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 D7.5
"Project manager and patients can control access rights with integrated blockchain"

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

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	D7.5 “Project manager and patients can control access rights with integrated blockchain”			
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), Balázs Orbán (GND), Sándor Fejér (GND), Walid Fdhila (SBA), Walter Hötzendorfer (RI)			
Keywords	Federated Collaboration, Federated Learning, Consent management, User Interface			

Table of Content

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

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

As explained in the Description of Action, D7.5, “Project manager and patients can control access rights with integrated blockchain” is closely related to WP7, task 3, “User interfaces and testing”, and demonstrates the necessary interaction between the patient User Interface (UI) and blockchain systems by integrating the patients' UI into the FeatureCloud platform, providing a generic UI that can be adopted and integrated into hospital platforms, and the potential for federated collaboration. D7,5 also touches on Task 1, “Programming interfaces and platform,” by integrating the blockchain system into FeatureCloud as the main product of WP6. Last, D7.5 also contributes to Task 5, “Stress testing,” by providing the groundwork for future evaluation of the platform and blockchain right management concerning the policies.

2 Executive Summary

D7.3 covers app development using the FeatureCloud platform (pip package) (CLI) and interacting with the controller (e.g., sending data to other clients). D7.4 covers the UIs for different users. D7.5 extends the UI to interact with the blockchain consent management system, which is the main product of WP6, by integrating the blockchain system into FeatureCloud. Hospitals can receive patients' consent, issue them to the blockchain system and query the submitted constants (see section 4.2). D7.3 explains the app development process for software developers to register and test new applications. D7.4 extends the work to provide comprehensive UI for developers while designing a deployable patients' UI on the hospitals' platform. This deliverable integrates the UI into the FeatureCloud platform to exemplify the required interaction between patients' UI and the blockchain system by providing a generic UI that can be adopted and integrated into the hospital's platform alongside employing FeatureCloud for federated collaboration (see section 5.2). D7.3 conducts stress testing of the platform by experimenting with various CPU architectures, operating systems, etc. and this deliverable provides the groundwork for future stress testing of the blockchain consent management system (see section 5.1).

3 Introduction (Challenge)

FeatureCloud, as a federated platform, provides a privacy-by-design solution and enables federated machine learning of healthcare data. Instead of collecting and storing the data from multiple participants in a centralized repository, FeatureCloud keeps the training data at their data providers’ sites and aggregates the models instead. As the machine learning (ML) models are trained locally before aggregation, mechanisms that prevent participants from acting maliciously (e.g., data poisoning or use of non-consented data) becomes necessary. To address this, FeatureCloud employs rather a detective approach to secure the audit processes (see deliverables D6.1 to D6.4). The approach uses blockchain technology as an audit trail, where internal operations for managing consents or conducting ML studies are committed to a permissioned blockchain specific to FeatureCloud using the appropriate smart contract.
In federated collaboration, two or more participants may join to run a workflow in the FeatureCloud platform. Besides functionalities for running the workflow and the ML studies, FeatureCloud provides a CLI to interact with the blockchain. In order to comply with data protection regulations such as GDPR and the right to erasure (see Deliverables D6.3 and D6.5), no personal data is stored on the blockchain. Instead, only the commitments of both patient consents and ML studies are pushed to the ledger. As such the FeatureCloud platform submits hashed consents and data to blockchain while providing a copy of the files to the hospitals to make the process auditable. In a federated collaboration that includes two clients, each of them will run the same components of the FeatureCloud platform while locally deploying patients' UI to capture and store the patients'

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

consents. Patients' data is always stored locally in the corresponding hospitals or institutions, and GDPR laws require hospitals to refrain from any application that may compromise privacy.
The patients’ UI allows a patient to give and revoke consent to use his data for research. During the auditing process, the hash of consents, input data, ML studies as well as the output models., will be compared against the corresponding hashes stored on the blockchain. Since the Machine Learning (ML) study results will be hashed via the controller, we provide options for hashing that can be extended in the future. For the sake of consistency, we recommend both patients' UI and the controller use the same methods for creating the hashes. For each workflow (WF), a specific ML study entry should be created in the blockchain to be used by participants to submit their results.

4 Methodology

FeatureCloud, as a federated platform, facilitates federated collaborations among different parties by providing various options to use the platform based on data owners' consent. One of the significant concerns of different hospitals and institutions that have access to patients' sensitive data is regarding the patients' consent. Once hospitals want to use patient data on the FeatureCloud platform (i.e., the local controller), they should ensure they have the right to use it in a specific WF with specific apps. While there might be legal bases other than consent which justify ML studies with FeatureCloud, if the legal justification is based on consent, hospitals have to acquire the patients' consents and store them in their local databases or physical files. When conducting ML studies with FeatureCloud, for documentation purposes the FeatureCloud consent management system requires hospitals to submit the required (digital or digitized) consents to the WF, along with the data. It is important to note that FeatureCloud only submits hashed consent and data commitments to the blockchain while the local copies of the files remain within the hospitals, thereby making the process auditable in the future. While securing consent management processes is explained in detail in D6.3 and D6.4, in this deliverable, we will elaborate on how the blockchain-based components for managing consents and ML studies are integrated into the platform. This way, the participants (e.g., hospitals) can use the platform to interact with the blockchain for committing operations on consents and ML studies’ executions. The blockchain will serve as an audit trail necessary for securing the audit processes.

4.1 Consent management in FeatureCloud

In federated collaboration, two or more participants may join to run a workflow in the FeatureCloud platform. In general, as a registered user in the platform, each client should run a local version of the controller, which is the only FeatureCloud component in participants' machines to have internet access and capable of communicating data across the platform. One of the participants should take the role of coordinator to assemble a workflow through the front end, using the available apps in the app store, and invite other participants to join. Each participant will run a local app instance in a Docker container, which is completely isolated for security and privacy reasons. For running the workflow, each client should add its own data to the controller to be available for apps. For hospitals and institutions that use patients' data for federated collaboration, the GDPR and/or national data protection laws may require obtaining patients' consent and keeping track of processing activities that the data were submitted to.
Patients' data is always stored in the hospitals or institutions where they were collected, and the GDPR requires hospitals to refrain from any processing of that data that may compromise privacy. In that regard, deploying the patients' UI on the FeatureCloud platform infringes on patients' privacy due to the necessary identification process that asks users to sign up into the website to interact with the consent management system. On the other hand, hospitals already have access to patients' data and personal information to provide the required credentials for the patients and capture their consent without revealing their identity to third parties. Accordingly, as depicted in Figure 1, all participants will deploy their own patient's UI to collect and manage consents and enable patients to

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

connect to their local databases. For that purpose, they can use the exemplary UI described in this deliverable as a blueprint or any other appropriate solution.

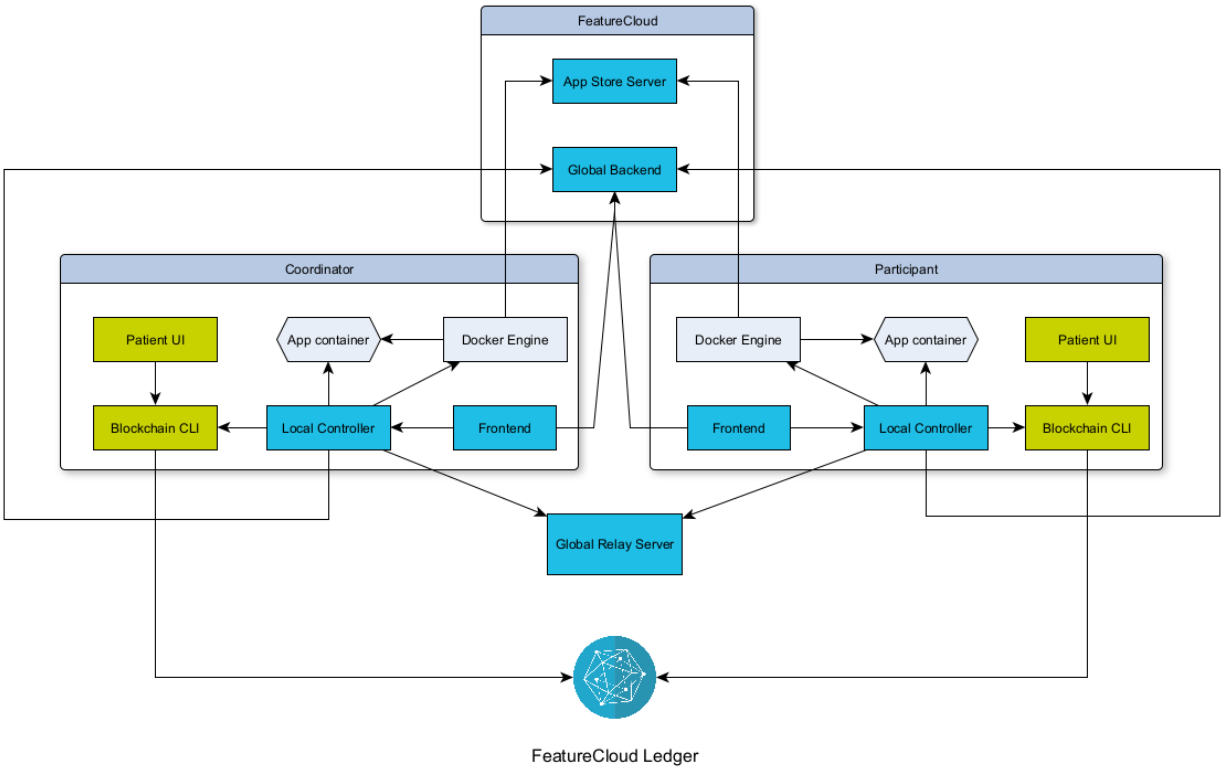


Figure 1. How the architecture of the FeatureCloud platform supports consent management. In federated collaboration for two clients, each client will run the same components of the FeatureCloud platform while deploying a local patients' UI to capture and store the patient’s consent.

Multiple factors prevent us from providing a general UI that can be deployed in all hospitals. For instance, different hospitals use various back-end and front-end technologies in their platform that require compatible technologies to design, implement, and deploy the UI. Moreover, many hospitals have strict security considerations in place, further complicating UI integration in hospital platforms. Meanwhile, each hospital may have a different consent template, and thus providing the same content for all institutions may not be legally feasible.

For running a WF, input data and consents should be provided to the controller; the controller hashes the consents, data, each app's Docker image ID, config file, and intermediary results. Then it submits the hashes to the FeatureCloud Ledger through the Blockchain CLI (see deliverables D6.2 and D6.3). Meanwhile, for auditing purposes, the controller provides a copy of the following, raw and hashed, for the clients at the end of execution:

- Input data
- Patient's Consent.
- Intermediary results.
- The config file.
- App versions

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

Hospitals are expected to dump a copy of the data in their local database to make the data consent auditable. The rest of the WF execution is the same as when the consent management system is not used. If the WF execution is interrupted or fails, corresponding data, consent, and files used by part of the WF (applications) will be provided as the output for clients and submitted to the blockchain. In the meantime, clients can repeat the entire WF execution at their discretion.

4.2 Consent management

Patients' consent can be acquired by hospitals as a paper or digital consent. In either case, the consent should be preprocessed and hashed before submission to the blockchain system. The blockchain system will keep track of patients’ data usage in different WFs. This will later allow external auditors to check the consented data that was used in previous federated collaborations. In the FeatureCloud platform, the controller is responsible for hashing the consent, so it needs to receive consent in a specific file format, e.g., JSON or PDF, to apply hashing algorithms. In the case of paper consent, we assume that hospitals will first digitize the consent. The digital consent can be created using the patients’ UI deployed on the hospitals' platforms. Patients are also capable of updating and revoking the consent through the UI which will be stored in local databases and its commitment submitted to the ledger.

Using the patients' UI and Blockchain CLI (Consent Management Services), the hospital registers a new local patient identifier PID, e.g., Patient p0001, which will be the root for all subsequent operations for that patient. The patient’s handwritten signature on paper consent will be enough to bind the PID to the patient itself. Then, the patients’ UI will submit the new consent commitment to the FeatureCloud Ledger through the Blockchain CLI. Hospitals are expected to store the signed binding locally. Note that it is possible to create multiple PIDs for the same patient (e.g., one per consent) to prevent linking its different consents.
The hospital must inform the patient about future research/studies and request consent to use their data. The scope of the consent can be broad, e.g., where the patient agrees to consent for all cancer research studies. The hospital will create a consent identifier (CID), e.g., Consent C0001, that is linked to that PID. The consent management workflow is described with the following steps (also see D6.4. for further details)
Patients can submit, update, or revoke their consents, and their commitments will be automatically submitted to the blockchain by the UI and Blockchain CLI. On the other hand, for using the data in a specific ML study, a FeatureCloud workflow including specific apps, the corresponding consent will be hashed alongside other workflow input and submitted to the FeatureCloud Ledger.

As shown in Figure 2, the hospital’s IT personnel is responsible for filtering the consented data and providing it to the project coordinator, who will create or join the WF and upload the data. The controller will also hash the data mentioned above and upload the hash to the FeatureCloud Ledger via the Blockchain CLI component (ML Study Services). At the end of the WF execution, whether successful or interrupted, the controller will dump a copy of the data, consent(s), WF configuration and results to the local database in order for the auditing process to be able to reproduce the same results, thus proving that the study was carried out by using only consented data.

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

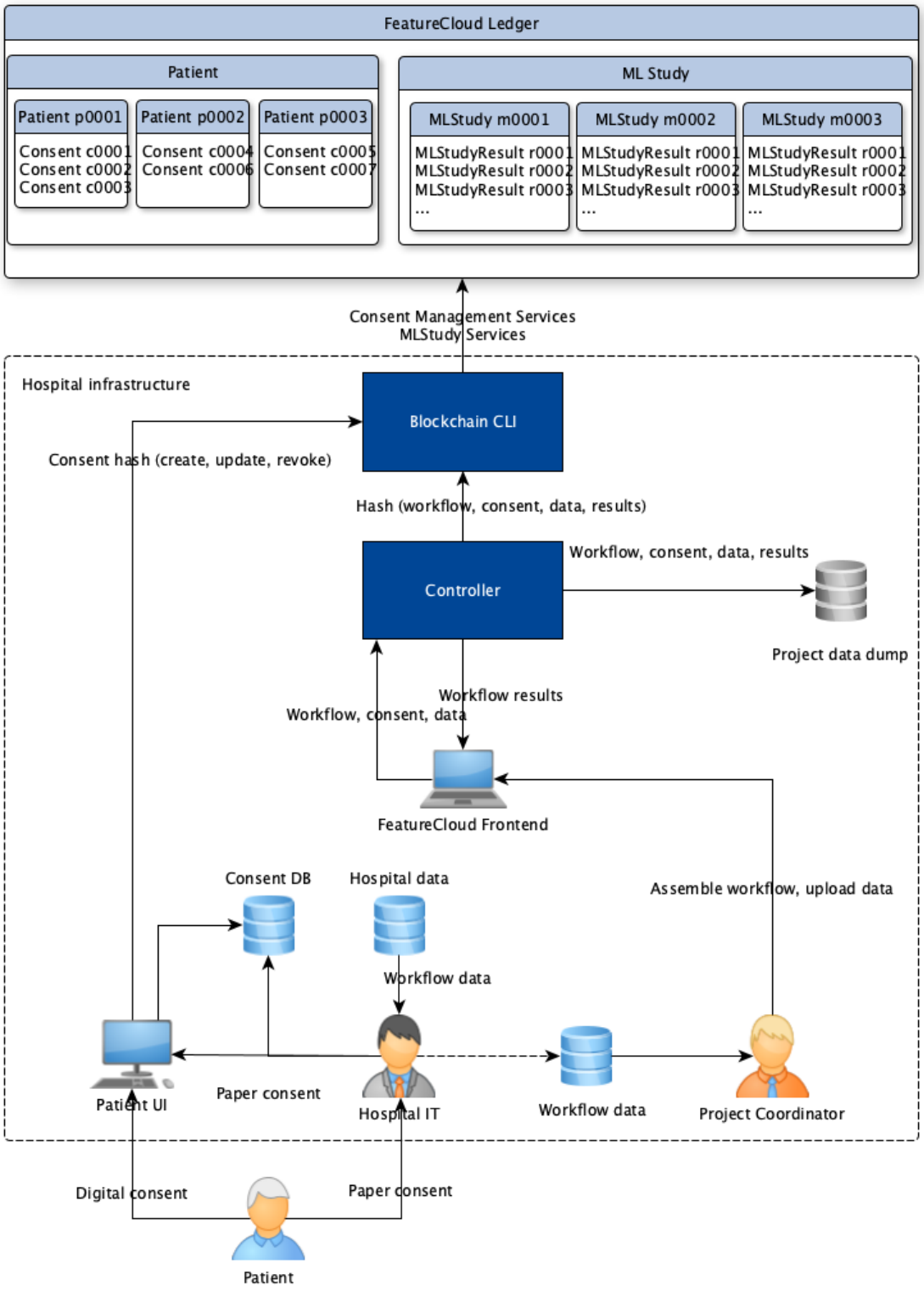


Figure 2. Capturing and incorporating patients' consents. Patients will submit their consents through the UI or on paper, handing it over to hospital IT. The FeatureCloud controller will process the consents and data in different WF stages and submit the required commitments to the FeatureCloud blockchain using the Blockchain CLI. The FeatureCloud blockchain will keep track of all patient operations on consents (consents lifecycles) as well as their use for specific ML studies separately.

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

4.3 Auditing the executed workflows

The audit process must ensure that conducted research is based on consented data. In the research phase all data related to WF and consents are saved in a project data dump locally, in the hospital infrastructure, alongside saving consent hashes and hashes based on study metadata in FeatureCloud Ledger. The auditor is then able to re-run a workflow and compare results. In a deterministic outcome, the compared hashes of the research and audit run must be the same. In a non-deterministic scenario, the results must be comparable. Note that the blinding factors (e.g., salt) used within hashes are also stored locally as they are necessary for the audit.

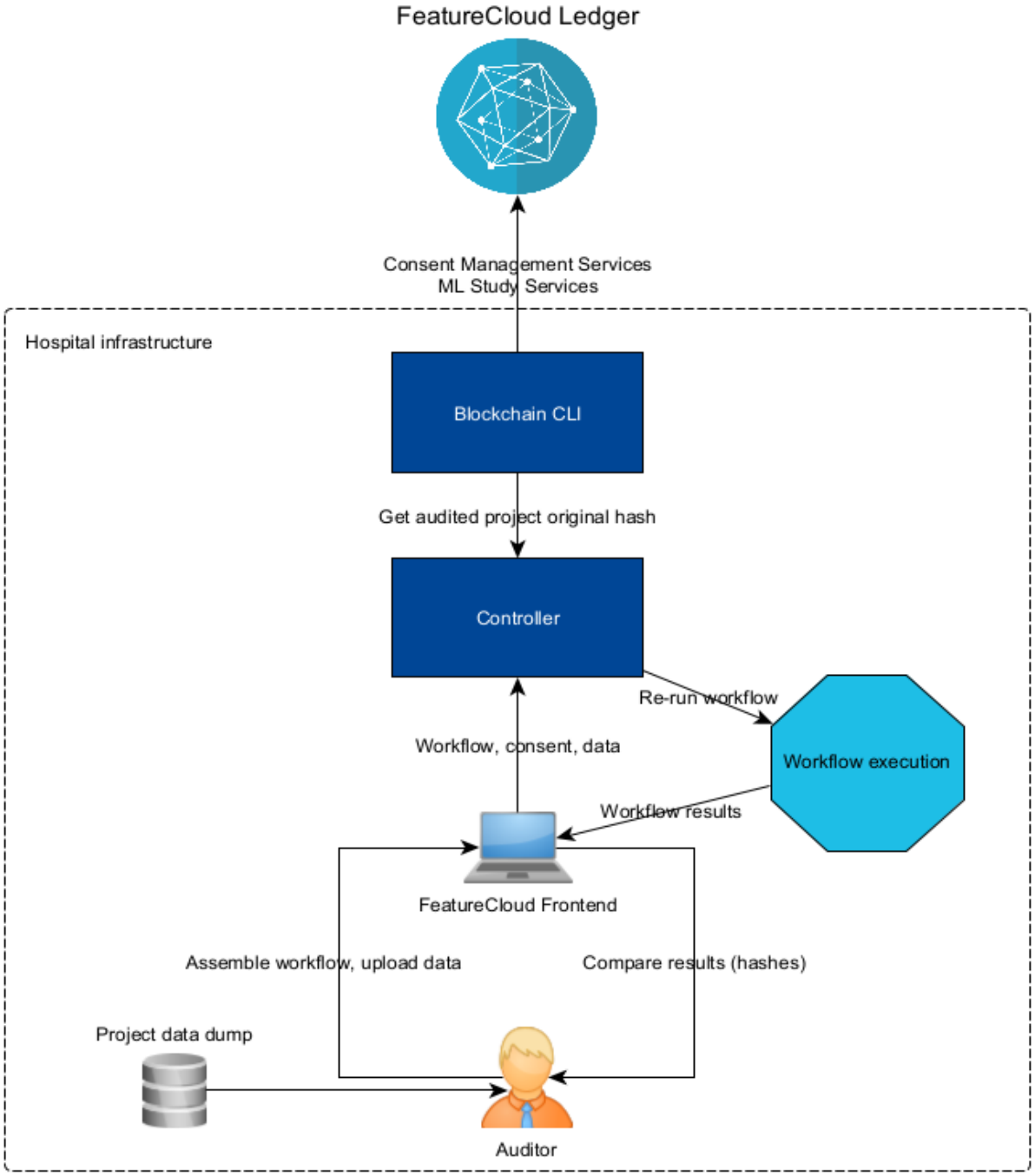


Figure 3. Audit process in FeatureCloud.

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

The auditing process is quite independent of integrating the blockchain consent management system into the FeatureCloud platform because it starts once the external auditor refers to a hospital after employing FeatureCloud for federated collaboration. The data and study are hashed, and the hash is stored in the blockchain system by the controller, which needs to be retrieved using the Blockchain CLI. The process is covered in detail in D6.4. In a WF, it is assumed that the participant will act on behalf of the patient to update or revoke consents on the blockchain. When an auditor examines a data input used for a specific study, they will verify whether or not it was consented at the time of the study. Therefore, the auditor will first inspect the “consent identifier” CID in the consent that was presented by the hospital (which is embedded and signed by the patient within the consent).

The auditor will verify the following:

- If the CID in the digital consent is somewhere on the blockchain.
- If the CID is present, they will identify the root in that chain (PID)
- They will check the local signed binding of the WF, which binds that PID to a global identifier ID_g (an actual person).
- They will compare the timestamp of the CID to that of when the study was conducted.

The auditor will then re-run the WF based on the initial data dumped when the original project run was carried out and compare the resulting hash to the saved one. In a non-deterministic scenario, the hashes won’t match, so the study results will be compared to each other.

5 Results

5.1 Patients consent management UI

The Patients UI allows patients to give, update and revoke consents to use their data for research purposes. Therefore, the UI contains a form that the patient needs to fill out. This form includes personal data, the timeframe of the given consent (start and expiration date), as well as the terms of the consent (Figure 4). From this form, a PDF file is generated that the patient can download and sign (Figure 5).

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

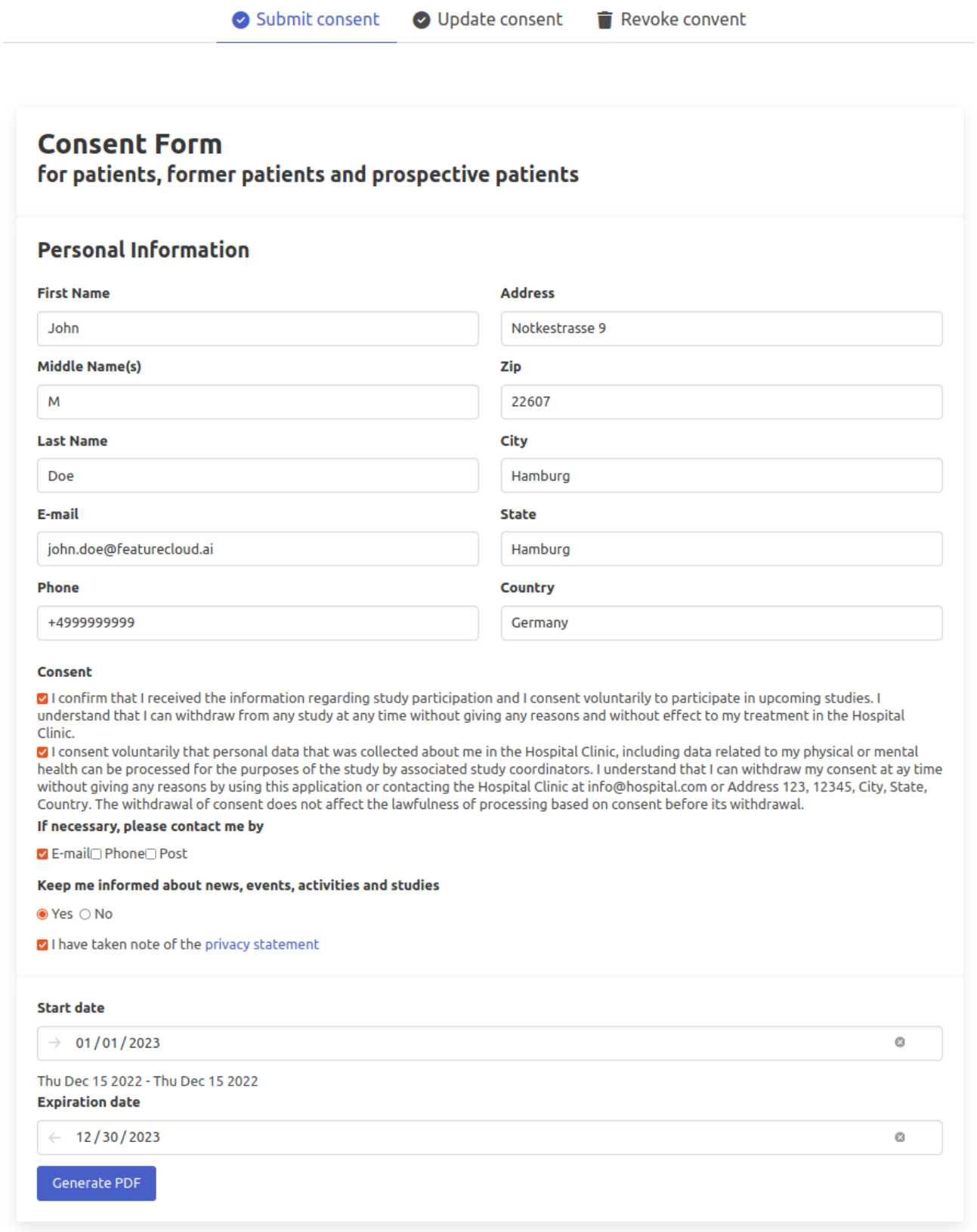
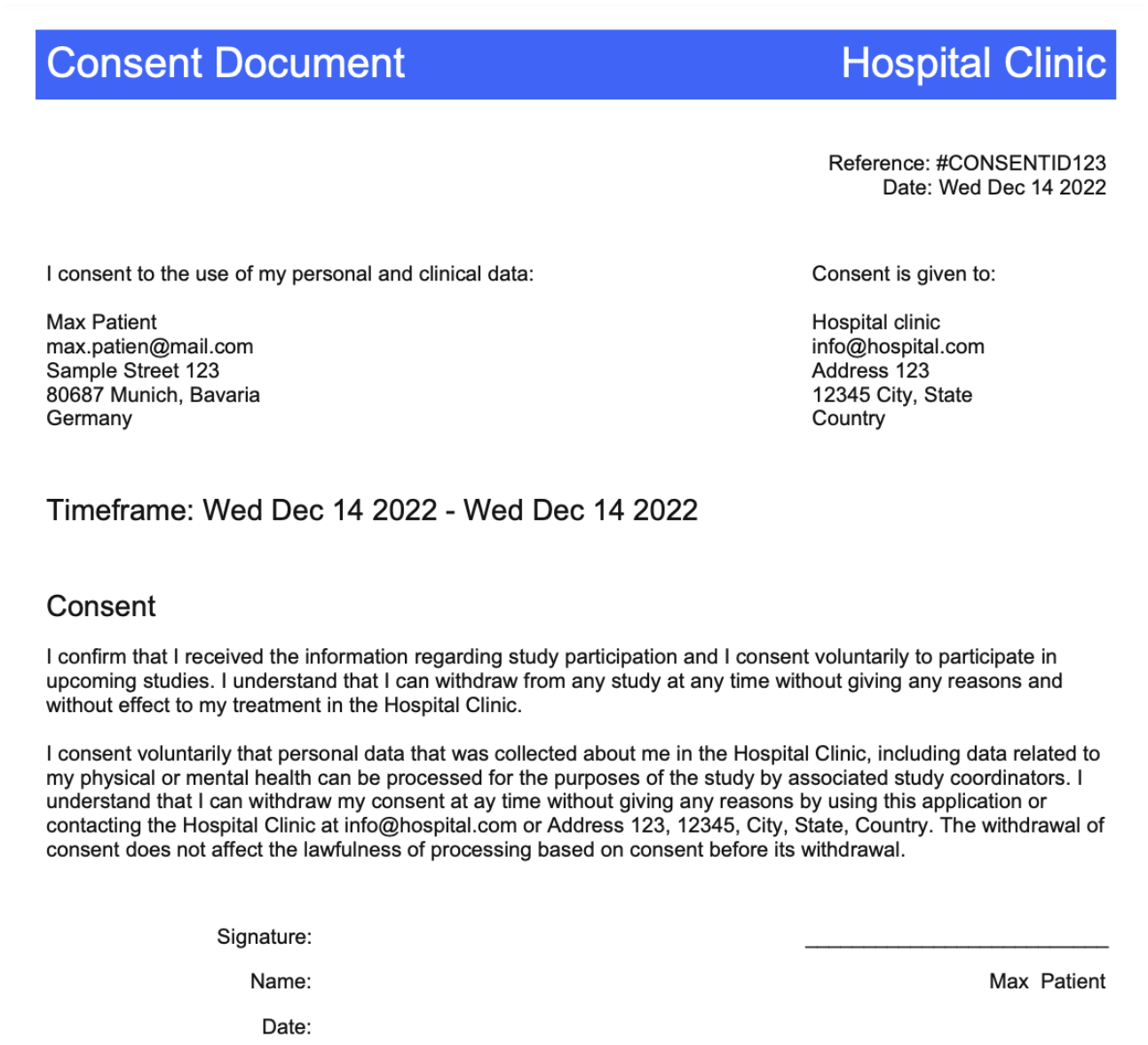


Figure 4. Patients’ UI for consent management.

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



Consent Document Hospital Clinic

Reference: #CONSENTID123
Date: Wed Dec 14 2022

I consent to the use of my personal and clinical data:

Max Patient
max.patien@mail.com
Sample Street 123
80687 Munich, Bavaria
Germany

Consent is given to:

Hospital clinic
info@hospital.com
Address 123
12345 City, State
Country

Timeframe: Wed Dec 14 2022 - Wed Dec 14 2022

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 ay 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.

Signature: _____

Name: Max Patient

Date:

Figure 5. Auto-generated consent form.

5.2 Interaction between FeatureCloud and blockchain

The interaction between FeatureCloud components that run inside a hospital and the FeatureCloud Ledger is implemented by the Blockchain CLI component. In the auditing process, the hash of data, consents, etc., will be compared with the corresponding hashes in the blockchain system. In the case of using different approaches for hashing, the outcome will not be the same, so the hospital should use the method that was used for the hashes in the blockchain system. Since the ML study results will be hashed via the controller, we provide options for hashing that can be extended in the future. In this way, we ensure hospitals can deploy a patients' UI that uses a convenient hashing approach for the hospital from a possible set of options covered via the controller. Even though in

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

the auditing process, hospitals can hash the data, consents, etc., again, for the sake of consistency, we recommend both patients' UI and the controller use the same functionalities for creating the hashes.

5.2.1 Patients’ UI

The patients’ UI uses the “registerPatient” command in the Blockchain CLI to register a patient with an auto-generated ID in the blockchain:

```
$ registerPatient p0001
```

This part of patients' UI is subjected to the technological aspect of a hospital's infrastructure, and it should be designed and implemented, correspondingly, and deployed by the hospital to allow the signup process for patients and the following patient registration in the blockchain. The patients' UI supports submitting, updating, and revoking operations for consent using the CLI as follows:

```
if (operation === "submit") {
   this.blockchainService.submitConsent(pid,
this.getHashForFile(consentFile));
} else if (operation === "update") {
   this.blockchainService.updateConsent(pid, cid,
this.getHashForFile(consentFile));
} else if (operation === "revoke") {
   this.blockchainService.revokeConsent(pid, cid,
this.getHashForFile(consentFile));
}
…
private getHashForFile(path: string): string {
// Hash the consent file
let hasher = new ParallelHasher(path);
hasher.hash(fileBlob).then(function(result) {
 return result;
});

return '';
}
```

5.2.2 The controller

When a new study is initiated, the controller hashes all data related to the WF as well as the consents and adds a new ML study in the FeatureCloud Ledger. This is triggered by the controller and executed by the Blockchain CLI through the MLStudy Services. The controller uses the following code to read and hash the data:

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

```
// Return hash of the content of a directory
func getHashForPath(path string) string {
  _, err := os.Stat(path)
  if err != nil {
     logger.Error(DATA, "0", "Directory not found %s", err.Error())
  } else {
     dirHash, err := hashdir.Make(path, "md5")
     if err != nil {
        logger.Error(DATA, "hash", "Hashing failed for directory %s: %s",
path, err.Error())
        return ""
     } else {
        return dirHash
     }
  }
  return ""
}
```

Different data will be hashed and submitted to the consent management system at different WF execution states by different participants. For each WF, a specific ML study entry should be created in the blockchain to be used by participants to submit their results. Such operation demands the controller of the coordinator to change the blockchain states to allow or deny the submissions at the start or end of the WF, correspondingly.

Table 1. WF execution states versus blockchain consent management system states.
The WF life cycle in FeatureCloud includes different states which demand different interactions with the consent management system. Each state in the WF execution lifecycle triggers a specific state in the consent management system using a specific command in Blockchain CLI. The local controller of the coordinator invokes Blockchain CLI commands during the WF’s lifecycle to support the submission of hashed consents into the blockchain system.

WF state	Blockchain CLI state	Blockchain CLI command	Operation
init	N/A	N/A	N/A
ready	announce	announce	Announce new ML study
prepare	announce	changeState, submitResult	Submit data in the consent management blockchain (consent hashes, metadata, data)
running	execution	changeState, submitResult	Submit intermediary results
stopped	postprocessing	changeState, submitFinalResult	Submit final results
finished	postprocessing	changeState, submitFinalResult	Submit final results

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

The first state of the WF is `init` where the WF is finalized by the coordinator. In fact, the coordinator creates a WF by selecting different apps in the app store, and then creates and shares tokens with other participants. Once all the participants join, the WF enters the ready state. At this point (WF configuration finished), the “announce” operation takes place. For announcing the start of a ML study to the blockchain system, the controller uses project ID as study ID (SID), e.g., m0001 in Figure 2.

```
announce project_id
```

The “prepare” state in the WF invokes the data upload from each participant. From this point, participants can submit their hashed studies, metadata, and consent into the blockchain system under the study, e.g., m001.

```
submitResult project_id result_id hash
```

When data upload is complete, the project goes into the running state, and the “changeState” operation occurs in the blockchain, transitioning into “execution.”

```
changeState project_id execution
```

While the project is running, intermediary results will be submitted to the blockchain.

```
submitResult project_id result_id hash
```

Once the WF execution is finished, it enters the “finished” state. In case of an error, the coordinator stops the execution, and WF enters the “stop” state. In both cases, the coordinators’ local controller changes the study's state to “postprocessing” to end the possibility of adding new hashes into the study.

```
changeState project_id postprocessing
```

In the “postprocessing” state, the hashes of the final results are uploaded into the blockchain.

```
setFinalResult project_id hash
```

For a better understanding of the project WF and the interaction between the controller and FeatureCloud Ledger, consult the sequence diagram in Figure 6.

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

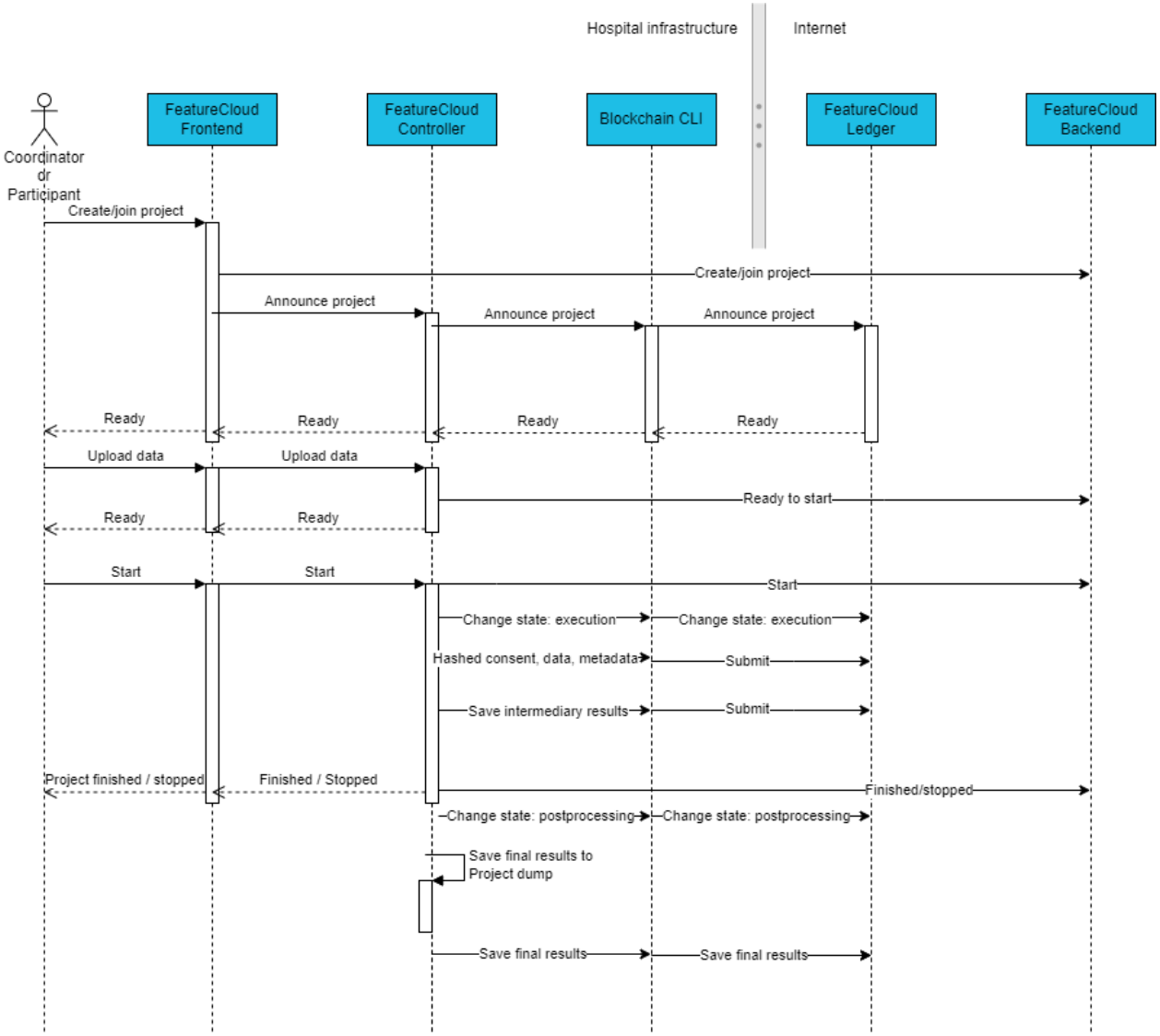


Figure 6. Interaction between FeatureCloud components and the FeatureCloud Ledger during WF execution

The controller submits the results to a study at the start of the WF for the first app:

```
submitResults project_id r_id timestamp dirHash
```

The controller submits the final results of a study after execution of the last app in a workflow :

```
submitFinalResults project_id dirHash
```

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

6 Open issues

Theoretically, the auditor can run the WF to reproduce the results, hash the results and compare the hashes with the corresponding ones in the blockchain system. The audit process can succeed for deterministic models, where it’s possible to reproduce the results provided with the same data and configurations. However, for many stochastic models, where the behavior of the model may change regardless of repeating the experiment with the same set of input data and configs, the auditor may not be able to reproduce the same results and, in turn, the same hash to compare with the one on the blockchain system. Moreover, many privacy-enhancing technologies are based on applying random noises to either data or models, which further exacerbates the stochasticity of the model, e.g., Differential Privacy (DP). Therefore, supporting data reproducibility of stochastic models is currently still an open research challenge in the community.

7 Deviations (if applicable)

The patients' UI is not deployed on the FeatureCloud platform as it requires an identification process to serve patients and violates user privacy. FeatureCloud therefore provides a generic UI that can be adopted and integrated into hospital platforms based on the scope of consent, the legal content of the consent, and the technical aspects of the hospital's infrastructure and software. FeatureCloud demonstrates that such UIs can be integrated into the FeatureCloud platform using the Controller, FeatureCloud CLI, and Blockchain CLI.

8 Conclusion

The integration of a blockchain consent management system into the FeatureCloud platform substantially facilitated design, implementation and deployment of the patients' UI into hospitals platforms by providing a generic adoptable UI and demonstrating its possible integration. It also enables tracking the WFs and studies for which specific patient data has been used, which in turn increases accountability and transparency and thus encourages more patients to submit their data consent for federated learning or analysis in transparent collaborations.

9 References

n.a.

10 Table of acronyms and definitions

CLI	Command Line Interface
concentris	concentris research management GmbH
GND	Gnome Design SRL
ML	Machine Learning
MS	Milestone
MUG	Medizinische Universitaet 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

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

	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
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)

The code block below shows the main steps of the process to store WF results:

1. Pre-condition: The last app in the WF stores the results in its output volume
2. Results are moved from output volume to a temporary directory
3. Contents of the temporary directory are zipped and stored in the controller’s working directory

```
tempDir, err := dm.TempDir("workflow-results")
…
internalTempDir := dm.MustLocalPath(tempDir)
hostTempDir := dm.MustOrchPath(tempDir)
…
err = w.orch.MoveDataToHost(hostTempDir, volumeName)
if err != nil {
  logger.Error(FLOW, strconv.Itoa(projectId), "Could not move data from volume
to host: "+err.Error())
  return err
}

…

resultsDir, err := w.ensureRunDir(projectId, w.lastInfos[projectId].Run)
if err != nil {
  logger.Error(FLOW, strconv.Itoa(projectId), "Unable to create run directory
due to %s"+err.Error())
  return err
}
resultsFileName := resultsDir + FormatResultsFileName(projectId,
w.lastInfos[projectId].Run, step)

err = storeResultsFile(internalTempDir, resultsFileName)
if err == nil {
  logger.Info(FLOW, strconv.Itoa(projectId), fmt.Sprintf("Stored results for
project %d, step %d", projectId, step))
}
```

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

```
…

func storeResultsFile(internalSourceDir string, fileName string) error {
  buf := new(bytes.Buffer)
  zipWriter := zip.NewWriter(buf)
  util.AddFilesToZip(zipWriter, ensurePathSuffix(internalSourceDir), "")
  err := zipWriter.Close()
  if err != nil {
     return err
  }

  err = ioutil.WriteFile(fileName, buf.Bytes(), 0644)
  return err
}
```

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