again.

Alternatives: Alternatives **S.1, S.2, S.3.1 and S.3.2** are applicable here.

B.13 View previously saved mixing run

Goals: To view the details of a previously saved mixing run.
Preconditions: None.
Summary: The performance of the selected mixing run is shown, accompanied by an image of the final result.
Priority: *Should have.*
Steps:

Actor actions:	FINGERPAINT *response:*
1. Taps the *View history* button.	2. Displays the history interface.
3. Selects the *View mixing run details.*	4. Displays all previously saved runs.
5. Taps one of the runs shown.	6. Displays the performance result of the selected run and an image of the final mixing result.

Alternative 1:

Actor actions:	FINGERPAINT *response:*
	4. Displays an *Insufficient access rights* error message.
5. Taps the *OK* button.	6. Displays the history interface again.

Alternative 2:

Actor actions:	FINGERPAINT *response:*
	4. Displays a *No Saved Runs* message.
5. Taps the *OK* button.	6. Displays the history interface again.

B.14 Export mixing run image

Goals: To export the resulting image of a mixing run.
Preconditions: None.
Summary: The image of the executed mixing run is stored locally on the user's device.
Priority: *Should have.*
Steps:

Actor actions:	FINGERPAINT *response:*
1. Actor actions of **B.9** or **B.10** or **B.13**.	2. FINGERPAINT responses of **B.9** or **B.10** or **B.13**.
3. Selects the *Export Image* option.	4. Displays the *Save* interface.
5. Selects a location on his/her device to save the image.	
6. Chooses a name for the image.	
7. Taps the *Save* button.	8. Displays a *Save was succesfull* message.
9. Taps the *OK* button.	10. Displays the mixing interface again.

Alternatives: Alternatives **S.1, S.2, S.3.1 and S.3.2** are applicable here.

B.15 Export mixing run performance graph

Goals: To export the performance graph of a mixing run.
Preconditions: None.
Summary: The performance graph of the executed mixing run is stored locally on the user's device.
Priority: *Should have.*
Steps:

Actor actions:	FINGERPAINT *response:*
1. Actor actions of **B.9** or **B.10** or **B.13**.	2. FINGERPAINT responses of **B.9** or **B.10** or **B.13**.
3. Selects the *Export Performance* option.	4. Displays the *Save* interface.
5. Selects a location on his/her device to save the performance graph.	
6. Chooses a name for the performance graph.	
7. Taps the *Save* button.	8. Displays a *Save was succesfull* message.
9. Taps the *OK* button.	10. Displays the mixing interface again.

Alternatives: Alternatives **S.1, S.2, S.3.1 and S.3.2** are applicable here.

B.16 Export mixing run animation

Goals: To export the resulting animation of a mixing run.
Preconditions: None.
Summary: The animation of the executed mixing run is stored locally on the user's device.
Priority: *Could have.*
Steps:

Actor actions:	FINGERPAINT *response:*
1. Actor actions of **B.9** or **B.10** or **B.13**.	2. FINGERPAINT responses of **B.9** or **B.10** or **B.13**.
3. Selects the *Export Animation* option.	4. Displays the *Save* interface.
5. Selects a location on his/her device to save the animation.	
6. Chooses a name for the animation.	
7. Taps the *Save* button.	8. Displays a *Save was succesfull* message.
9. Taps the *OK* button.	10. Displays the mixing interface again.

Alternatives: Alternatives **S.1, S.2, S.3.1 and S.3.2** are applicable here.

B.17 View multiple mixing performance results from previous mixing runs

Goals: To view the mixing performance of multiple mixing runs in the same graph.
Preconditions: None.
Summary: The performance of the selected mixing runs are shown in the same graph.
Priority: *Should have.*
Steps:

Actor actions:	FINGERPAINT *response:*
1. Taps the *View history* button.	2. Displays the history interface.
3. Selects the *View performance* option.	4. Displays all previously saved mixing runs.
5. Selects two or more mixing runs from the list.	
6. The user taps the *Submit* button.	7. Displays the mixing performances of the selected mixing runs in one graph.

Alternative 1:

Actor actions:	FINGERPAINT *response:*
	4. Displays an *Insufficient access rights* error message.
5. Taps the *OK* button.	6. Displays the history interface again.

Alternative 2:

Actor actions:	FINGERPAINT *response:*
	4. Displays a *No Saved Runs* message.
5. Taps the *OK* button.	6. Displays the history interface again.

B.18 Export performance graph of multiple mixing runs

Goals: To export the performance graph of multiple mixing runs.
Preconditions: None.
Summary: The performance graph of multiple mixing runs is stored locally on the user's device.
Priority: *Should have.*
Steps:

Actor actions:	FINGERPAINT *response:*
1. Actor actions of **B.17**.	2. FINGERPAINT responses of **B.17**.
3. Selects the *Export Graph* option.	4. Displays the *Save* interface.
5. Selects a location on his/her device to save the performance graph.	
6. Chooses a name for the performance graph.	
7. Taps the *Save* button.	8. Displays a *Save was succesfull* message.
9. Taps the *OK* button.	10. Displays the mixing interface again.

Alternatives: Alternatives **S.1, S.2, S.3.1 and S.3.2** are applicable here.

B.19 Remove mixing run from history

Goals: To remove the result of a mixing run.
Preconditions: None.
Summary: The selected mixing run is removed from the history.
Priority: *Should have.*
Steps:

Actor actions:	FINGERPAINT *response:*
1. Taps the *View history* button.	2. Displays the history interface.
3. Selects the *View mixing run details.*	4. Displays all previously saved mixing runs.
5. Presses and holds one of the mixing runs shown.	6. Displays an options menu.
7. Taps the *Remove file* option.	8. Displays an *Are you sure?* message.
9. Taps the *Yes* button.	10. Deletes the selected mixing run from storage.
	11. Displays all previously saved mixing runs.

Alternative 1:

Actor actions:	FINGERPAINT *response:*
	4. Displays a *No Saved Runs* message.
5. Taps the *OK* button.	6. Displays the history interface again.

Alternative R.1:

Actor actions:	FINGERPAINT *response:*
	6. Displays an *Insufficient access rights* error message.
7. Taps the *OK* button.	8. Displays the history interface again.

Alternative R.2:

Actor actions:	FINGERPAINT *response:*
9. Taps the *No* button.	10. Displays all previously saved mixing runs.

B.20 Remove concentration distribution

Goals: To remove a concentration distribution.
Preconditions: None.
Summary: The selected concentration distribution is removed from local storage.
Priority: *Should have.*
Steps:

Actor actions:	FINGERPAINT *response:*
1. Taps the *View Distributions* button.	2. Displays all previously saved concentration distributions.
3. Presses and holds one of the concentration distributions shown.	4. Displays an options menu.
5. Taps the *Remove File* option.	6. Displays an *Are you sure?* message.
7. Taps the *Yes* button.	8. Deletes the selected concentration distribution from local storage. 9. Displays all previously concentration distributions.

Alternative 1:

Actor actions:	FINGERPAINT *response:*
	4. Displays a *No Saved Distributions* message.
5. Taps the *OK* button.	6. Displays the history interface again.

Alternatives: Alternatives **R.1 and R.2** are applicable here.

B.21 Remove mixing protocol

Goals: To remove a mixing protocol.
Preconditions: None.
Summary: The selected mixing protocol is removed from local storage.
Priority: *Should have.*
Steps:

Actor actions:	FINGERPAINT *response:*
1. Taps the *View Protocols* button.	2. Displays all previously saved mixing protocols.
3. Presses and holds one of the mixing protocol shown.	4. Displays an options menu.
5. Taps the *Remove File* option.	6. Displays an *Are you sure?* message.
7. Taps the *Yes* button.	8. Deletes the selected mixing protocol from local storage. 9. Displays all previously mixing protocols.

Alternative 1:

Actor actions:	FINGERPAINT *response:*
	4. Displays a *No Saved Protocols* message.
5. Taps the *OK* button.	6. Displays the history interface again.

Alternatives: Alternatives **R.1 and R.2** are applicable here.