

---

---

# REQUIREMENTS ENGINEERING

---

---

FLUTTER SMART SENSING LIBRARY  
FOR MEDICAL AND PSYCHOLOGICAL STUDY APPS

EDITED BY

LEONHARD ALKEWITZ, FLORIAN GEBHARDT, HERMANN FRÖHLICH,  
MUKHTAR MUSE, FELIX SCHLEGEL

# Contents

<b>1</b>	<b>Adding a new Feature/User Story</b>	<b>2</b>
1.1	Creating a new User Story . . . . .	2
1.2	Adding a new Feature . . . . .	2
1.2.1	Creating an Issue . . . . .	3
1.2.2	Reviewing an Issue . . . . .	5
1.2.3	Weight of Issues . . . . .	5
<b>2</b>	<b>Decisions about adding/removing features or user stories</b>	<b>6</b>

## 1.1 Creating a new User Story

It is important to remember that integrating a new user story into a map may also require changes to other parts of the system. It is therefore important to carefully plan and coordinate the integration of new functionality to ensure that it fits seamlessly into the existing system.

## 1.2 Adding a new Feature

To add a new feature, the whole process takes place within our GitLab environment. You can then create an issue. An issue is used to describe the new feature or desired functionality in terms of a specific task that needs to be implemented.

After the issue is created, a separate feature branch is created for the feature. All changes and developments take place in this separate feature branch to isolate the development of the feature from other changes in the main branch. This allows developers to work on the new feature without affecting the stability of the main branch and ensures that the new feature can be thoroughly tested before being merged into the main branch.

### 1.2.1 Creating an Issue


When creating the issue, you should use the appropriate template to clearly define the objective of the feature.

For example, if you want to add a new sensor to the app, you can select the Issue Feature option and fill out the template with the relevant information. The template includes sections for feature description, use case, benefits, requirements, and links/references.

1. In the feature description section, you can explain what type of sensor you want to add and how it should function.
2. In the use case section, you can describe how the new sensor will be used by users. In the benefits section, you can explain how the new sensor will improve the app and provide value to users.
3. In the requirements section, you can list any technical or functional requirements that must be met for the new sensor to work properly.
4. Finally, in the links/references section, you can provide any relevant links or references that may help developers understand your requirements and implement the new feature accordingly.

#### New Issue






Title (required)

Type 

Issue

Description

Feature-issue

**B** *I* U ~~ABC~~ `<>`     

```
## Feature description
<!-- Provide a general description of the issue -->


## Use cases
<!-- Tell us the use case from this feature -->


## Benefits
<!-- Tell us the Benefits from this feature -->
<!-- Who can benefit from this feature and why ? -->

## Requirements
<!-- Tell us the requirement to use this feature -->

## Links / references
<!-- If there is a link, u should always write it down -->

/label ~"type::feature"
/cc @npz16
/assign_reviewer @kjoy97
```

Preview 

Supports Markdown. For quick actions, type .

In addition to this, you should consider who should be selected as the assignee for the issue, be it yourself or another person responsible for implementing the feature. Assigning an assignee clearly establishes responsibility. To further specify the issue and place it in the overall context of the project, you can define an Epic and a Milestone, if available.

An Epic summarises related features or tasks and allows for better organisation and prioritisation. A milestone helps track the progress of the project and mark important stages.

In addition, you can add a predefined label to further mark the issue and make it easier to identify. This makes it easier to filter and categorise issues within the project management system.

To plan and track the progress of the issue, it is useful to set a date by which the issue should be completed. You should also specify the sprint iteration in which the issue is to be worked on. This allows for effective planning and coordination of workflows within the development team.

By including this additional information and details, the issue becomes more comprehensive and structured. This facilitates communication, planning and progress of the feature in the project.

<b>Assignees</b>	<b>Weight</b>
<div>Unassigned</div>	<div>Enter a number</div>
<b>Epic</b>	<b>Due date</b>
<div>Select epic</div>	<div>Select due date</div>
<b>Milestone</b>	<b>Iteration</b>
<div>Select milestone</div>	<div>Select iteration</div>
<b>Labels</b>	
<div>Labels</div>	

### 1.2.2 Reviewing an Issue

As soon as the development of the feature is completed, a so-called merge request is created.

In this merge request, the completed issue is submitted for review by another developer.

The reviewer checks the code, the functionality and the implementation of the feature.

In the process, possible problems, improvement requests or questions are discussed in the form of threads within the merge request.

The merge request remains open until all threads have been successfully processed and resolved.

Once the reviewer is satisfied and all open issues are resolved, he can approve the merge request.

This merges the issue into the main branch and integrates the new feature into the overall system.

This structured approach ensures that all changes and developments are thoroughly reviewed and discussed before they enter the main branch. This ensures the quality and stability of the system and potential errors or inconsistencies can be identified and corrected at an early stage.

All in all, each feature is developed in its own feature branch, goes through the merge request process with reviews and discussions before it is finally merged into the main branch. In this way, we ensure effective collaboration and controlled integration of new features into the overall system.

### 1.2.3 Weight of Issues

The weighting of an issue is done in our regular sprint meetings, which are held every two weeks. In these meetings, we discuss together how an issue should be weighted. The weight of an issue indicates the estimated effort required to complete it within a sprint. We use a scale of 1 to 5, with a value of 1 representing the lowest and least effort, while a value of 5 signals the highest and greatest effort.

Through this structured approach, we can estimate the workload for each sprint and ensure that new features and user stories are integrated into our system effectively and professionally. This allows us to continuously improve the user experience and meet the needs and requirements of our users.

**Resolve "specify import / export format"**

Leonhard Alkewitz requested to merge feat/39-specify-import-exp... into main 1 month ago

Overview (45) Commits (16) Pipelines (9) Changes (7) All threads resolved!

**What this MR does / why we need it:**

- add a component for import / export and enables the developer using the library to select, which format should be used for
  - import
  - export
- add a description which formats are supported and how they are defined exactly
- created example sensor data files for all file formats (can be used for testing later)
- created validation schemas for XML and JSON and tested them with example sensor data files

**Make sure that you've checked the box below before you submit MR:**

- ☒ Rule conforming documented\*
- ☒ Style conforming code\*
- ☒ Tests were created, run successfully and are complete
- ☐ (optional) Additional information documented (in Wiki, README, ...)
- ☐ (optional) User story was tested

**Which platform/language**

- ☒ Flutter
- ☐ Android
- ☐ iOS

**Which issue this MR fixes (optional)**

- #39 (closed)

**CHANGELOG/Release Notes (optional)**

\*For more information check out the wiki: <https://gitlab.uni-ulm.de/groups/se/-/amendungsprojekt-22-23/-/wiki/Conventions&documentation>

Closes #39 (closed)

Edited 1 month ago by Leonhard Alkewitz

👍 0 🙋 0 🔄 0

Pipeline #16050 passed for 5a13f027 on feat/39-specify-import-export-format 1 month ago  
Test coverage 90.77% (0.23%) from 1 job

> View 0 exposed artifacts

8~ Approved by [Avatar]

Merged by Leonhard Alkewitz 1 month ago Revert Cherry-pick

**Merge details**

- Changes merged into main with a70b0c0d
- Deleted the source branch.
- Closed #39 (closed)

## 2 Decisions about adding/removing features or user stories

When it comes to adding or removing a feature, this is also discussed in the sprint meeting and then decided. It happens that certain features have to be removed, either because they cannot be implemented within the given time frame or due to a lack of capacity. In such cases, we may need to prioritize other features and remove those that are less important or require more effort to implement. It is the same with the integration of features. We have found that in some cases we were missing certain functionalities of which we had no prior knowledge and which had to be added subsequently.

Through this structured approach, we can effectively manage our workload and ensure that we are able to deliver high-quality features within the given time frame.