



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826078.

Privacy preserving federated machine learning and blockchaining for reduced cyber risks in a world of distributed healthcare



Deliverable D7.4
“User interfaces usable for patients, project managers, and developers”

Work Package WP7
“Integrated FeatureCloud health informatics platform and app store”

Disclaimer

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826078. Any dissemination of results reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

Copyright message

© FeatureCloud Consortium, 2022

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both. Reproduction is authorised provided the source is acknowledged.

Document information

Grant Agreement Number: 826078		Acronym: FeatureCloud	
Full title	Privacy preserving federated machine learning and blockchaining for reduced cyber risks in a world of distributed healthcare		
Topic	Toolkit for assessing and reducing cyber risks in hospitals and care centres to protect privacy/data/infrastructures		
Funding scheme	RIA - Research and Innovation action		
Start Date	1 January 2019	Duration	60 months
Project URL	https://featurecloud.eu/		
EU Project Officer	Christos MARAMIS, Health and Digital Executive Agency (HaDEA) - Established by the European Commission, Unit HaDEA.A.3 – Health Research		
Project Coordinator	Jan BAUMBACH, UNIVERSITY OF HAMBURG (UHAM)		
Deliverable	D7.4 “User interfaces usable for patients, project managers, and developers”		
Work Package	WP7 “Integrated FeatureCloud health informatics platform and app store”		
Date of Delivery	Contractual	31/12/2022 (M48)	Actual 19/12/2022 (M48)
Nature	Demonstrator	Dissemination Level	Public
Lead Beneficiary	01 UHAM		
Responsible Author(s)	Mohammad Bakhtiari (UHAM), Julian Alexander Späth (UHAM), Sándor Fejér (GND)		
Keywords	User-Interface, Federated Collaboration, App development		

Table of Content

1	Objectives of the deliverable based on the Description of Action (DoA)	5
2	Executive Summary	5
3	Introduction (Challenge)	5
4	Methodology	5
4.1	Frontend technologies	6
4.1.1	TypeScript/Angular	6
4.1.2	Bulma	6
4.1.3	Additional packages	6
4.2	Backend technologies	6
4.2.1	Python/Django	6
4.2.2	Golang	6
5	Results	7
5.1	User interface for app developers	7
5.1.1	User Registration	7
5.1.2	Testing apps using test UI	10
5.1.3	Command Line Interface (CLI)	13
5.1.4	Creating an app	14
5.1.5	Providing feedback	15
5.2	User interface for project managers	17
5.2.1	Registration and user Account	17
5.2.2	The FeatureCloud app store and personal app library	18
5.2.2.1	App Details	19
5.2.3	Creating a project and assembling a workflow	20
5.2.4	Inviting participants and joining projects	20
5.2.5	Choosing data and starting a workflow	21
5.2.6	Workflow progress, logs, and results	22
5.2.7	Providing feedback	22
5.3	User Interface for patients	22
5.3.1	Submitting a consent	22
6	Open issues	25
7	Deviations (if applicable)	25
8	Conclusion	25

9	References	25
10	Table of acronyms and definitions	26
11	Other supporting documents / figures / tables (if applicable)	26
11.1	User registration and Login	26
11.2	CLI: starting the controller on local machine	27
11.3	App details	30
11.4	Running a workflow	34
11.5	Patients UI	35

1 Objectives of the deliverable based on the Description of Action (DoA)

D7.4, “User interfaces usable for patients, project managers, and developers,” is related to task 3, “User interfaces and testing” of WP7 as described in the Description of Action. We set up intuitive web-based human-computer interfaces for patients (right management) as well as for medical doctors and hospitals (workflow management) and developers (registration and testing of new apps). D7.4 also touches on task 1, “Programming interfaces and platform,” on providing a user interface (UI) for app developers, and task 2, “App store and workflow management,” to allow the community to plugin federated machine learning methods.

2 Executive Summary

This deliverable covers user interfaces developed for three categories of users: developers, project managers, and patients. For app developers, the convenience of using a platform to design, develop and test their applications is covered (see section 5.1). Project managers who want to plug in provided apps to conduct federated collaboration are covered as well (see section 5.2). These two groups of users will interact with the FeatureCloud front-end and share some UIs, including registration (see sections 5.1.1 and 11.1) and app store (see sections 5.2.2 and 11.3). The patients will use a UI that should be deployed on hospitals’ platforms to manage their consent through the blockchain consent management system. In this context, the foundation has been laid to allow for the integration of mechanisms developed by WP2 to capture and audit patients’ consent for data used in a federated collaboration (see section 5.3).

D7.2 already provided the basic functionality to develop apps, distribute them using the app store and execute them using the FeatureCloud controller. These functionalities are extended and supported in the UIs. D7.3 extended the app execution in the testbed while D7.4 extended the options using the command line interface to run the controller, tests, and app deployment (see section 5.1.3).

3 Introduction (Challenge)

A variety of user groups have the chance to utilize the FeatureCloud platform from different perspectives, including patients, for consent management, project managers, for federated collaboration, and developers, to implement applications for the app store. FeatureCloud provides common UIs with similar utilities for all user groups, including user registrations UI. Also, some UIs are shared between two groups of users, for instance, the app store, which provides developers with the chance to promote their apps and project managers with the opportunity of assembling a desirable workflow based on the provided apps in the app store. Moreover, FeatureCloud extends the UIs to cover user-group-specific features, e.g., app registration page, and a consent management UI for developers and patients, respectively. In section 4, we cover technologies that were used to develop the UIs and connect them to other components like the controller, back-end, etc. In section 5, we demonstrate how different UIs are designed and which functionalities are supported in the FeatureCloud platform.

4 Methodology

This section describes the tools, frameworks, and packages that were used to implement FeatureCloud’s UIs.

4.1 Frontend technologies

4.1.1 TypeScript/Angular

For building the frontend UIs that are accessible on our platform featurecloud.ai, we used the Angular development platform (*Angular*, no date). Angular is a modern, state-of-the-art web development platform built on TypeScript (*JavaScript with syntax for types*, no date). TypeScript is a strongly typed programming language that builds on JavaScript.

4.1.2 Bulma

For the design of the UIs, we used the Bulma framework (*Bulma: Free, open source, and modern CSS framework based on Flexbox*, no date). Bulma is a free and open-source CSS framework that provides various ready-to-use front-end components, styles, and templates that can be used to build modern UIs.

4.1.3 Additional packages

Table 1: Packages used for our Angular Frontend

Package	Description
fontawesome(<i>Font awesome</i> , no date)	Icons
ng-util/markdown(<i>Npm: @ng-util/markdown</i> , no date)	Markdown editor
bulma-toast(<i>Npm: Bulma-toast</i> , no date)	Toast messages
ng5-slider(<i>npm: ng5-slider</i> , no date)	Slider component
primeng(<i>PrimeNG</i> , no date)	Various components and styling

4.2 Backend technologies

Our frontends communicate to two different backend servers that were developed with two different technologies.

4.2.1. Python/Django

Our global backend, handling user login, apps, and less-sensitive information, is developed with the Python-Django (*Django*, no date) framework and deployed on a central FeatureCloud server.

4.2.2 Golang

The local backend, also called controller, is implemented in Go (Meyerson, 2014). The controller handles the local tasks, including preparing sensitive patient data.

5 Results

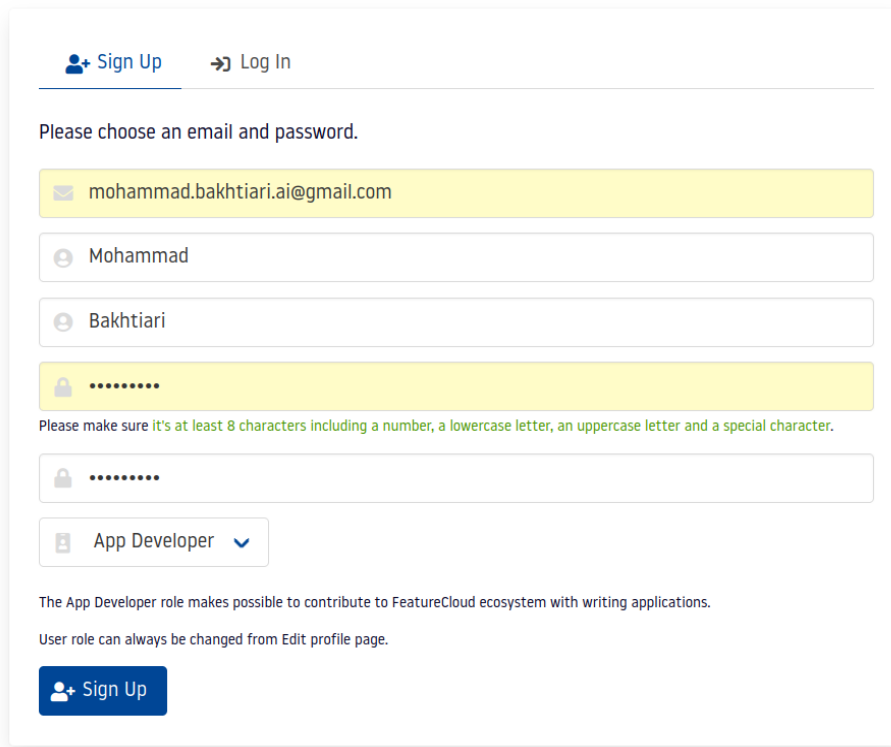
UHAM, in collaboration with other partners, has developed the FeatureCloud platform, which contains UIs for different types of users. We broadly categorize our UIs into three categories based on the users that can utilize them: patients, app developers, and project managers. The app developers have sufficient programming skills to develop federated applications using the FeatureCloud platform. They can use FeatureCloud UIs for testing and running workflows (WFs) using their applications. Project managers are hospital’s staff who manage WF by either assembling a WF and sharing the tokens with other partners to join or joining a WF. Unlike app developers, project managers will use the project UI to run a WF on real-world data, not testing apps. Patients are the last category of users to have their own UI to manage their consent. In the following sections, we elaborate on UIs provided in FeatureCloud based on the users who will utilize them.

5.1 User interface for app developers

App developers, such as skilled programmers, may use the FeatureCloud platform to develop applications and present those apps in the app store. In many cases, apps use Machine Learning (ML) or Deep Learning (DL) models for training models on the data and providing predictions. Later, project managers can use the apps in the app store to conduct research in a federated fashion. Like any other user in the FeatureCloud platform, app developers should sign up and create a virtual site to start experimenting with the platform. Once they want to develop an app, they should register it to be accessible in the app store and test it in the testbed to validate its results as a standalone federated app. Next, they should try their app in a series of possible WFs, in which their app can be used alongside other apps for further validation. App developers may use different UIs; some are also helpful for project managers, e.g., registration, projects, app store, workflow template, etc. In this section, we cover all designed UIs for app developers and explain for which purposes they can be used.

5.1.1 User Registration

All users in the FeatureCloud ecosystem should sign up and create an account. Their user role can vary. Here, we exemplify the registration UI for developers while other users, including project managers, with the Collaborator user role, should follow the same process. FeatureCloud users can always change their role through the Edit profile page. After signing up, users should confirm their email address using the link automatically sent to them. They can request another confirmation email if needed. Figure 1 shows the sign-up page for users while figures 2 and 3 show edit profile page and login confirmation.



The screenshot shows the 'Sign Up' page of the FeatureCloud platform. At the top, there are links for 'Sign Up' (with a user icon) and 'Log In' (with a key icon). Below these, a prompt asks the user to 'Please choose an email and password.' The form contains several input fields: an email field with the text 'mohammad.bakhtiari.ai@gmail.com', a first name field with 'Mohammad', and a last name field with 'Bakhtiari'. There are two password fields; the first is highlighted in yellow and contains eight dots, with a note below it stating 'Please make sure it's at least 8 characters including a number, a lowercase letter, an uppercase letter and a special character.' The second password field also contains eight dots. Below the password fields is a dropdown menu for the user role, currently set to 'App Developer'. A note explains that the 'App Developer' role allows contributing to the FeatureCloud ecosystem by writing applications and that the role can be changed from the 'Edit profile' page. At the bottom, there is a blue 'Sign Up' button with a user icon.

Figure 1. Signup page. Users should create an account in the FeatureCloud platform by providing their first and last name and a password, at least eight characters long on the signup page.

First Name

Mohammad

Last Name

Bakhtiari

Email

mohammad.bakhtiari.ai@gmail.com

About Me

User role

App Developer ▾

The App Developer role makes possible to contribute to FeatureCloud ecosystem with writing applications.

Submit

Old password

.....

New password

Please make sure it's at least 8 characters including a number, a lowercase letter, an uppercase letter and a special character.

Retype password

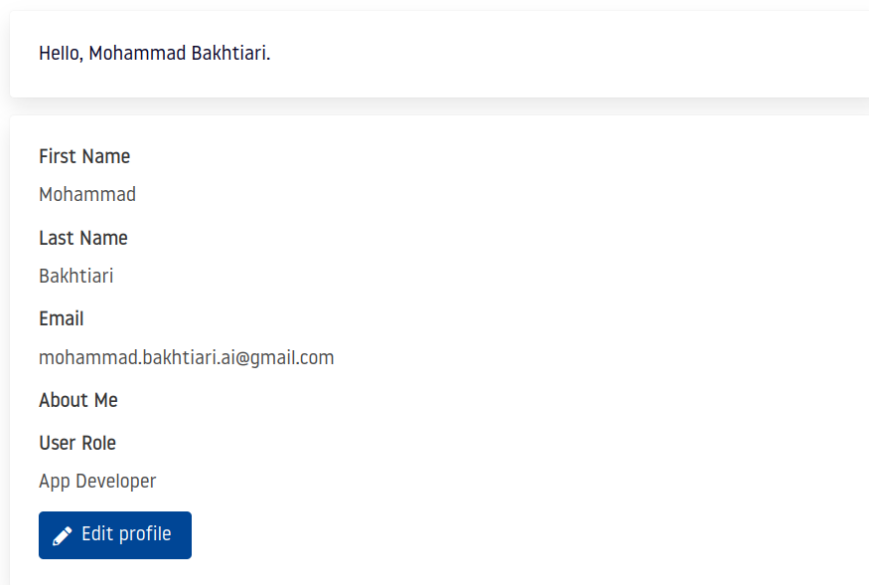
Change password

Email

Delete account

Figure 2. Edit profile page.

After signing up, users should log in into the featurecloud.ai website using the same credential (Figure S1).



A screenshot of a web interface for login confirmation. At the top, a light gray box contains the text "Hello, Mohammad Bakhtiari." Below this, a larger white box contains a list of user details: "First Name" (Mohammad), "Last Name" (Bakhtiari), "Email" (mohammad.bakhtiari.ai@gmail.com), "About Me", "User Role" (App Developer), and a blue button with a pencil icon and the text "Edit profile".

Figure 3. Login confirmation

5.1.2 Testing apps using test UI

There are two ways to execute apps in the FeatureCloud platform:

- **Workflow:** The FeatureCloud WF is designed to support running a chain of apps in a row, where data owners (Hospitals, institutions, etc.) can collaborate in a federated manner while respecting the data privacy laws to conduct data analysis and/or machine learning tasks.
- **Testbed:** On the other hand, developers, who want to provide the community with a specific app to tackle federated learning problems, need to test their apps to be sure it works as a stand-alone app. For that purpose, FeatureCloud provides developers with a testbed that simplifies the app development process.

Before running an app, developers should build it and put the clients' data and generic files, with the acceptable format, in the FeatureCloud controller's data directory. The FeatureCloud testbed is available at <https://featurecloud.ai/development/test> where developers can create tests by providing the app information. The testbed can contain multiple test-runs simultaneously. Each of these runs has a lifecycle that can be in any of the given states (initializing, running, or finished states).

Setup

1. What is the name of your Docker image?

Image

featurecloud.ai/fc_deep_networks:cpu

2. How many clients do you want to test?

1 2 3 4 5 6 7 8 9 10

3. What is the workspace directory of the clients?

Client 1

/home/mohammad/PycharmProjects/FeatureCloud/data/ deep/c1

Client 2

/home/mohammad/PycharmProjects/FeatureCloud/data/ deep/c2

4. What is the generic directory that all clients will have access to?

All clients will have access to 'Generic directory'. You can use this to pass common configuration or data.

Generic directory

/home/mohammad/PycharmProjects/FeatureCloud/data/ deep/generic

5. What communication channel to use?

☒ Local channel

☐ Internet channel

'Local channel' keeps the traffic on your machine and is faster. 'Internet channel' sends the traffic through the FeatureCloud server and is slower but closer to the real world setting.

6. How often do you want the testbed clients to exchange information?

Query interval (seconds)

3

Setting to a lower number is advised when having a large number of iterations each taking little time, setting to a higher value is recommended when calculations/operations need more time. Default value is 3 seconds.

7. Where should the results of the test run be saved?

/home/mohammad/PycharmProjects/FeatureCloud/data/tests/ deep

Start

Figure 4. Testing UI. App developers can test their app by providing required information about the target app and the test settings.

As it is shown in figure 4, first, they must fill in the app image name in the image box. In fact, all FeatureCloud apps are dockerized, and their images are available in the FeatureCloud docker repository. Consequently, each app image name starts with the address of the docker repository, featurecloud.ai, and ends with a unique name for the app in the repository. Here, we used the deep learning app as an example. Currently, the FeatureCloud testbed supports ten clients in front-end configuration; however, it is technically possible to use the testbed to run an app for more than ten clients. So, the developer should choose the number of clients and give the path to clients' data on a local drive. Developers should also provide the path to generic data shared among all clients, e.g., config.yml file.

Federated apps require communication rounds frequently, and developers must decide whether they want to use the internet or local channels for that purpose. Meanwhile, the FeatureCloud controller needs to check periodically for possible communication requests and conduct them if there are any. Accordingly, developers can decide about query intervals. Like input data, once FeatureCloud apps produce the output, it should be stored in the FeatureCloud controller's data directory. Again, it is up to developers to provide the path to the convenient subdirectory.

Once the testbed configuration is completed (Figure 5), after pressing the start button, app execution starts, and there will be panels to inform developers of the status or outputs of the app instance for each client. Also, there are tools in place to manage the testbed. The FeatureCloud testbed provides three panels, configuration, overview, and traffic, to show the necessary information about the set-run to users. Based on such information as the app’s status, or operational or logical states, users can decide either to continue with app execution or to interrupt it in the test-run.

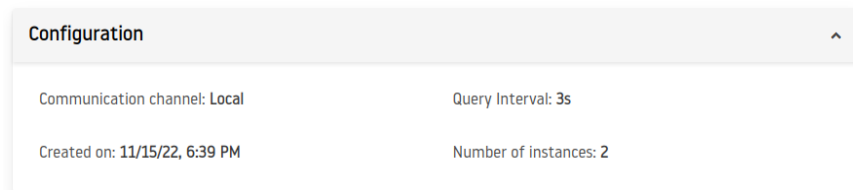


Figure 5. The configuration panel shows the testbed config details.

To interrupt the test-run, users can stop the test run or go back to the list of created apps and directly delete it. If the user stops the test run, it kills the app docker container; however, the test run will still be available in the controller. After the test-run is stopped, users can delete it in the testbed tests list. Another alternative to interrupt the test-run is directly deleting it. Regardless of the app state, the test-run will be first stopped and then deleted from the controller. Users can find out about different simulated clients’ statuses in the overview panel. As it is shown in Figure 6, there are seven columns for each client.

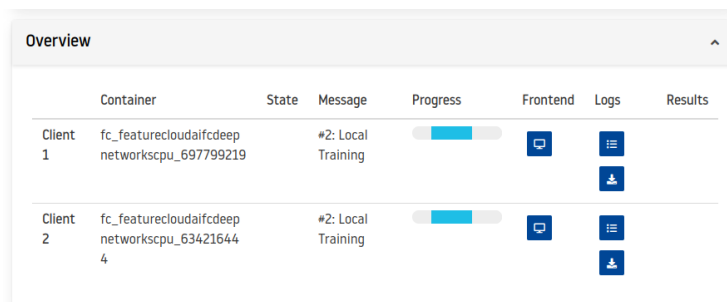


Figure 6. Overview panel in the test run.

The container shows the ID that can be used to interact with the Docker engine. State and message columns show the logical state of the app and the app’s message to the end-user, respectively, which can change during the app execution. Using the app’s front-end button, the end-user can access the app’s frontend, which can be interactive for some apps, e.g., the clustering visualization app. Users can access the app’s logs or download them at the end of app execution using the log button. Once the app execution is done, users can download the results using the download button.

Traffic				
No.	Client	Time	Size	Type
1	a1a6f04221a3183d	Nov 15, 2022, 6:39:42 PM	87764	broadcast
2	a1a6f04221a3183d	Nov 15, 2022, 6:39:57 PM	1157775	smpc
3	a1a6f04221a3183d	Nov 15, 2022, 6:39:57 PM	1157619	smpc
4	41f1ccd6ce98cef9	Nov 15, 2022, 6:40:02 PM	1158158	smpc
5	41f1ccd6ce98cef9	Nov 15, 2022, 6:40:02 PM	1157836	smpc
6	41f1ccd6ce98cef9	Nov 15, 2022, 6:40:02 PM	1157899	smpc_agg
7	a1a6f04221a3183d	Nov 15, 2022, 6:40:02 PM	1157761	smpc_agg
8	a1a6f04221a3183d	Nov 15, 2022, 6:40:03 PM	175509	broadcast
9	a1a6f04221a3183d	Nov 15, 2022, 6:40:27 PM	1157610	smpc
10	a1a6f04221a3183d	Nov 15, 2022, 6:40:27 PM	1157808	smpc
11	41f1ccd6ce98cef9	Nov 15, 2022, 6:40:29 PM	1157954	smpc
12	41f1ccd6ce98cef9	Nov 15, 2022, 6:40:29 PM	1157834	smpc
13	41f1ccd6ce98cef9	Nov 15, 2022, 6:40:29 PM	1157768	smpc_agg
14	a1a6f04221a3183d	Nov 15, 2022, 6:40:29 PM	1157846	smpc_agg
15	a1a6f04221a3183d	Nov 15, 2022, 6:40:30 PM	175509	broadcast
16	41f1ccd6ce98cef9	Nov 15, 2022, 6:40:56 PM	1157955	smpc
17	41f1ccd6ce98cef9	Nov 15, 2022, 6:40:56 PM	1157787	smpc
18	a1a6f04221a3183d	Nov 15, 2022, 6:40:57 PM	1157827	smpc
19	a1a6f04221a3183d	Nov 15, 2022, 6:40:57 PM	1157887	smpc
20	41f1ccd6ce98cef9	Nov 15, 2022, 6:40:57 PM	1157676	smpc_agg
21	a1a6f04221a3183d	Nov 15, 2022, 6:40:57 PM	1157794	smpc_agg
22	a1a6f04221a3183d	Nov 15, 2022, 6:41:00 PM	175509	broadcast

Figure 7. Traffic panel in the test run. Traffic panel provides information about all the transactions over local or internet networks between clients.

During the app execution, the FeatureCloud testbed provides traffic information for each communication in the platform. Communications in the testbed are recorded based on chronological order. As it is shown in Figure 7, each record includes the senders' client ID, assigned by the controller, the time that communication was completed, based on the coordinator's machine time, data size in bytes, and the type of communications, i.e., Gather, Broadcast, or SMPC.

5.1.3 Command Line Interface (CLI)

Using the FeatureCloud pip package, users can quickly develop federated apps that can be run on the FeatureCloud platform without being obliged to use the front-end UI. CLI can be more convenient for developers to develop apps while they only need to install the “featurecloud” python package:

```
$ pip install featurecloud
```

and call the following command to create an app:

```
$ featurecloud app new <APP_NAME> <template_name>
```

Where APP_NAME is the name to assign to the directory containing the app, and template-name is the URL of a specific sample app provided on FeatureCloud GitHub repositories. After implementing the app, users can use the CLI to build their app, run the controller, and start a test run:

```
$ featurecloud app build fc_test
$ featurecloud controller start
$ featurecloud test start --app-image fc_test
```

Using the CLI, developers can run their apps by providing all the information that is asked for in the testing UI. FeatureCloud CLI is fully documented and publicly available on the FeatureCloud GitHub repository. The implementation of CLI to start the controller is provided in section 11.2.

For publishing an app, once the development is finished, developers should register their app in the app store. App registration process is explained in 11.3. Assuming the app’s docker image name is fc_test, developers can use the CLI to push the app to FeatureCloud docker registry:

```
$ featurecloud app publish featurecloud.ai/fc_test:latest
```

5.1.4 Creating an app

App developers want to present their apps to different research communities and persuade them to utilize their applications. To facilitate the app publication process and finding its target research groups, FeatureCloud asks developers to register their app using the app registration form. FeatureCloud requires developers to provide the source code of apps in public repositories like GitHub.

In the app registration form, developers should assign a unique name and image name for the app and its docker image file that will be stored in the FeatureCloud docker registry. Each app will receive an auto-generated slug, a unique, human-readable identifier for your app, on the app store, <https://featurecloud.ai/app/>. This slug is based on the provided app name, as an address to directly access the app page. Developers have the option to change the address in the app registration form. All apps should provide a Readme file to explain how that app can be used with the sample data, and developers are required to provide such information while markdown is supported. Developers should assign a subset of labels to their apps to be easily findable in the app store.

As it is shown in Figure 8, they should explicitly declare if their application provides a front-end. They also need to make it clear whether their app needs to be manually stopped from the FeatureCloud user interface. If the app displays a result or visualization, and it does not have an internal termination implementation, the user should decide when to proceed further with the workflow. Once the experiments are done, developers can publish their app in the app store by changing the state of “Published to app store” from unpublished to published. Finally, app developers should provide a URL to the hosting repository for the source code and an icon for their app.

5.1.5 Providing feedback

In case app developers encounter any issue during the usage of the FeatureCloud platform, or they have an idea for improvement/feature request, they can provide feedback using the feedback form under the “Help” main menu item (Figure 9).

On this form the users are asked to provide a subject, description and some specific details about their issue: the type of the issue (Bug, How to, Problem, Suggestion, Other), the operating system they are using. The users can also upload supporting documents or any relevant data that might be helpful for the issue’s further investigation.

Based on the submitted data an issue is created in the feedback/bug tracker system, the reporter is notified via email about this and about any further updates on the issue.

App data

Name

Image Name

featurecloud.ai/

Slug

https://featurecloud.ai/ai-store/














A unique, human readable identifier for your app

Type

Short Description

Long Description

Markdown is supported

H B I             

Labels

Start typing to see suggestions. Press Enter or Tab to assign a label missing from the suggestion list.

☒ This application provides a frontend

☐ This application needs to be stopped from FeatureCloud user interface

If the app displays a result or a visualization and it does not have an internal termination implementation, it is the user's decision when to proceed further with the workflow.

Published to AI Store

Source Code URL

App Icon

The app icon width compared to its height should be 3:2 (e.g. 300x200)



 Choose a file...

Figure 8. App register page. FeatureCloud apps can be published in the app store and later be certified. To provide enough information for the end-users, e.g., project managers, each app should provide some details about its functionalities, code, etc.



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826078.

Page 16 of 35

Do you have any problems using the FeatureCloud platform or a specific app?
Do you have a question or a request?

Please submit an issue

Subject

Description

Describe your issue

Related App

n/a

▼


Type

▼

Operating System

▼

Add file(s)

 Add file ...

Submit

Figure 9. Feedback form.

5.2 User interface for project managers

5.2.1 Registration and user Account

The user interface for project managers is mainly provided via the website featurecloud.ai. The first thing project managers need to do is to create an account using the Sign-Up menu item in the top right corner. The signup form (shown in Figure 1) expects an email address, first and last name, and a secure password that must fulfill various security requirements. Finally, the user role “Collaborator” can be selected, and the user will sign up. After signing up/logging in, the project managers need to create a site. A site describes the institute rather than a specific user and is used for identification in

a federated workflow. After that, the user has various new options on the website to run a federated machine learning workflow together with other sites.

A site is an entity that identifies you in a federated workflow. By that, individual users are not visible to other participants. Users need to be assigned to a site in order to run federated workflows. You can create a new site or ask your site administrator to add you. Site registration and edit page is shown in Figure S2.

5.2.2 The FeatureCloud app store and personal app library

A major part of the FeatureCloud platform is the integrated App Store (Figure 10). Here, project managers can filter and search the store for appropriate apps they want to use in the future. It contains various preprocessing methods, machine learning algorithms, evaluation and visualization methods, as well as whole workflows that can be added to a project manager’s personal library.

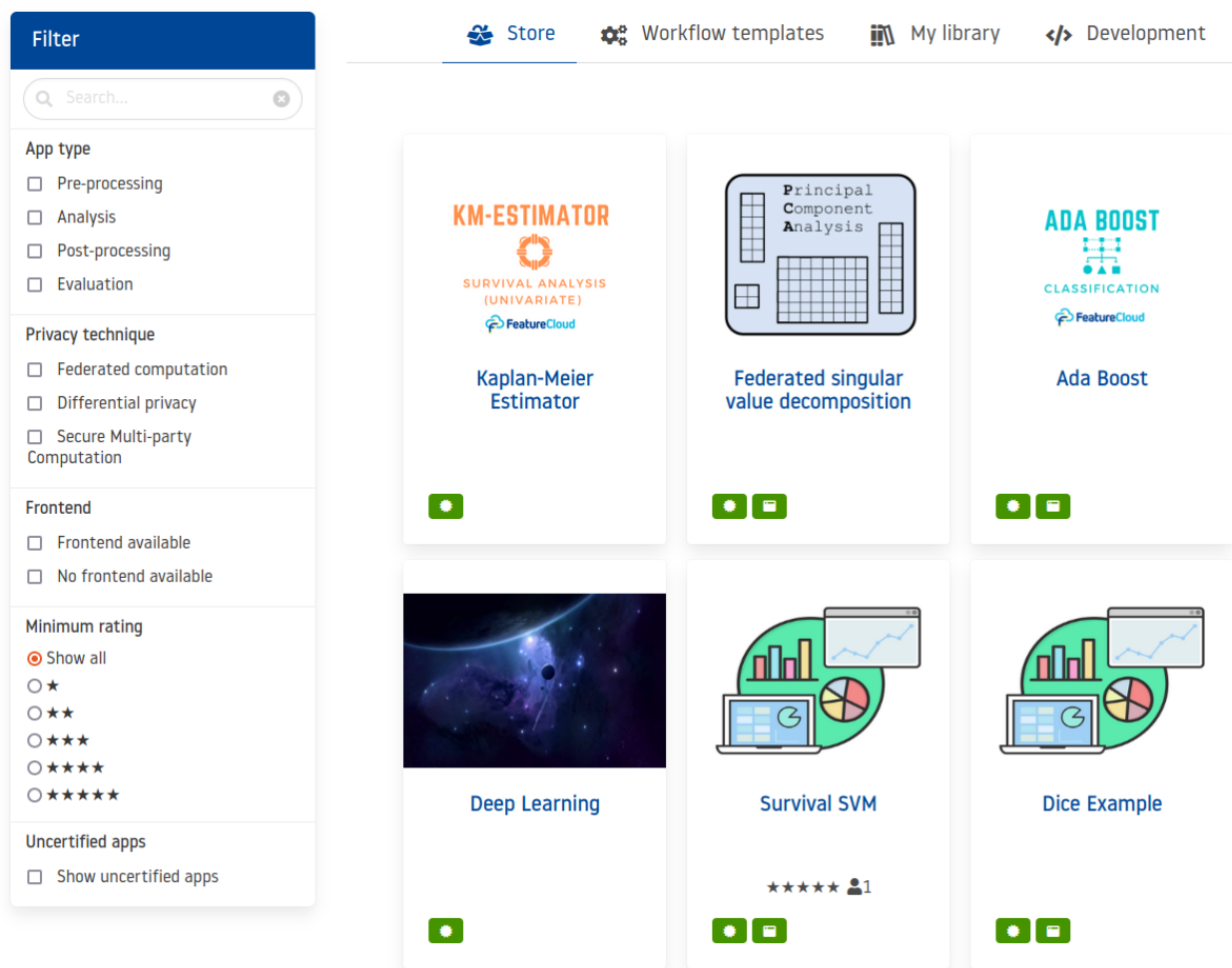


Figure 10. FeatureCloud app store.

The FeatureCloud platform presents an app store for the FeatureCloud community, where developers can publish their apps and make them available for different hospitals, institutes, etc. Currently, the app store supports apps and workflow as products that are available for end-users. Apps are generally categorized into the following categories:

1. Pre-processing: All the apps which are focused on preparing data for analysis apps fall in this category; Pre-processing apps are commonly used as the first apps in FeatureCloud workflow. It includes apps concerned with loading, splitting, and distributing data: e.g., [Cross-Validation](#), [Data distributor](#), etc. Also, normalization apps will be found under this category.
2. Analysis: Federated machine learning and data analysis apps are categorized as analysis apps. Analysis apps are the leading apps that end-users use in a workflow to collaborate, therefore, they have global aggregation at some point. Besides machine learning apps like [Deep Learning](#) and [Random-forest](#), end-users can also find specific federated data analysis apps like [sPLINK](#) and [Flimma](#) among the analysis apps.
3. Evaluation: At the end of a workflow, once the results of analysis apps are available, evaluation apps can be used for calculating some criteria based on the results and ground truth, if it's available. Also, visualization apps, like [evaluation\(Classif.\)](#), fall under this category, which is responsible for providing plots for end-users to have a better understanding of the outcome.

Even though federated learning is a way to tackle privacy challenges, there can be more considerations to provide a better level of privacy awareness or preservation in federated apps. For that purpose, FeatureCloud provides privacy-preserving mechanisms like Secure Multi-Party Computation (SMPC) (Zhao *et al.*, 2019), which apps can leverage to enhance their privacy. For end-users, there is a privacy technique panel to filter apps that employ specific privacy-preserving techniques:

- Federated computation: All apps that refrain from moving any data to centralized custody by using federated computation will be found in this privacy-preserving category.
- Differential privacy
- Secure Multi-party Computation

For some tasks, FeatureCloud end-users may have multiple apps they can utilize, and a review and scoring mechanism is in place to guide them. FeatureCloud users can review every app to point out the strong and weak sides of the app. In addition, end-users can rate apps from one to five stars. End-users can choose between alternative apps based on reviews and rates. Meanwhile, not all apps in the app store are certified by the FeatureCloud certification process; if some users want to use such apps, they can be found in the app store by marking the “show uncertified apps” box. End-users can use uncertified apps at their own risk.

5.2.2.1 App Details

Each app in the FeatureCloud app store has a dedicated page on featurecloud.ai to provide information that will be used by app developers, end-users, and the FeatureCloud certification team (Figure S3). The page is editable (Figure S5 and Figure S6) for developers where they should give a name, description (Figure S4), link to a public GitHub repository containing the implementation, tags to express the application of the app, and docker image-name, which will be used to pull the app from FeatureCloud docker repo. The app page includes review (Figure S7) and statistics (Figure S8) tabs.

5.2.3 Creating a project and assembling a workflow

The main goal for the project manager in FeatureCloud is to run a federated workflow with other sites. For this, FeatureCloud provides a “Projects” subpage that leads a project manager through the various steps of a federated workflow. To create a project, the coordinator goes to the projects page and presses the “Create” button. Here, the user enters the project name and a detailed description and presses the “Add” button. Afterward, the workflow needs to be assembled. For this, based on the apps in the user’s library, the coordinator can choose apps and their execution order and press “Finalize” when everything is in the right order. Figure 11 shows the assembled workflow consisting of a Cross-Validation, Normalization, Logistic Regression, and Evaluation (Classification) app.

← Back
Clone project
Create workflow template

PROJECT

Test Project

Name

Test Project

Description

This is a project with a test workflow for demonstration purposes.

Available applications

Application	Description	Type	
Cross Validation	A cross validation app, that creates different splits.	Pre-Processing	→
Evaluation (Classif.)	An app, computing various metrics to evaluate the performance of a classification model	Evaluation	→
Logistic Regression	Logistic Regression classifier	Analysis	→
Normalization	Normalizes input by variance.	Pre-Processing	→

Workflow

Application	
Cross Validation	↑ ↓ 🗑️
Normalization	↑ ↓ 🗑️
Logistic Regression	↑ ↓ 🗑️
Evaluation (Classif.)	↑ ↓ 🗑️

Save
Finalize

Figure 11. Assembling a workflow in FeatureCloud.

5.2.4 Inviting participants and joining projects

Now that the workflow is assembled, the project coordinator can access the project details page in the top section. Here, the coordinator can double-check that all previous actions are correct, scroll to the bottom, and create tokens that can be shared with other sites to join the projects (Figure 12).

Invitations		
Token	Site	Action
cb953c0d-264f-43f0-8e58-7f5ccb5c1184	Munich	
4d9d82f8-4748-42aa-a786-7bee30babfd9		Copy Delete
21a9fac0-5ecd-4597-9b3e-56496f356cd4		Copy Delete
+ Token		

Figure 12. Inviting other participants. The invited sites can use these tokens after they create an account for joining the project through the projects page by pressing the “Join” button.

5.2.5 Choosing data and starting a workflow

To continue, project managers and participants must run the FeatureCloud controller in the background. This is prominently mentioned on the page if the controller is not running.

To run the FeatureCloud controller, Docker needs to be installed and ran.

An alternative is to download the “start controller” script (macOS/Linux, Windows) from the FeatureCloud website and run it with a double-click on Windows or macOS/Linux using the following command:

```
sh start_controller.sh
```

After starting the FeatureCloud controller, the project manager presses the “Start” button to initiate the project. Following that, the project manager and the participants can upload the data using the “Add file...” button. After adding the files needed by the apps of the workflow, the “Continue workflow” button can be pressed and confirmed with the “Yes” button to start the workflow. When all participants are ready, the workflow is started automatically (Figure 13). Intermediate results can be downloaded directly after an app has finished, and logs can be followed during the execution.



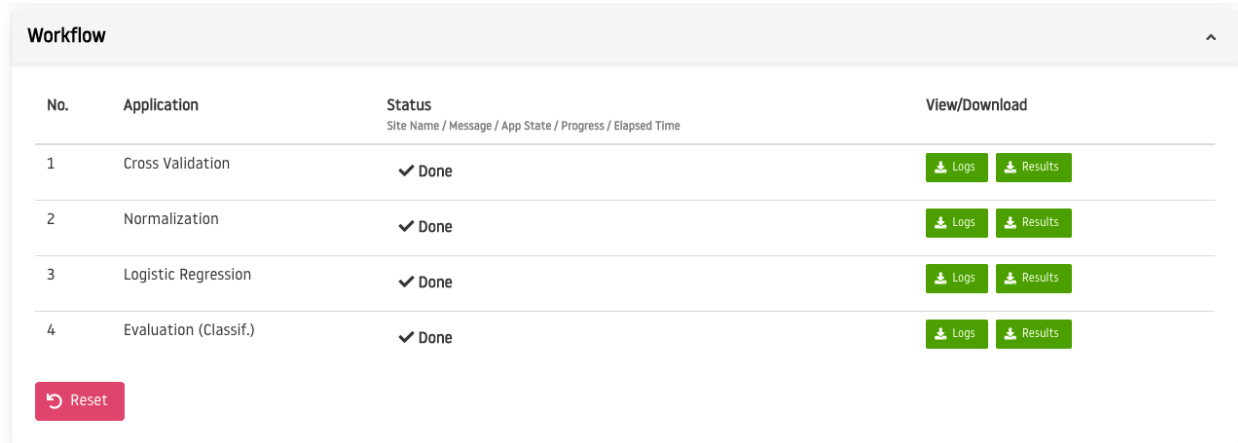
Workflow			
No.	Application	Status <small>Site Name / Message / App State / Progress / Elapsed Time</small>	View/Download
1	Cross Validation	✓ Done	Logs Results
2	Normalization	<div>  Running </div> <div> Julian Späth (Munich) </div> <div> Running application </div> <div>  </div> <div> 00:00:02 </div>	Logs
3	Logistic Regression	■ Not started	
4	Evaluation (Classif.)	■ Not started	
Stop workflow			

Figure 13. Running workflow in FeatureCloud.

5.2.6 Workflow progress, logs, and results

After the workflow has finished, every participant can download the logs and results for each intermediate app as well as the final results of the workflow by downloading the results of the last app in the workflow (Figure 14).



Workflow			
No.	Application	Status	View/Download
		Site Name / Message / App State / Progress / Elapsed Time	
1	Cross Validation	✓ Done	Logs Results
2	Normalization	✓ Done	Logs Results
3	Logistic Regression	✓ Done	Logs Results
4	Evaluation (Classif.)	✓ Done	Logs Results
Reset			

Figure 14. Finished workflow in FeatureCloud.

Results are downloaded as a zip folder containing all data of the corresponding intermediate step (app results). The final workflow results are always stored in the last app of the workflow. The logs of each app can be downloaded as a log file and inspected if necessary. An overview of the necessary steps for running workflow in the FeatureCloud platform is depicted in Figure 26.

5.2.7 Providing feedback

In case project managers encounter any issue during the usage of the FeatureCloud platform or they have an idea for improvement/feature request, they can provide feedback using the feedback form under the “Help” main menu item (Figure 9), similar to app developers, as described in chapter 5.1.5 Provide feedback.

5.3 User Interface for patients

For the handling of data consent, we provide an intuitive frontend for patients to submit or revoke their consent. Hospitals should adopt and deploy the patients' UI to get patients' consent and store it in hospitals' local databases. The hospitals should submit the consent to the blockchain as a hash for a potential audition later. This section describes how patients can use their UI to manage their rights on the hospital's platform. More information regarding the consent management system and its integration into the FeatureCloud platform can be found in D7.5.

5.3.1 Submitting a consent

Patients should sign up for the UI that is deployed by the hospital using the credentials that the hospital provided to them. Afterwards, they can submit new consent regarding their data. The scope of consent, a legal issue, can vary and should be decided by hospitals; however, regardless of the

scope, patients should go through the UI to get information regarding the scope and personal information.

Hospital Clinic

Patient Consent Management

✓ Submit consent

✓ Update consent

🗑 Revoke consent

Consent Form

for patients, former patients and prospective patients

Personal Information

First Name	Address
<input type="text" value="Please enter your first name"/>	<input type="text" value="Please enter your street and house number"/>
Middle Name(s)	Zip
<input type="text" value="Please enter your middle name (s)"/>	<input type="text" value="Please enter your ZIP code"/>
Last Name	City
<input type="text" value="Please enter your surname"/>	<input type="text" value="Please enter your city"/>
E-mail	State
<input type="text" value="Please enter your e-mail address"/>	<input type="text" value="Please enter your state"/>
Phone	Country
<input type="text" value="Please enter your phone number"/>	<input type="text" value="Please enter your country"/>

Consent

☐ I confirm that I received the information regarding study participation and I consent voluntarily to participate in upcoming studies. I understand that I can withdraw from any study at any time without giving any reasons and without effect to my treatment in the Hospital Clinic.

☐ I consent voluntarily that personal data that was collected about me in the Hospital Clinic, including data related to my physical or mental health can be processed for the purposes of the study by associated study coordinators. I understand that I can withdraw my consent at any time without giving any reasons by using this application or contacting the Hospital Clinic at info@hospital.com or Address 123, 12345, City, State, Country. The withdrawal of consent does not affect the lawfulness of processing based on consent before its withdrawal.

If necessary, please contact me by

☐ E-mail ☐ Phone ☐ Post

Keep me informed about news, events, activities and studies

☐ Yes ☐ No

☐ I have taken note of the [privacy statement](#)

Figure 15. Patients' UI

For submitting a new consent, the patient can fill out a form with all personal information (Figure 15), and depict a timeframe in which the consent is valid (Figure 16). Based on this information, a PDF will be created.




Figure 16. Consents validation time period and download.

As it is shown in Figure 16, the generated PDF can be downloaded, signed by the patient and uploaded again through the form (Figure 17). The uploaded consent can then finally be submitted to the hospital by clicking the green “Submit Consent” button. For this, a date and digital signature is needed. This will trigger the hospital internal consent management, and push the consent hash to the blockchain.

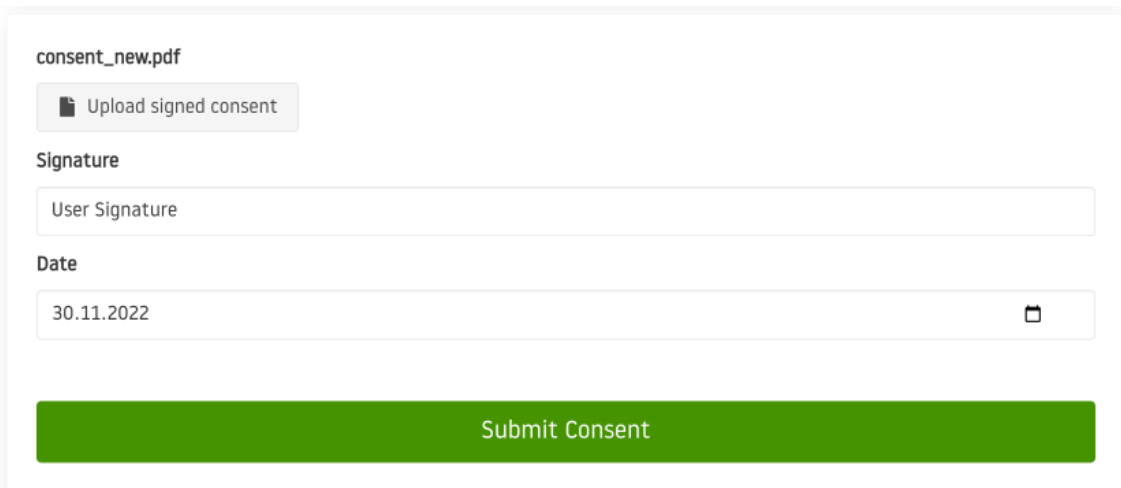


Figure 17. Upload and submit the signed consent.

After submitting the consent, the UI shows a confirmation box (Figure 27) to the patient. For revoking a consent, the patient can use a similar form and enter his personal information and the Consent ID for which he wants to revoke the consent. Similar to the submission, the request will be forwarded to the hospital, and the process of submitting it to the blockchain will happen locally at the hospital. The revocation form is shown in Figure 28.

6 Open issues

No open issues.

7 Deviations (if applicable)

Patients UI is not deployed on the FeatureCloud platform because it requires an identification process, to provide services for patients, which compromises user privacy. Therefore, FeatureCloud provides a generic UI that can be adopted and integrated into hospitals' platforms based on the scope of the consent, the legal content of the consent, and also the technical aspect of infrastructure and software in the hospitals. FeatureCloud demonstrates such UIs can be integrated into the FeatureCloud platform using the controller, FeatureCloud CLI, and blockchain CLI.

8 Conclusion

The provided UIs have significantly simplified the supported functionalities for a variety of users. For app developers, the CLI eases the testing and deployment of applications in the app store while the app store, in turn, simplifies the promotion of apps for developers and finding desired privacy-aware apps for project managers. The patients' UI exemplifies the automation of consent management and submission and audition of consent through the blockchain consent management system in a transparent manner.

9 References

Angular (no date). Available at: <https://angular.io/> (Accessed: 22 November 2022).

Bulma: Free, open source, and modern CSS framework based on Flexbox (no date). Available at: <https://bulma.io/> (Accessed: 22 November 2022).

Django (no date) *Django Project*. Available at: <https://www.djangoproject.com/> (Accessed: 28 November 2022).

Font awesome (no date). Available at: <https://fontawesome.com/> (Accessed: 22 November 2022).

JavaScript with syntax for types (no date). Available at: <https://www.typescriptlang.org/> (Accessed: 22 November 2022).

Meyerson, J. (2014) 'The go programming language', *IEEE Software*, 31(5), pp. 104–104. Available at: <https://doi.org/10.1109/ms.2014.127>.

Npm: Bulma-toast (no date) *npm*. Available at: <https://www.npmjs.com/package/bulma-toast> (Accessed: 22 November 2022).

npm: ng5-slider (no date) *npm*. Available at: <https://www.npmjs.com/package/ng5-slider> (Accessed: 22 November 2022).

Npm: @ng-util/markdown (no date) *npm*. Available at: <https://www.npmjs.com/package/@ng-util/markdown> (Accessed: 22 November 2022).

PrimeNG (no date). Available at: <https://www.primefaces.org/primeng/> (Accessed: 22 November 2022).

Zhao, C. *et al.* (2019) 'Secure Multi-Party Computation: Theory, practice and applications', *Information sciences*, 476, pp. 357–372. Available at: <https://doi.org/10.1016/j.ins.2018.10.024>.

10 Table of acronyms and definitions

CLI	Command Line Interface
concentris	concentris research management GmbH
GND	Gnome Design SRL
MS	Milestone
MUG	Medical University Graz
Patients	In this deliverable, we use the term “patients” for all research subjects. In FeatureCloud, we will focus on patients, as this is already the most vulnerable case scenario and this is where most primary data is available to us. Admittedly, some research subjects participate in clinical trials but not as patients but as healthy individuals, usually on a voluntary basis and are therefore not dependent on the physicians who care for them. Thus, to increase readability, we simply refer to them as “patients”.
RI	Research Institute AG & Co. KG
SBA	SBA Research Gemeinnutzige GmbH
SDU	Syddansk Universitet
SMPC	Secure Multi-Party Computation
TUM	Technische Universitaet Muenchen
UHAM	University of Hamburg
UI	User Interface
UM	Universiteit Maastricht
UMR	Philipps Universitaet Marburg
WF	Workflow
WP	Work package

11 Other supporting documents / figures / tables (if applicable)

11.1 User registration and Login

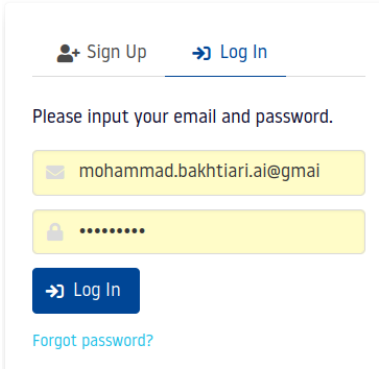


Figure S1. Login page

Site users

Please enter email address to add a user to this site

Add user

First Name	Last Name	Email	Admin
Mohammad	Bakhtiari	mohammad.bakhtiari.ai@gmail.com	✓

Site data

Cancel Update

Name

UHAM

Address

Notkestrasse 9

Zip

22607

City

Hamburg

Country

Germany

First Name

Mohammad

Last Name

Bakhtiari

Phone

+4917630362751

Figure S2. Edit and update the site

11.2 CLI: starting the controller on local machine

Implementing the CLI command for starting the controller.

```
import os
import click
from FeatureCloud.api.imp.exceptions import FCException
from FeatureCloud.api.imp.controller import commands

@click.group("controller")
def controller() -> None:
    """Controller start/stop. Obtain general information about the
    controller."""

@controller.command('start')
```

```
@click.argument('name', type=str, default=commands.DEFAULT_CONTROLLER_NAME,
nargs=1, required=False)
@click.option('--port', default=8000, help='Controller port number. Optional
parameter (e.g. --port=8000).', required=False)
@click.option('--data-dir', default='data', help='Controller data directory.
Optional parameter (e.g. --data-dir=./data).', required=False)
@click.option('--gpu', help='Start controller with GPU access. If this
succeeds, controller can allow GPU access for apps.', default=False,
required=False)
def start(name: str, port: int, data_dir: str, gpu: bool) -> None:
    """Start a controller instance.
    NAME is the controller instance name
    Example: featurecloud controller start my-fc-controller --gpu=True
    """
    try:
        commands.start(name, port, data_dir, gpu)
        click.echo(f'Started controller: {name}')
    except FCException as e:
        if str(e).find("port is already allocated") > -1:
            click.echo(f'Controller could not be started. Port {port} is
already allocated.')
        else:
            click.echo(f'Controller could not be started. Error: {e}')
```

Starting the controller based on the local machine settings and given arguments.

```
import tqdm
import time
import docker
import os

from FeatureCloud.api.imp.exceptions import FCException, ContainerNotFound
from FeatureCloud.api.imp.test.helper import http

from FeatureCloud.api.imp.util import getcwd_fslash, get_docker_client

CONTROLLER_IMAGE = "featurecloud.ai/controller"
CONTROLLER_LABEL = "FCControllerLabel"
DEFAULT_PORT = 8000
DEFAULT_CONTROLLER_NAME = 'fc-controller'
DEFAULT_DATA_DIR = 'data'
LOG_FETCH_INTERVAL = 3 # seconds
```

```
LOG_LEVEL_CHOICES = ['debug', 'info', 'warn', 'error', 'fatal']

def start(name: str, port: int, data_dir: str, with_gpu: bool):
    client = get_docker_client()

    data_dir = data_dir if data_dir else DEFAULT_DATA_DIR
    # Create data dir if needed
    try:
        os.mkdir(data_dir)
    except OSError as error:
        pass

    # cleanup unused controller containers
    prune_controllers()

    # remove controller having the same name, if any
    try:
        client.api.remove_container(name, v=True, force=True)
    except docker.errors.NotFound:
        pass
    except docker.errors.DockerException as e:
        raise FCException(e)

    # pull controller and display progress
    try:
        pull_proc = client.api.pull(repository=CONTROLLER_IMAGE, stream=True)
        for p in tqdm.tqdm(pull_proc, desc='Downloading...'):
            pass
    except docker.errors.DockerException as e:
        raise FCException(e)

    cont_name = name if name else DEFAULT_CONTROLLER_NAME
    # forward slash works on all platforms
    base_dir = getcwd_fslash()

    device_requests = None
    gpu_command = ""
    if with_gpu:
        # '--gpus all' option
        device_requests = [docker.types.DeviceRequest(count=-1,
capabilities=[['gpu']])]
        gpu_command = '--has-gpu'

    try:
```

```
client.containers.run(
    CONTROLLER_IMAGE,
    detach=True,
    name=cont_name,
    platform='linux/amd64',
    ports={8000: port if port else DEFAULT_PORT},
    volumes=[f'{base_dir}/{data_dir}:{data_dir}',
'/var/run/docker.sock:/var/run/docker.sock'],
    labels=[CONTROLLER_LABEL],
    device_requests=device_requests,
    command=f"--host-root='{base_dir}/{data_dir}' --internal-
root={data_dir} --controller-name={cont_name} {gpu_command}"
)
except docker.errors.DockerException as e:
    raise FCException(e)
except Exception as e:
    raise FCException(e)
```

11.3 App details

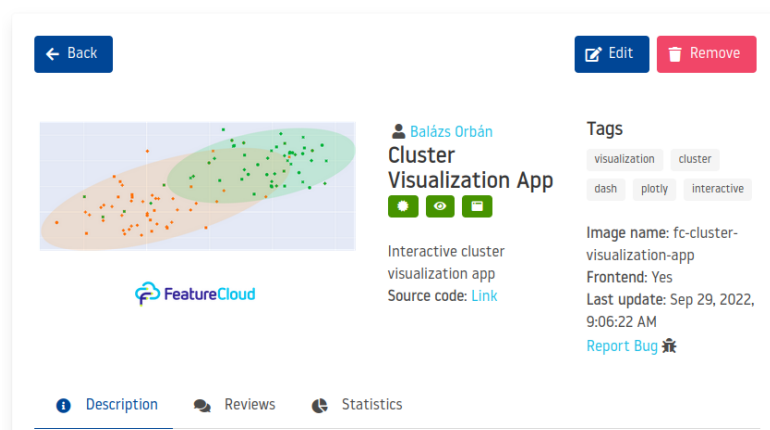


Figure S3. FeatureCloud Cluster Visualization app page

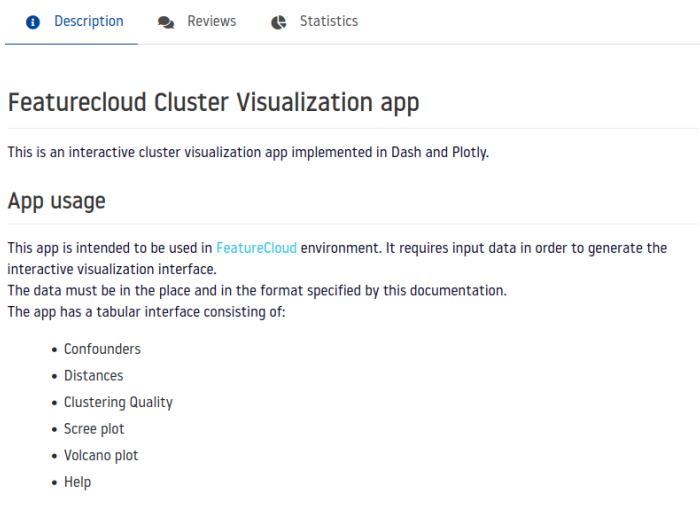


Figure S4. Description tab of FeatureCloud Cluster Visualization app.

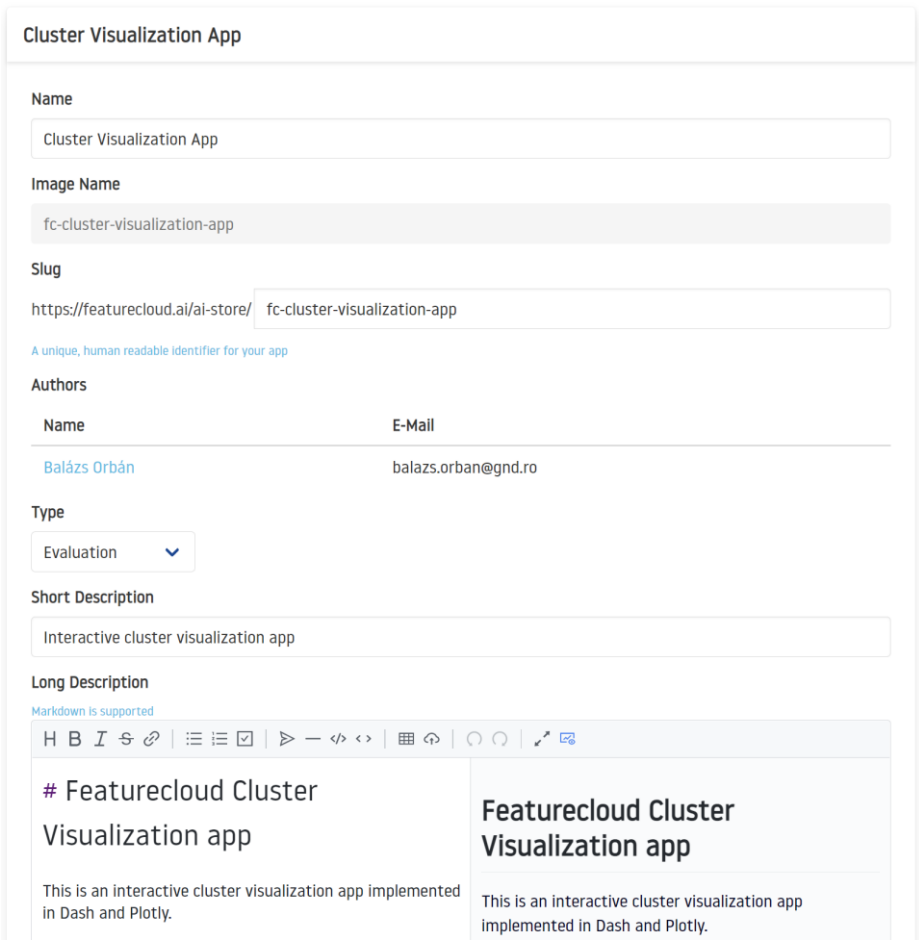


Figure S5. Edit page of FeatureCloud Cluster Visualization app (part 1)

Labels

visualization ✕ cluster ✕ dash ✕ plotly ✕ interactive ✕

Start typing to see suggestions. Press Enter or Tab to assign a label missing from the suggestion list.

☒ This application provides a frontend

☐ This application needs to be stopped from FeatureCloud user interface

If the app displays a result or a visualization and it does not have an internal termination implementation, it is the user's decision when to proceed further with the workflow.

Published to AI Store


Published ▼

Certification status

☒ Certified app

Source Code URL

App Icon



The app icon width compared to its height should be 3:2 (e.g. 300x200)


 Choose a file...

Figure S6. Edit page of FeatureCloud Cluster Visualization app (part 2)

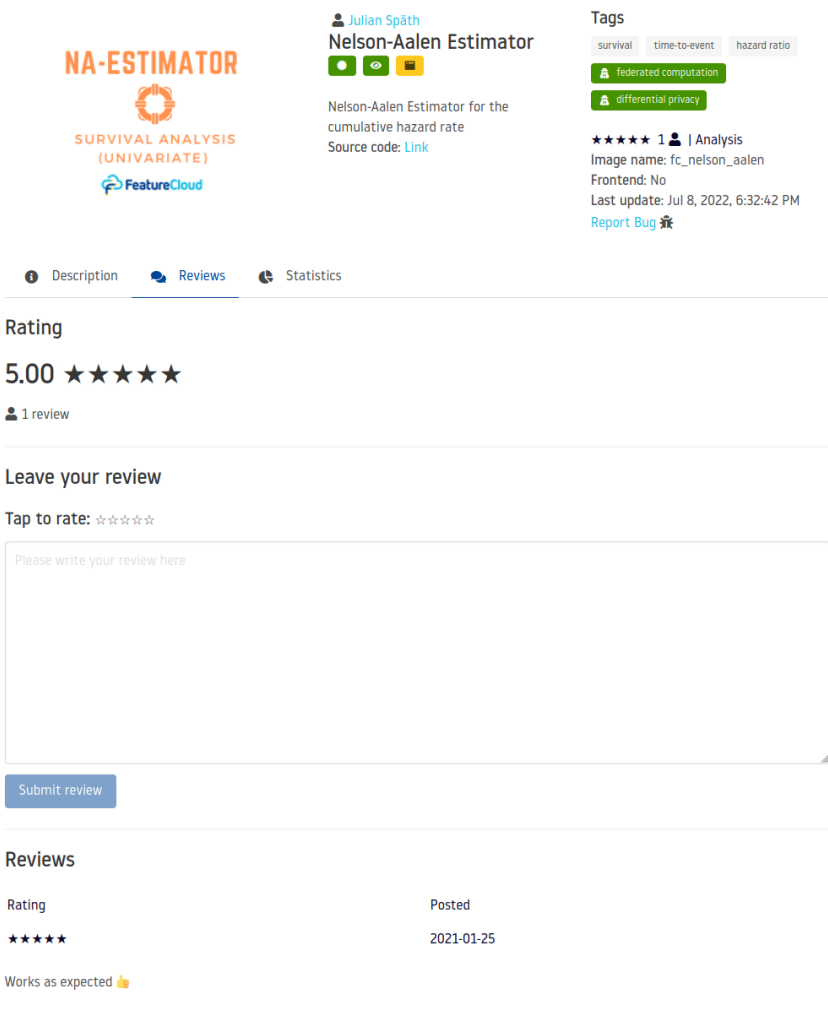


Figure S7. Nelson-Aalen Estimator app’s Review tab.

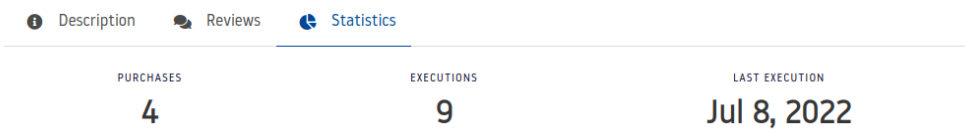


Figure S8. Nelson-Aalen Estimator app’s Statistics tab.

11.4 Running a workflow

FeatureCloud Run a Workflow

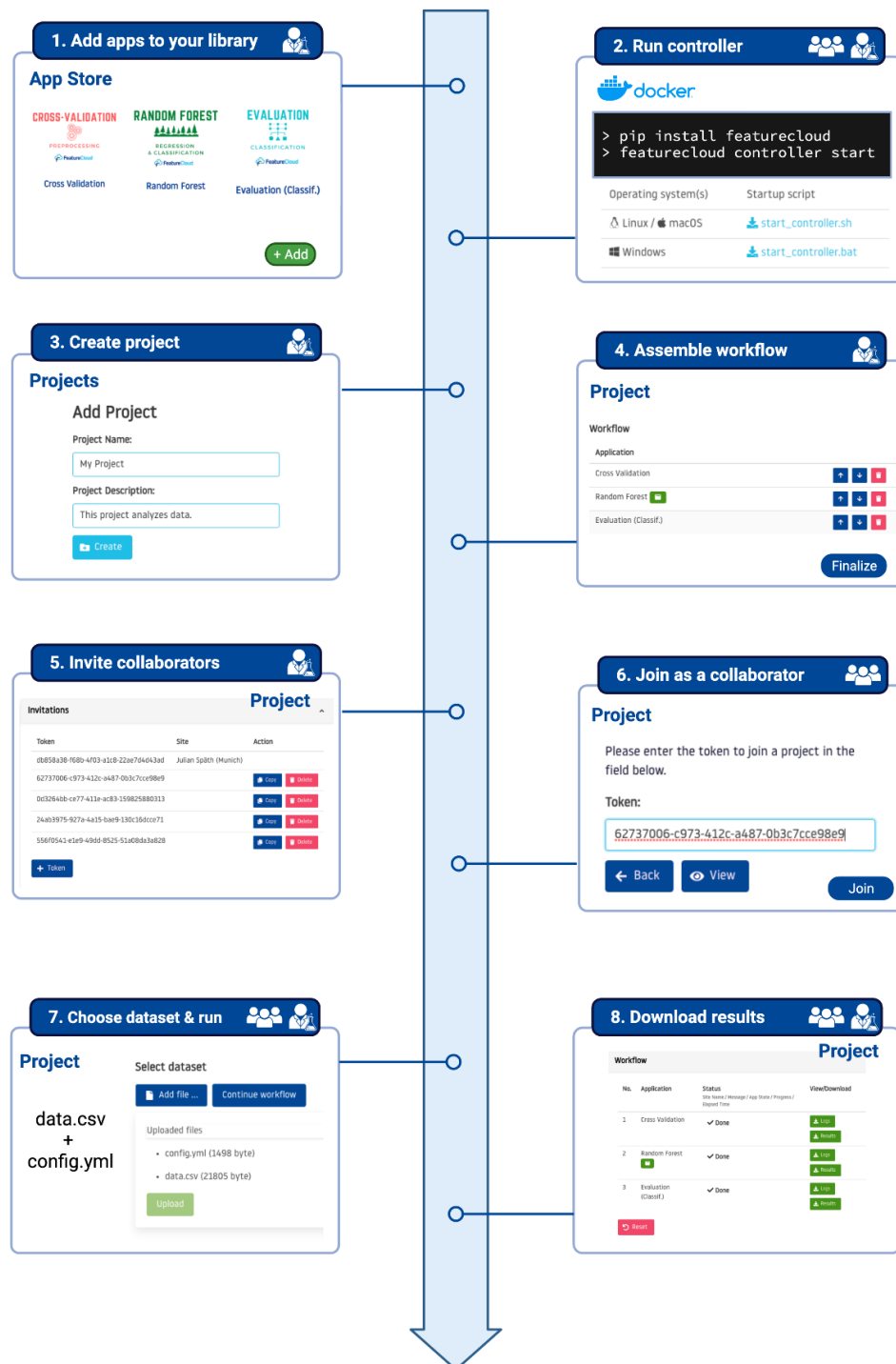


Figure S9: Using the UI for project managers to run a workflow.

11.5 Patients UI

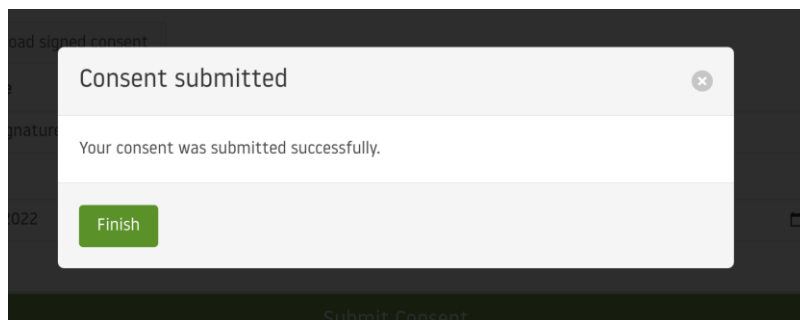


Figure S10. Consent submission confirmation.

Revoke consent

Personal Information

First Name <input type="text" value="Please enter your first name"/>	Address <input type="text" value="Please enter your street and house number"/>
Middle Name(s) <input type="text" value="Please enter your middle name (s)"/>	Zip <input type="text" value="Please enter your ZIP code"/>
Last Name <input type="text" value="Please enter your surname"/>	City <input type="text" value="Please enter your city"/>
E-mail <input type="text" value="Please enter your e-mail address"/>	State <input type="text" value="Please enter your state"/>
Phone <input type="text" value="Please enter your phone number"/>	Country <input type="text" value="Please enter your country"/>

By signing this form you are revoking the consent from us for processing your personal data for the following purposes (please tick the boxes where you grant consent).

Unique Consent ID

Signature

Date

Revoke Consent

Figure S11. Consent revocation.