Software Requirements Document

Flutter smart sensing library
for medical and psychological study apps

edited by

Leonhard Alkewitz, Florian Gebhardt, Hermann Fröhlich,
Mukhtar Muse, Felix Schlegel

Contents

1 Introduction

Study apps can be helpful for the successful conduct of medical and psychological studies. Especially the use of sensors, for example from smartphones or wearables, can provide useful data to support the study. Therefore, the development of such apps should be as simple as possible in order to be able to use them without great effort. That's why we develop a smart sensing library, which can be used to easily implement cross-platform apps with Flutter.
This library will be able to track sensors from the smartphone and some wearables, collect the sensor data and save it. Additionally, it will be possible to either get live data or store and process the data.

2 Definitions

2.1 Glossary

2.2 Actors

Actor:	developer
Description:	*developer* means a person, who develop a study app with the library (which is the prodcut of this project).
Representative:	The development team creating this document assumes the role of a *developer* when creating the demo app (c. f. 3.5)

Actor:	end-user
Description:	*end-user* means a person, who uses a study app, which was implemented with the library (which is the prodcut of this project).
Representative:	The development team creating this document or the stakeholder assumes this role when testing the *Demo App*(c. f. 3.5).

Actor:	user
Description:	In most cases it makes no difference, whether the requirement is motivated by the end-user or the developer, because the developer only displays the information of the library requested by the end-user or delegates the input of the end-user. Therefore, we speak of *user*, if we want to express, that the developer as well as the end-user motivate the requirement.
Representative:	

3 Functional requirements

3.1 Management of devices

3.1.1 Smartphone

	get smartphone information **FR1**
Description	The library must be able to get information about the smartphone on which the smart sensing app, which is developed with the library, is installed on. This information include at least • the current operation system (OS) and it's version (e. g. iOS 16.1) • the currently free storage on the smartphone
Precondition	The end-user must allow the app to get the smartphone information. If the end-user deny, the library must provide a hint
Rationale	The library must collect data about the current smartphone to figure out, which sensors are available and which implementation to use (e. g. based on current OS). But the end-user must have the control, which data the app respectively the library collect.
Priority	high

	add smartphone sensors **FR2**
Description	The library must provide the opportunity to the user to add all sensors available on the smartphone to a list of usable sensors.
Precondition	The end-user must allow the library to discover all sensor by getting the smartphone information and the library got the information about the smartphone.
Postcondition	All discovered and added sensors can be tracked from now on. The tracking can be started by „adding“ the sensors (c. f. FR5→ p. 4).
Rationale	There must be an option for the end-user to make the sensors on its smartphone traceable. Otherwise the end-user would not have the freedom to choose which sensors to use.
Dependencies	FR1→ p. 3
Priority	high

3.2 Organization of sensors

3.2.1 Adding sensor

	check precision **FR3**
Description	The library must check, whether the selected precision is valid. It is valid, if • it is an unsigned integer • it is between 0 and 10 (0 means, that the data is round to an integer using the standard rounding; 1-10 means the data is round to decimal with 1-10 fractional digits)
Rationale	If the precision is not an unsigned integer, it is not possible to round and if the number of digits to round is to big, the requested precision is higher than the accuracy of the sensors.
Priority	medium

	check time interval **FR4**
Description	The library must check, whether the selected time interval is valid. It is valid, if • the numbers are unsigned integers • all numbers are 0 (which means, the user want live data) • the third and fourth number is between 1 and 59 (the unit for time interval is seconds or minutes) • the second number is between 1 and 23 (the unit for time interval is hours) • the first number is between 1 and 7 (the unit for time interval is days)
Precondition	The user provided 4 numbers representing the time interval (days, hours, minutes, seconds).
Rationale	If the time interval is set to an invalid value, it is not possible to request data from the sensor, which is the main task of this library.
Priority	high

	add sensor for tracking **FR5**
Description	The library must provide an API to start the tracking of a sensor.
Precondition	The sensor, the user want to add, needs to be known to the library. This means this sensor is in a list of implemented sensors.
Postcondition	After start tracking the sensor, it will be added to a list of tracked or used sensors and the sensor data are accessible from this moment on.
Rationale	The user must have the opportunity to add sensors, so they will be tracked, because otherwise there is no option to control, which sensors from the devices will be tracked and which not.
Dependencies	FR2→ p. 3, FR3→ p. 4, FR4→ p. 4
Priority	high

	set unit of sensor data **FR6**
Description	In the process of adding a new sensor, the library must provide an opportunity for setting the unit, in which the sensor data should be made available.
Precondition	Therefor the library must be able to deal with different units and provide the user a set of units, it can deal with and end-user can select from.
Postcondition	From this moment on the data will be displayed in the selected unit.
Rationale	The user must have the opportunity to set the units, because different users are used to use different units (e. g. $°C$ vs. F for temperature)
Dependencies	FR5→ p. 4
Priority	low

	set precision of sensor data **FR7**
Description	In the process of adding a new sensor, the library must provide an opportunity for setting the precision, in which the sensor data should be made available. Therefor the library must provide an option, the end-user can set the number of fractional digits.
Postcondition	From the moment on, the end-user set the number, the sensor data will be round to the set number of fractional digits.
Rationale	The end-user must have the opportunity to set the precision, because different sensors provide data in different precision, but the end-user maybe want all data in the same precision.
Dependencies	FR5→ p. 4, FR3→ p. 4
Priority	medium

	set time interval of sensor data **FR8**
Description	In the process of adding a new sensor, the library must provide an opportunity for setting the time interval, in which the data should be „requested“ from the sensor respectively the sensor data provided.
Precondition	Therefor the library must be able to deal with different time units and provide the user the option to select a time units and input unsigned integers representing the time interval (c. f. FR4→ p. 4)
Postcondition	The unsigned integers combined with the time units will represent the time interval and the sensor data will be collected in this time interval.
Rationale	The end-user must have the opportunity to set the frequency of getting the data, because the different sensors do not produce data at the same frequency. Furthermore it is possible, that the end-user is not interested in getting the sensor data in same regularity for different sensors. For example the end-user only want to know the temperature once in the morning, but the average activation of the smartphone every hour.
Dependencies	FR5→ p. 4, FR4→ p. 4
Priority	high

3.2.2 Editing sensor

	edit properties of sensor **FR9**
Description	The library must provide the opportunity to change the following properties of the sensor, which must be set when adding the sensor: • the unit of the sensor data • the precision of the sensor data • the time interval, the sensor data is collected
Precondition	The new values set must be valid and the sensor, whose specifications will be changed, must be currently tracked.
Postcondition	The sensor data will be provided corresponding to the new values set.
Rationale	The end-user must have the opportunity to change the properties of the sensor, because end-users can make mistakes when input values and that's why input must be editable.
Dependencies	FR3→ p. 4, FR4→ p. 4, FR5→ p. 4
Priority	medium

3.3 Getting sensor list

	get sensors	**FR10**
Description	The library must store a list of all sensors, which can be started or are currently tracked. Furthermore this list can be returned on demand.	
Rationale	There must be an option to track all sensors and their status, so the end-user can trace what is happening and search for sensors. Additionally the library must know all sensors for managing them.	
Dependencies	FR2→ p. 3, FR5→ p. 4	
Priority	high	

	get used sensors	**FR11**
Description	The library stores, which sensors are currently tracked and can return a list of all sensors, which are currently tracked.	
Rationale	There must be an option to get all running sensors for checking the status of a specific sensor, e. g. when stopping the tracking of a sensor.	
Dependencies	FR10→ p. 7	
Priority	low	

3.3.1 Removing sensor

	check sensor is used	**FR12**
Description	The library can check, whether a sensor is currently tracked, so was started previously. Therefor the user can provide a sensor name and the sensor, identified by this name, will be checked.	
Postcondition	The library return, whether a sensor is currently used (so tracked) or not.	
Rationale	To prevent bugs when removing or dealing with sensors, it is useful to be able to check, whether a sensor is started and can be stopped or edited.	
Dependencies	FR11→ p. 7	
Priority	high	

	stop sensor tracking **FR13**
Description	The library must provide an API for removing a sensor from the list of tracked sensors.
Precondition	The sensor, the end-user want to remove, needs to currently tracked, so started before and known to the library.
Postcondition	After „removing“ the sensor from the list, the sensor will no longer appear in the list of currently tracked sensors and the tracking of the sensor will be stopped. This means, that the sensor still exist and can be started again (c. f. FR5→ p. 4), but will not be requested for data anymore.
Rationale	The end-user must have the opportunity to stop tracking sensors, because otherwise there is no option to control, which sensors from the devices will be tracked and which not.
Dependencies	FR5→ p. 4, FR12→ p. 7
Priority	high

3.3.2 Search sensor

	search by name **FR14**
Description	The library must provide an API for searching a sensor by its sensor name. Therefor the API must accept a name (and optional the information, whether only currently tracked or all matching sensors should be returned). If matching sensors exist, they should be returned to the user.
Precondition	The provided sensor name is valid and matches any sensor, otherwise there cannot be a list returned.
Rationale	If the end-user want to get a specific sensor, searching by name is a simplification for the end-user because he do not read through all sensors
Priority	high

3.4 Management of data from sensor

3.4.1 Fetching data from sensor

	check sensor (data) is available **FR15**
Description	The library can check, whether a sensor is available respectively the sensor can provide data.
Precondition	The sensor was added and is currently running.
Rationale	Before data from a sensor is requested, this sensor must be available, because otherwise it cannot provide sensor data.
Dependencies	FR5→ p. 4
Priority	high

	fetch data from sensor **FR16**
Description	The library must provide an API, which is able to deal with a sensor name and fetch the data from the sensor, identified by the provided name.
Precondition	The provided sensor name is valid and the sensor, identified by the name, must be available, which must be checked before.
Rationale	The user must be able to get information of the sensors, because they not only collect data but also provide them.
Dependencies	FR15→ p. 9
Priority	high

3.4.2 Saving sensor data

	save live sensor data **FR17**
Description	The library must be able to save (live) data from a sensor into the local storage of the smartphone. Therefor the data will be saved in combination with at least the • name of the sensor providing this data • the unit the data are represented in • the timestamp, the data were produced
Precondition	The sensor, whose data should be saved, must be currently running and tracked, so it can provide data.
Postcondition	The data of the sensor will be permanently stored in the local storage and can be read later.
Rationale	It is necessary to be able to save data into a storage, because otherwise you cannot export data or get historic data.
Dependencies	FR15→ p. 9, FR16→ p. 9
Priority	high

	save high-frequent sensor data **FR18**
Description	The library must be able to deal with high-frequent data and save them. In this context, *high-frequent* means, that the sensor provides at least one data point in a second. This type of sensor data needs to be stored in an efficient way, so not every data point individually.
Precondition	The sensor, whose data should be saved, must be currently running and tracked, so it can provide data.
Postcondition	The data of the sensor will be permanently stored in the local storage and can be unambiguous read later.
Rationale	It is necessary to deal with high-frequent data, because they are produced by different sensors like the accelerometer. It is not possible to simply store them, because due to the amount of data points to size of the sensor data would be to large for the local storage or at least not acceptable.
Dependencies	FR15→ p. 9, FR16→ p. 9
Priority	high

	save processed sensor data	**FR19**
Description	If the end-user requested processed data, the data in combination with the query will be saved in the storage.	
Precondition	The query must be valid, so only containing known aggregate functions and retrieving methods from FR22→ p. 11, FR21→ p. 11, FR23→ p. 12	
Postcondition	If a end-user use the same query again, the data will not be aggregated again, but the stored values will be returned.	
Rationale	If requests are saved, the data do not need to be re-aggregated, if the end-user do the same request again.	
Dependencies	FR22→ p. 11, FR21→ p. 11, FR23→ p. 12	
Priority	low	

	remove sensor data	**FR20**
Description	The library must provide an API for deleting sensor data points. This means, the user can provide a time interval and a sensor name and all data points from the sensor, identified by the name, are deleted from the local storage.	
Precondition	The user need to provide a valid sensor name, whose data the user want to delete. Furthermore the user needs to provide a valid time interval and the sensor needs to have data points in this time interval.	
Postcondition	The requested data will be deleted permanently, so there is no chance to restore the data or to retrieve this data again.	
Rationale	The user must have the opportunity to delete sensor data because otherwise he would not have the control over the data.	
Dependencies	FR4→ p. 4	
Priority	high	

3.4.3 Retrieving sensor data

	retrieve sensor data by specific moment in time	**FR21**
Description	The library must provide an API, so the user can retrieve the data of a specific sensor at a specific moment in time.	
Precondition	The user need to provide a sensor, whose data the user want and the sensor needs to have a data point at this moment in time.	
Rationale	The user must have the opportunity to get historic data namely on a specific moment in time for later evaluation.	
Dependencies	FR16→ p. 9, FR17→ p. 10, FR18→ p. 10	
Priority	high	

	retrieve sensor data by time interval **FR22**
Description	The library must provide an API, so the user can retrieve the data between to specific moments in time. This means, that a user can provide a begin time and an end time respectively a time interval and the library return all collected data between this two times.
Precondition	The user needs to provide a valid sensor name, whose data the user want. Furthermore the user needs to provide two time stamps or a time stamp and time interval and the sensor needs to have data points in this time interval.
Rationale	The user must have the opportunity to get historic data namely a series of historic data in a certain time interval. This can be useful for later evaluation.
Dependencies	FR4→ p. 4, FR16→ p. 9, FR17→ p. 10, FR18→ p. 10

	apply aggregate functions on sensor data **FR23**
Description	The library must provide an API, so the user can apply aggregate functions on sensor data and retrieve the results. The supported aggregate functions will be • average / arithmetic mean • median • maximum • minimum • sum • count (prerequisite: the user also provide a specific value, which should be counted or nothing, so the amount of data points will be counted) • mode • range, so the distance between the minimum and the maximum • standard deviation All this function only make sense, if they are applied on a series of data points. The aggregate functions can be combined, if possible.
Precondition	The user need to provide a valid sensor name and two time stamps or a time interval and time stamp, which define the data set the user want to process. Furthermore the user needs to provide an aggregate function, which should be also applicable.
Postcondition	The requested data set will not be manipulated in the storage directly, but there will be a copy which will be manipulated and returned.
Rationale	The user must have the opportunity to process the sensor data, which is useful for later evaluation.
Dependencies	FR21→ p. 11, FR22→ p. 11, FR4→ p. 4
Priority	high

	combine queries **FR24**
Description	A query, so a request against the library for sensor data of a specific sensor within a specific time interval or on a specific moment in time (maybe with applied aggregate functions) should be applicable to multiple sensors. This means, if the developer provide a set of sensor names, all sensor identified by the names, will get the same provided query. The response of every sensor is bundled into on response.
Precondition	The developer need to provide valid sensor names. Furthermore the developer needs to provide two time stamps or a time interval and time stamp and the sensors needs to have data points in this time interval.
Rationale	When working with the library it can be useful, if you do not need to do the same request for every sensor you are interested in, but use one query on multiple sensors and get a bundle of responses to work with. This is easier for working with the library and can be helpful if you want to compare different sensor data.
Dependencies	FR21→ p. 11, FR22→ p. 11, FR23→ p. 12
Priority	medium

3.4.4 Importing and exporting sensor data

	specify import / export format **FR25**
Description	The library must provide an opportunity to set the format, in which the sensor data can be imported or will be exported.
Precondition	The specified format must be supported by the library.
Postcondition	If a end-user want to import or export data, this will be done in the specified format as long as the end-user do not select another format.
Rationale	The specification of this format allows different end-users to use different formats depending on their preferences.
Priority	medium

	import sensor data **FR26**
Description	The library must provide an opportunity to import sensor data satisfying a specific format.
Precondition	The import format (c. f. FR25→ p. 14) must be supported by the library.
Postcondition	The imported data will be added to the local storage (database) and can be requested by the user from this moment on.
Rationale	The end-user maybe want to import data, which were collected earlier and exported before and should be re-imported now.
Dependencies	FR25→ p. 14
Priority	medium

	export sensor data **FR27**
Description	The library must provide an opportunity to export sensor data from the local storage in a specific format. Therefor the end-user must be able to select a format, two timestamps and a sensor name. The library must export all data from the sensor, identified by the name, which are between the start and the end time stamp.
Precondition	The import format (c. f. FR25→ p. 14) must be supported by the library and the input values must be valid.
Postcondition	The exported data will be on the target volume in the specified format.
Rationale	The end-user maybe want to use the data beyond the app or want archive the data externally.
Dependencies	FR25→ p. 14
Priority	medium

3.5 Demo app

Additionally, the library respectively the smart sensing plugin should be tested. This can be done by implementing a demo app using the library. In general the demo app must be able to demonstrate the successful implementation of all previously listed requirements (c. f. 3.1 - 3.4). That means, the demo app should ...

- ... be able to get the information about the smartphone it is currently running on and display the information or a hint or error, that getting this information was not possible.
- ... select a specific sensor and start tracking the sensor
- ... select a specific sensor and edit it properties, so change the unit, precision, time interval.
- ... select a specific sensor and stop tracking the sensor
- ... search for a specific sensor
- ... select a specific sensor and display the live data from this sensor
- ... select a specific sensor and display processed or historical data from this sensor

4 Non-functional requirements