applied sciences

MDPI

Article

FLIP: A New Approach for Easing the Use of Federated Learning

Borja Arroyo Galende [1,*], Silvia Uribe Mayoral [2], Francisco Moreno García [1] and Santiago Barrio Lottmann [1]

[1] Escuela Tecnica Superior de Ingenieria de Telecomunicacion (ETSIT), Universidad Politécnica de Madrid, 28040 Madrid, Spain
[2] Escuela Tecnica Superior de Ingenieria de Sistemas Informaticos (ETSISI), Universidad Politécnica de Madrid, 28031 Madrid, Spain
[*] Correspondence: borja.arroyog@upm.es; Tel.: +34-910672632

Abstract: Over the last few years, there have been several attempts to provide software tools for the development of federated learning (FL) models. However, both the complexity of the concept itself and the high entry barrier of these tools have meant that their adoption has been limited. Considering the related benefits, especially in terms of preserving data privacy, and the need for this type of solution in specific areas where data sharing is impossible, not only from a practical point of view but also from a legal and even ethical perspective, it is necessary to advance in solutions that allow its use to be democratised and its deployment to be extended. With this objective in mind, FLIP (Federated Learning Interactive Platform) has been developed as a comprehensive, easy-to-use fully functional web-based FL network management platform that eases and accelerates the usage of federated datasets by researchers in real scenarios. In this sense, FLIP has achieved a SUS score of 84.64, confirming a high level of perceived usability as expected. Taking this into account, FLIP can help increase the productivity and adoption of FL by a wider audience.

Keywords: federated learning; web-based platform; usability; multinode deployment

check for updates

Citation: Arroyo Galende, B.; Uribe Mayoral, S.; Moreno García, F.; Barrio Lottmann, S. FLIP: A New Approach for Easing the Use of Federated Learning. *Appl. Sci.* **2023**, *13*, 3446. https://doi.org/10.3390/app13063446

Academic Editor: Giacomo Fiumara

Received: 9 February 2023
Revised: 1 March 2023
Accepted: 6 March 2023
Published: 8 March 2023

1. Introduction

The huge and rapid development of artificial intelligence (AI) during the last few years can be considered as the main consequence of the combination of two key factors: the increase in computational power at an affordable cost and the enormous amount of available data due, among other reasons, to the spread of the Internet of Things (IoT) [1]. In an ideal scenario without bureaucratic, ethical, or legal barriers [2], big data could be used to test and develop new tools, applications, and technologies without any kind of restriction and in a rapid way. However, as the gathering and storage of sensitive data is increasing, the need to define new collaborative frameworks that ensure their privacy while allowing their use is vital for not limiting their impact [3].

In the context of AI, Federated Learning (FL) has been mainly defined with this purpose, that is, to allow the use of massive data based on a privacy-preserving mechanism [4]. Different domains, such as healthcare [5–7], especially for dealing with critical situations, such as the COVID-19 pandemic [8]; surveillance [9]; or industry [10] have benefited from the development of solutions based on this approach. However, while there are some examples of different FL platforms developed both by academia and industry [7,11], they tend to be complex solutions with a significant learning curve that requires high IT expertise [12], which limits their wide-scale adoption.

In order to contribute to reversing this situation, in this paper we present FLIP (Federated Learning Interactive Platform) as a novel approach for an easy-to-use web-based platform that allows real FL networks creation and FL models management, including training and execution. It is specifically focused on reducing the AI engineering load related to this kind of deployments and minimising current barriers related to the technical knowledge needed for creating FL solutions in real scenarios, and thus boosting FL adoption. Our evaluation, based on complete functional validation and a SUS questionnaire among 14 users, demonstrates that FLIP enables a fully functional FL system with a high perceived usability level, close to 85%, which can be considered as "acceptable" in the acceptability scale, and above industry standard for the industry benchmark.

The rest of the paper is organised as follows: Section 2 is devoted to analysing the background of the top FL software solutions currently available in both industrial and research fields, with the aim of identifying the main problems and shortcomings that are hindering the adoption of this AI strategy. According to this analysis, this section also presents the core contributions proposed by FLIP. Section 3 is intended to describe our approach, placing special focus on the developed architecture, as well as on aspects related to the network creation and the FL models management. Section 4 presents the functional validation of the solution together with the verification of the achievement of the initial objectives, that is, the analysis of the perceived usability of FLIP. Finally, Section 5 presents the conclusions together with a review of the main limitations and future work for the platform.

2. State of the Art and Contributions

As mentioned above, in recent years, the capacity for data generation, storage, and computation has increased significantly [13]. Regarding this, the concern for privacy in the transmission of raw data has led to the development of new strategies that make use of local capabilities to avoid such transmission and, thus, limit the security breaches that it could entail, but without renouncing the use of massive data in obtaining machine learning models. This is the premise behind FL, whose main objective is to explore the training of statistical models directly on remote devices [14].

There are several software alternatives that support FL in research and industrial fields. In the research domain, LEAF [15] provides sample implementations and simulates statistical heterogeneity with federated datasets, but it comes with a lack of deployability due to its complexity of use. TensorflowFederated (TFF) [16] is also a research-orientated platform with two layers of APIs, but, on the one hand, it does not provide real FL functionality, since it only simulates the multinode setup in a single machine, and on the other hand, it requires using (and knowing) Tensorflow [17] and Keras [18] as the basis for producing algorithms. Similarly, a more recent approach is EasyFL [12] which is defined as easy-to-use software that sequentially executes a simulation on a single machine but does not allow real deployment of the FL network. As can be seen, these platforms lack the capability to build FL prototypes and run real FL in production. Moreover, except for EasyFL, the other approaches are not user-friendly enough to help with FL use. In the industrial field, Fate [19] and PaddleFL [20] are deployable with containerisation and are designed for real FL scenarios, but they are complex systems for beginners that require deep knowledge of deployment and usage. Fed-BioMed [21] targets hospitals and clinical studies, but presents some limitations in terms of generalisation to other cases. Finally, Flower [22] is a complete platform that not only supports simulation, but also executes real nodes in a federated setup, while it is model-agnostic, but it does not provide a user-friendly way to interact with to create the FL network and algorithms. In this case, industrial approaches allow the deployment of real FL in production, but they lack usability for widespread adoption. Table 1 summarizes the information presented in this paragraph for the most promising solutions.

Table 1. Comparison between several near-production-ready frameworks and our solution.

Frameworks	Docs: 1–5	Containerized: Y/N	Extendability: 1–5	Graphical User Interface: Y/N	Encrypted Distributed Communications: Y/N	Production Ready: Y/N	ML Framework Agnostic: Y/N
Fate	3	Y	4	Partial	Y	Y	Partial
EasyFL	4	Y	4	N	N	N	N
Fed-BioMed	4	Y	3	N	Y	Y	Partial
Flower	4	N	5	N	Y	N	Y
FLIP (ours)	-	Y	5	Y	Y	Y	Y

According to this analysis, the main challenge regarding current FL software solutions is to be able to provide a fully functional platform for deploying FL models in real networks in an easy manner. Taking this into account, our evaluation demonstrates that FLIP can be considered as a usable tool to create FL networks and then use them to train FL algorithms as expected. To do so, our main contributions can be summarised as follows.

- We provide a platform that is capable of deploying a real multi-node network for FL training and execution.
- We identify two different roles with regard to the usage of the platform in order to simplify its operation: the platform administrator and the data scientist. The former is in charge of deploying the different nodes, providing the access permissions, and managing the FL network, while the latter just need to be focused on the development of the FL algorithms and then to seamlessly run them in the network.
- The proposed platform is based on web interfaces to facilitate user interaction.
- All the modules of the platform are easily deployable, the administrator should set up an environment file (.env), and a bash script takes care of the rest.

Following this approach, the intended ease of use of FLIP is reflected in the results obtained for the SUS questionnaire, which is explained further in the fourth section.

3. Design and Implementation

The first step in designing a software solution is the definition of use cases. Use cases can be interpreted as a set of holistic functional requirements that ignore the actual implementation of the system and focus only on the interactions a potential end user may have with the system. As a FL system that should manage both the node federation itself and the execution of federated tasks, the following use cases are expected:

- Plan/Schedule federated task: the creation of a new federated task (FL task) relates to the definition of the parameters required for performing a training. The crucial parameters to define are the model, the data source, and the source code of the algorithms for the flower network and strategy (server and client).
- Access training results: once training has finished, the updated model and the files related to execution must be available for the user.
- Monitor federated task queue: users can track the list of federated tasks that are awaiting execution, as they have been scheduled, but another training is being performed already.
- Cancel a federated task: a task that was already scheduled but not finished can be cancelled if the execution is not relevant any more.
- Validation of algorithms uploaded for federated task: as the source code is requested to establish the federated training, an inspection of the code must be performed to avoid malignant intentions must be performed.
- Manage users and authentication: access to the platform requires identification to track user activity and filter the content for each profile, and the management of these users and their roles is also required.

- Manage client nodes and addition: configuration and addition of client nodes to the federated network is necessary to dynamically add new data spaces.
- Manage environment and setup: initial setup and configuration must be completed previously in order to deploy the platform and being able to work with it.

It is also important to note that the main and basic functionality of the platform is to orchestrate FL tasks. To coordinate this, tasks are executed sequentially in the order in which they were stacked in the queue, to avoid the use of shared resources that can disrupt the network. Once the training plan is defined and the federated task is scheduled, the uploaded algorithms must be validated in order to queue the federated task.

To setup a federated network in which FL techniques can be applied, several factors must be taken into account. One of the most relevant issues is the need of integrating the data science process to plan the training (define the model and the dataset/datasets used) with the network and system resources administration, as these resources should be distributed correctly. In this sense, the requirement of two profiles or roles is designed in the aim of separating the role of an "Admin" (system and network administrator) and the users (data scientists). This approach allows each profile to handle their adequate task and to optimise the coordination between them.

The list of use cases depending on the role is shown in Figure 1 and it is important to mention that the setup and deployment of the platform at each of the nodes involved in the federated network must be carried out by an administrator. Normal users can monitor the queue of FL trainings scheduled, the history of past trainings, the status of a specific training and the cancelation of the training scheduled, and the download of the results of the training once finished. Administrator user permissions include the ability to manage users and organisations, manage the addition and configuration of new client nodes, and the task of validating the algorithms uploaded by users for the FL training.

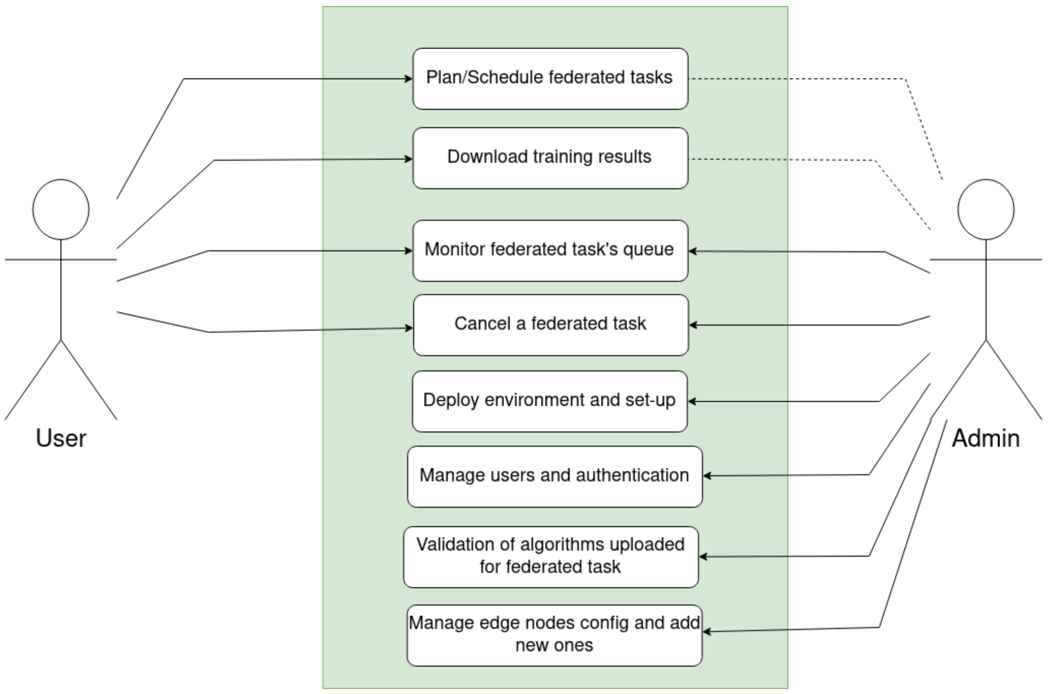
Plan/Schedule federated tasks
Download training results
Monitor federated task's queue
Cancel a federated task
Deploy environment and set-up
Manage users and authentication
Validation of algorithms uploaded for federated task
Manage edge nodes config and add new ones
User
Admin

Figure 1. Use cases diagram divided into "Admin" users and normal users.

3.1. Architecture

The modules that conform the platform define a set of interrelated services that cover the various functionalities expected for such a system. All these modules are depicted in Figure 2, and are distributed into two main actors, **server** and **client**, the first being the manager or coordinator of the federation; while the latter refers to a dataset storage place, where a replica of the global model is placed in each iteration. Both entities are detailed in the next paragraphs.

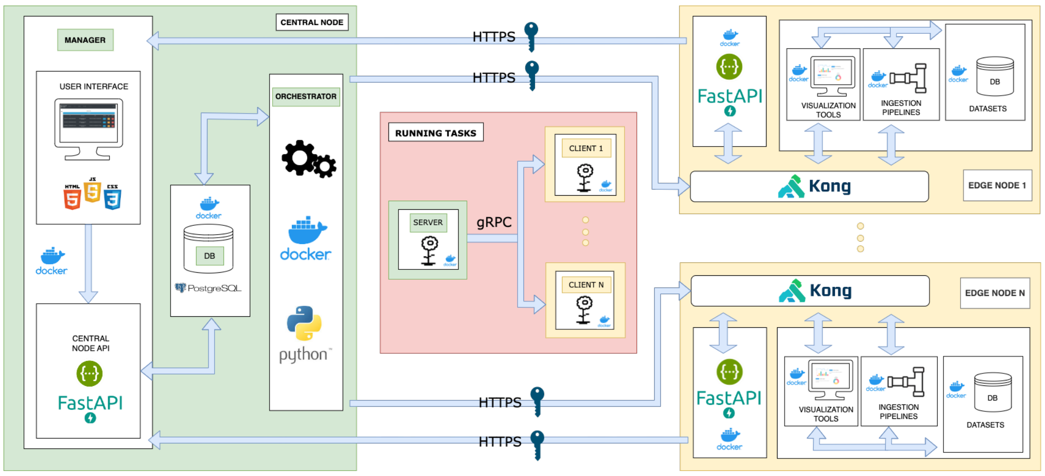

Figure 2. Responsibility-based design of the platform's architecture.

The server has the core responsibilities of the federation, it is the actor that enables manageability and usability to the system, and it is responsible for triggering upcoming federated tasks. It is made up of a manager, which is a web app with a back end directly communicated with the persistent storage service. The front-end (dashboard) together with the back-end provides user accessibility and manages the state of federated tasks. The back-end operates not only in terms of user-related actions, but also in terms of server-to-server synchronisation messages; thus, the status of a task is handled via this API as well. Moreover, the orchestrator handles and triggers all actions related to a federated task on the respective machines. It is responsible for launching computing containers on both the server and client nodes, and it also provides the queue mechanism that supports sequential execution of federated tasks.

On the other hand, the client is supported by a network-securing actor or proxy and an API that exposes the relevant actions that can be triggered on that node. The data are also part of the node, and all the necessary integrations for data access and data harmonisation are external dependencies on top of which the platform should operate.

Finally, in the middle of the figure, the task subspace is defined as a set of computing entities that operate together to run a federated tasks. Bear in mind that this subdomain is an abstraction because each of the services runs on top of the node to which it belongs, but we wanted to emphasise that it acts as a whole and autonomous subsystem.

3.2. Functionalities

The two entities, server and client, have substantial behavioural differences. Most of the duties fall on the former, while the latter executes the data-exchange process. These functionalities are explored in further detail in the following two sections.

3.2.1. Server

- Define the training plan, including the ML (Machine Learning) model that resides at runtime on the clients, along with the aggregation strategy, the federated dataset to be used, and the training flow.
- Orchestrate the deployment of compute containers on client nodes.
- Monitor execution on each node using exception handling when necessary.
- Change the execution status of a task to cancelled.
- Save the final aggregation status of the FL task.
- Send the aggregated model to the clients to start a new training round.
- Generate public documentation for the REST API following a standard such as OpenAPI.

- Provide a simple and functional user interface to ensure access to services without the need to use REST actions.
- CRUD operations on federated datasets.
- CRUD operations in users and organisations.
- Deployment scripts to facilitate the subscription of new nodes.
- Task statuses are stored in persistent storage.
- Incoming and outgoing communications are encrypted using a secure protocol.
- The aggregation strategy operates on data that can come from any model written in Python, independent of the library from which it comes.

3.2.2. Client

- Execute a Flower client associated with the subset of the federated dataset of that client node.
- End the execution of the Flower client.
- Clean up the compute container environment to allow a new task to be launched.
- Generation of an encryption key to authorise incoming communications from the server.
- Generate public documentation for the REST API following a standard such as OpenAPI.
- Incoming and outgoing communications are encrypted using a secure protocol.
- Containers running the Flower client are flexible with respect to the libraries used.

3.3. Platform Implementation

The technologies used enhance an isolated, microservice orientated, and secure framework for production environments that require the usage of federated machine learning solutions. These technologies both gather ready-to-use services and third-party libraries that conform the stack of open-source tools on which the platform relies.

As a first level of abstraction and influencing all the modules that compose the platform, Docker is selected as the tool that manages both images and containers. Hardware-independent, isolated execution provides a way to safely manage the resources via virtualisation. Not only does it enable one way to secure the execution of federated tasks, but it also protects the host from user code.

The high-level federated execution layer is implemented on top of the Flower library, whose functionalities guarantee high throughput of the computing modules and do not require specific libraries for producing the algorithms.

Communication is handled in two different ways depending on the objective and content of the communication. If the message conveys model parameters, the protocol used is gRPC (via Flower), which enables a huge gain in the (de)serialization stage. Otherwise, REST actions are enough to change the state of the nodes, either in user-to-server messages or server-to-server flows. Moreover, a graphical user interface (GUI) is featured via a web front-end, coded aside from the central API through Jinja templates.

The external endpoints are exposed via an API Gateway, Kong, which gathers all the different internal paths into a unique port, hence improving usability of the system. Both gRPC and HTTP/1 are handled via the Kong proxy mechanism.

3.4. Deployment and Execution Flow

Having in mind all the details covered up in the design up to this point, this section intends to describe the transitions that suffer the platform along a task execution. Figure 3 demonstrates these states with all the corresponding components that have already been described as part of the system, starting from idle and unconnected in the upper left corner and ending in the automated deployment of the computing subsystem in the desired nodes, as shown in the lower right corner. All these states are detailed as follows:

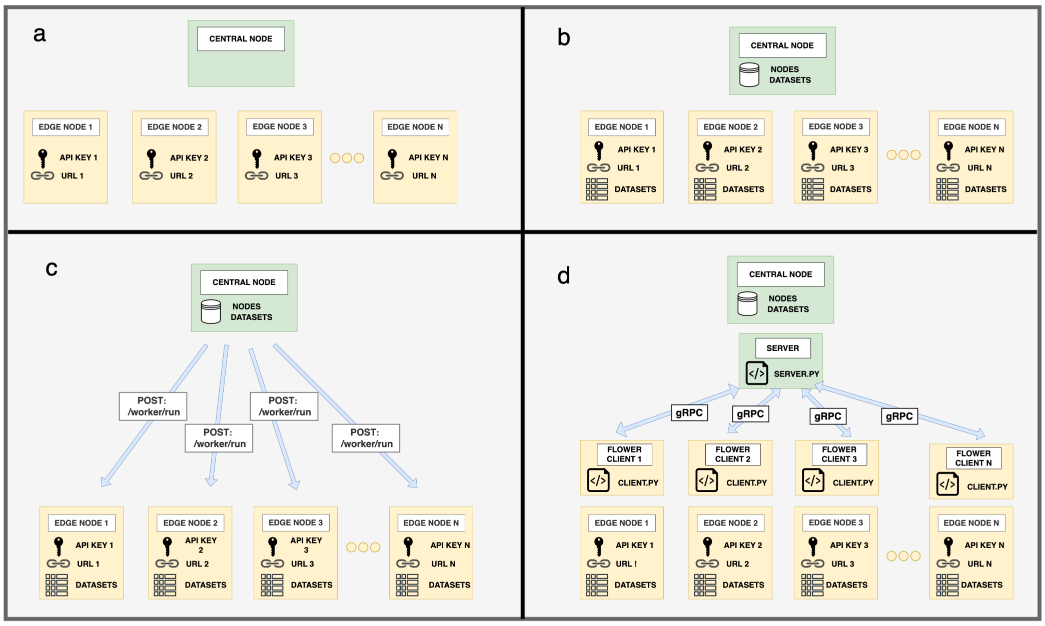

Figure 3. Changes in state that occur within the platform. (**a**) Unconnected system with all components ready to be included in the federation. (**b**) Connected components where the system acts as a whole and the information is stored within the central node. (**c**) Synchronisation of the idle system to a running state. (**d**) Task execution state with all computing containers launched and interconnected.

1. The first state, as already mentioned, corresponds to the deployment of each of the actors involved in the system, but there is no link between them. In other words, the server or manager is not aware of the existence of any client, and thus the federation is not set up yet.
2. The second state, situated on the upper right corner, shows that the information has already been added to the server, hence the federation has been established, and the system is waiting in an idle state in which there is no task running or planned to be run and just awaits a new incoming job.
3. The third state, located in the lower left corner, occurs when a new task has been received, and the orchestrator plans the execution to be triggered. A set of invocations targeting client nodes occurs, and all actors are aware of the new situation of the system.
4. The fourth and final state depicts the actual execution status of a federated job in which all the computing components have already started their TCP connections and are performing all the actions involved in the task.

Platform details from a software design perspective should be clear now. The next section focuses on the user actions needed to (1) deploy the system as an administrator and (2) launch and monitor a FL task as a researcher.

3.5. Platform Usage

As mentioned, the platform requires two different roles to properly manage use cases and comply with legal or ethical policies. The first role brings together all software management, including deployment and access granting among others, while the second focuses on the creation of federated machine learning tasks, either for training or validation of algorithms, which are handled by the platform automatically.

3.5.1. System Administration

Firstly, to deploy the system, an administrator should take over the process. It is important to bear in mind all regulations that could constraint the data environment. This is why, generally speaking, the system administrator and the data scientist should not be the same person. Moreover, our proposal is to establish a distribution of tasks among parties so that each party entering the federation provides an IT employee who can deploy the system. This avoids the need for signing contracts for external collaborators, thus reducing the legal overhead.

The steps that describe the deployment process, of which the first three do not need direct access to the servers where data resides, while the rest do, are as follows:

1. Download/create the Docker image containing the code for the federated server.
2. Introduce the spec into the .env file, based on a template provided within the software package.
3. Run the system as Docker containers.
4. Download/create the Docker image containing the code for the federated client in each of the data silos.
5. Introduce the spec into the .env file, based on a template provided within the software package.
6. Run the each client as Docker containers.

Once the system is deployed, all the activities performed by the administrator are handled via a web interface, in which the admin can easily manage users, validate FL tasks and manage nodes.

3.5.2. FL Research

As a prerequisite, the administrator shall grant access to the researcher in charge of designing and implementing federated algorithms. The researcher just needs to focus on two different aspects of the FL task:

- Encapsulate the algorithm as Python scripts following the Flower conventions and guidelines. This way, the researcher needs to provide a client class and a server class, where the first is responsible for interacting locally with data, while the second aggregates the model updates.
- Define the metadata associated to the execution, such as training rounds, libraries required, etc.

Note that all interactions a researcher performs with the platform take place through the web interface. There is no need for additional knowledge other than the coding skills required to run traditional machine learning experiments. In other words, the abstractions that the researcher should implement are straightforward.

Once the FL task is defined by the researcher and validated by an admin, the system queues it and eventually executes it without additional human intervention. After completion, the researcher can view and download the results via the website.

3.6. A Final Note on Platform Usability

Having the platform design in mind, we now cover the real usability and maturity of tools such us FLIP within the general ML picture. In this regard, a rule of thumb for knowing when to choose FL is, in our experience, the following:

- If there is no restriction in terms of data sharing and data fits in memory or can be processed in batches, a centralised version should be preferred. There is still a leap in difficulty compared to centralised approaches, and the results are not guaranteed to be as good due to data properties.
- If there are limitations due to policies such as the GDPR, relying on a data privacy-preserving mechanism such as FL is the unique feasible option. In this case, FLIP can enhance productivity in the research field due to its high usability.
- If there are no data sharing restrictions, but the data cannot be processed in a single machine, either because vertical scalability is not an option or due to time constraints, FL should be considered as an alternative. In this case, we assume that horizontal scalability is feasible for the expert and that the difficulties with respect to model aggregation in the central node are achievable, either because the data have a similar

nature (similar distribution) throughout all chunks or because there is a huge understanding of the process. In this case, there is no need to consider a system with security concerns such as FLIP, as the researcher can directly interact with the data. In this way, we recommend FLIP if the researcher lacks knowledge of how to run federated learning within lower-level platforms, such as Flower.

4. V&V: Validation and Verification of the Platform

In order to analyse the fulfilment of the main aspects that will be covered by this platform, we have defined a two-step confirmation process: first, we perform a functional validation of the framework with the objective of ensuring proper operation and then define a specific study to verify the usability level with a set of potential users.

4.1. Experimental Setup

In order to validate and verify the platform, a specific federated environment is defined, consisting of a server and three clients with different capabilities. Furthermore, we have selected a public COVID-19 dataset from OWID [23] that contains information on the evolution of the pandemic with daily indicators by country on health statistics and demographic data and divided it into three different subsets (with 10%, 40%, and 50% samples each) as shown in Table 2.

Table 2. Summary of the capabilities of the client nodes and the size of the dataset used for functional validation.

Server	RAM Size	CPU	Graphic Card	Dataset Size (No. of Samples)
MacBook PRO	16 GB	2.20 GHz intel i7	No	12,441
Unix Server 1	48 GB	2.80 GHz intel i9	No	49,767
Unix Server 2	32 GB	3.50 GHz intel i7	NVIDIA TITAN RTX 24GB	62,208

The objective of the model to be applied within this environment is to try to predict new cases of COVID-19 by considering a set of specific variables from the samples in the dataset. To do so, a simple neural network (NN) has been defined, where the input for the model was composed of a vector that contained a subset of variables in the original dataset. These variables have been selected because they are present in almost all samples, thus easily allowing for a horizontal federation.

4.2. Functional Validation

To validate the platform deployment, two methods are tested: running a bash script on each node and using Jenkins. The first approach is easier and faster, but it requires the registration of client nodes on the server's database through the dashboard. On the contrary, the use of Jenkins is more complicated because it requires a previous setup of Jenkins on all nodes. However, once this is completed, the registration of clients is not necessary and the maintenance is easier. Additionally, Jenkins benefits from the advantages of continuous integration.

In general, both approaches have their own pros and cons, and the choice of which to use depends on the specific needs and requirements of the deployment. In some cases, the simplicity and speed of the bash script approach may be sufficient, while in other cases, the advanced features and capabilities of Jenkins may be necessary.

To thoroughly validate the model and the platform on which it is executed, we run the neural network model in both a local and a federated environment. This firstly allows us to assess the correct execution of the whole system and to compare the throughput and resource demands of both settings and determine the benefits and drawbacks of each approach.

For experiments using the NN model in a local environment, we chose to use a total of 20 epochs, which means that the dataset iterates 20 complete times during training. This number is selected based on the simplicity of the model and the fact that the prediction required is not complex. In the federated environment, we establish 20 global rounds and 1 local epoch for the NN model, which means that the algorithm iterates 20 times, but after each epoch, the aggregation strategy is performed to update the model's parameters. We choose the federated average, which consists of averaging the weights, as the model aggregation rule, as our intention is to focus on the performance of our platform without entering too much detail regarding distributed optimisation theory. Moreover, it is simple and achieves great performance if the federated datasets originate from the same distribution.

The dataset for each node is divided into 80% for model training and 20% for model validation. This allows us to evaluate the performance of the model on unseen data and to test its generalisation performance.

The results in Table 3 indicate that, in this specific case, federating NN improves performance compared to isolated training. Bear in mind that we do not want to state that federated learning improves isolated training, as it depends on the properties of federated datasets. A possible explanation is that all the datasets have the same distribution, with no attribute loss or noisy values, and thus the overall score is quite similar.

Table 3. Results of the verification experiments.

Server	Loss LR	Loss NN	Loss NN FL	Time Local vs. Federated	Max RAM Local vs. Federated
MacBook PRO	0.09145	0.08537	0.07672	+45 s	+21,636 kb
Unix Server 1	0.10235	0.09229	0.08824	+31 s	9992 kb
Unix Server 2	0.09332	0.083167	0.07666	+15 s	+8640 kb

Moreover, with respect to time, throughput becomes affected because the server must wait for the delivery of all partially fitted model replicas, including the slower ones, thus the overall score is lowered. Despite the longer time, these nodes are rewarded with better results.

It is also worth noting that memory usage is similar for all cases, so the use of federation does not significantly increase server resource usage. This, together with the time, allows a low impact on the performance so that algorithms, which can be resource-hungry, can still use most of the memory.

The three statements just mentioned can also be justified by looking at Figure 4, which shows the specifications of the dataset and the server resources on the one hand, and the performance comparison between FL and isolated on the other.

In general, these experiments and their results validate the functionality of the platform, as it is effective for the training of federated algorithms.

4.3. Objectives Verification

The main objective defined for this work is to create a platform that allows the use of FL in a simple way, so researchers could work with clinical data and machine learning algorithms without having to worry about the architecture and deployment of the system. To verify it, a specific study is defined consisting of the following steps.

- Step 1: setting the main objective of the study, in charge of defining the main objective of the experiment, that is, if it is possible to claim that this solution is easy to use.
- Step 2: setting the study context, in charge of defining the guidelines about how the study is conducted.
- Step 3: defining the main tools for the study, responsible for identifying the means to be applied during the study.
- Step 4: the implementation of the study and the collection of results focused on both, conducting the analysis according to the previous guidelines and the tools to be used and then collecting the information from the users.

- Step 5: analysis of the results and drawing of conclusions, in charge of discussing whether the main objective was achieved according to the results gathered from the study.

With the aim of not only checking if the platform is usable, but also comparing its level of usability with an existing solution, in Step 2 we carry out a comparative study. To do so, we first select the solution to be compared with, that is, Flower. We choose this one since it can be considered as the basis of our platform as previously explained and due to the fact that it also provides the main capabilities needed for a real federated learning implementation.

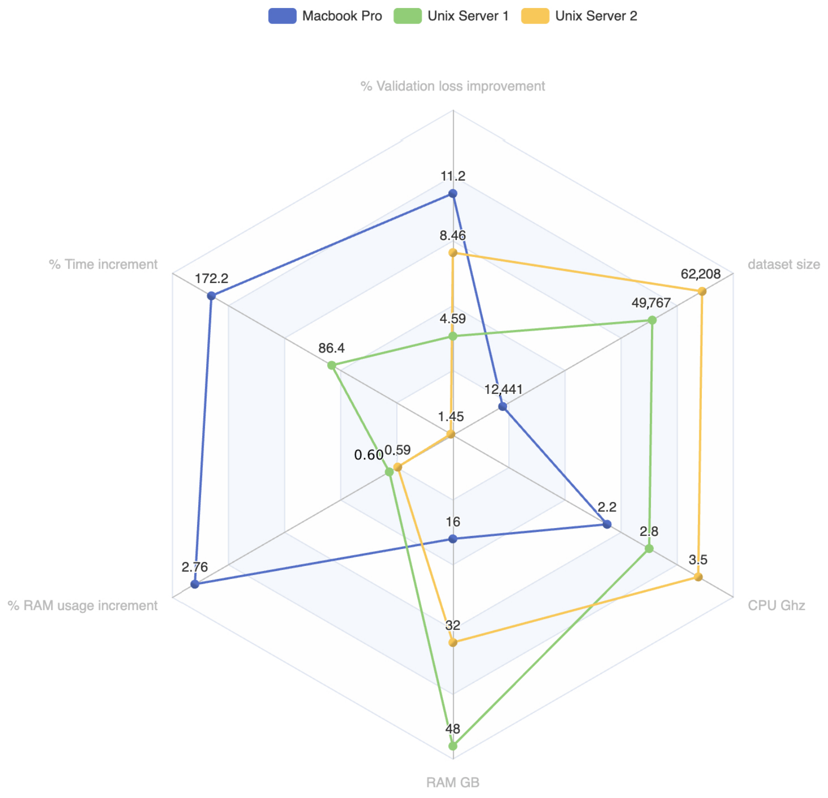

Figure 4. NN model performance vs. server capacities radar chart.

Regarding the tools to be used, for the analysis of user acceptance in terms of ease of use, we perform a SUS (System Usability Scale) test, since it can be considered the most widely adopted regulated questionnaire for the appraisal of perceived usability [24].

Regarding the usability test implementation, it is divided into two different parts: first, the users are asked to complete a set of tasks in the Flower scenario and then to answer a specific SUS to provide their opinion about their experience. Once this first step is completed, a second round is conducted following the same approach, focused on the use of our platform. For both rounds, users are provided with a list of instructions with the different steps that a machine learning expert should follow in a real scenario. These steps are included in Table 4.

Along with the standard SUS questions, an additional set of four initial questions is also included at the beginning of the test. They are related to the level of studies and the background knowledge about data science, federated learning, and computer system administration that helped us to understand the profiles of the involved participants. The test was finally performed by a total of 14 users, which is enough for a sample of usability tests according to the literature [25]. Figure 5 shows the results for these initial questions:

Table 4. Comparison of model training steps between Flower and FLIP.

Step	Action	Flower Alone	FLIP
1	Creation of server working environment	1. ssh connection in server node 2. Python virtual environment creation 3. Dependencies installation	Transparent to the user (Automated by the platform)
2	Creation of client working environment	1. ssh connection in client node 2. Python virtual environment creation 3. Dependencies installation	Transparent to the user (Automated by the platform)
3	Flower server deployment	1. Copy of server script to the server node via scp 2. Execution of the server script on the server node	1. Login on the platform
4	Flower clients deployment	1. Copy of client scripts to all client nodes via scp 2. Execution of the client script on all client nodes	1. Task creation on platform
5	Results gathering	1. Copy all the result files from server node via scp	1. Download all the result files from the platform

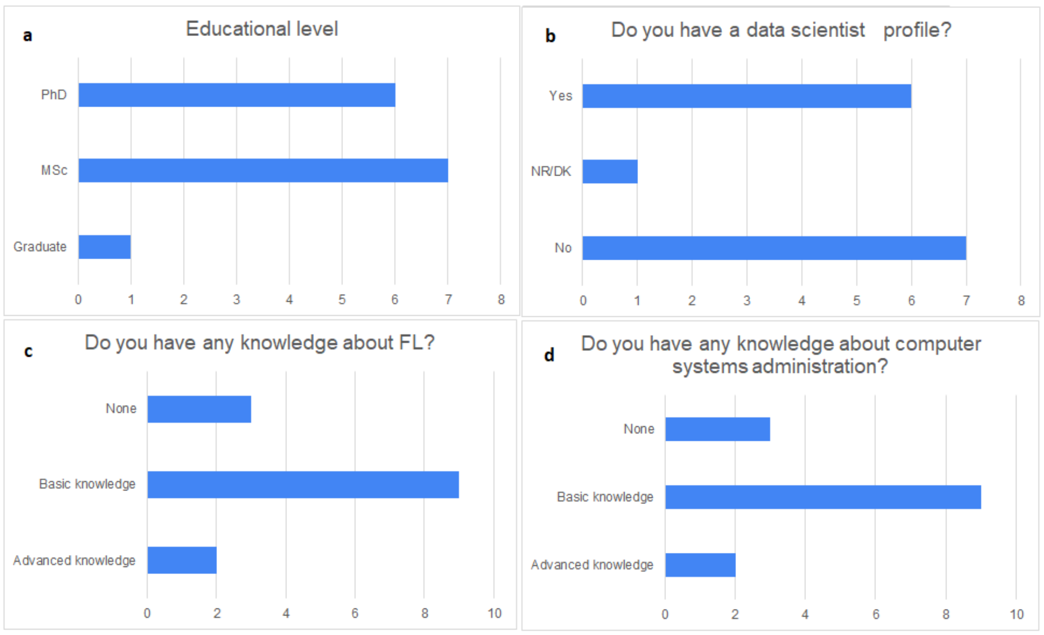

Figure 5. Results of the initial set of questions. (**a**) Educational level; (**b**) users with a data scientist profile; (**c**) knowledge about federated learning; and (**d**) knowledge about computer system administration.

According to these initial results, the participants have a high educational level (at least graduate), half of them have a data scientist profile, and a total of 11 have basic or advanced knowledge about federated learning and computer system administration.

Once provided with this initial information, users are asked to complete a SUS questionnaire for each of the solutions tested. For comparative analysis and presentation of results, we rely on the System Usability Scale Analysis Toolkit provided by [26]. To help interpret the results, it is important to first note the reference to each of the parameters considered. First, the SUS study score represents an average perceived usability score between 0 and 100, which is calculated based on the formula from [27], and considers 68 as the average rate. The percentile value [24] shows the SUS study scores on a cumulative percentile curve for the relation between the SUS score and the percentage of systems that exhibit a lower perceived usability level. The quartile scale, as the authors explain in [28], can be used to contextualise and compare the SUS scores with the scores in their dataset, which contains 3500 SUS scores divided into four quartiles. The acceptability scale contextualises the SUS study scores in descriptions ranging from "Not acceptable" to "Acceptable", based on data from [29]. The Net Promoter Score (NPS) [30] describes how likely users of a product are to recommend the analysed system to others. Finally, the industry benchmark scale is a non-empirical scale derived from [31] where the authors claim that scores above 80 are an "industrial goal". Figure 6 and Table 5 show the results obtained for both systems.

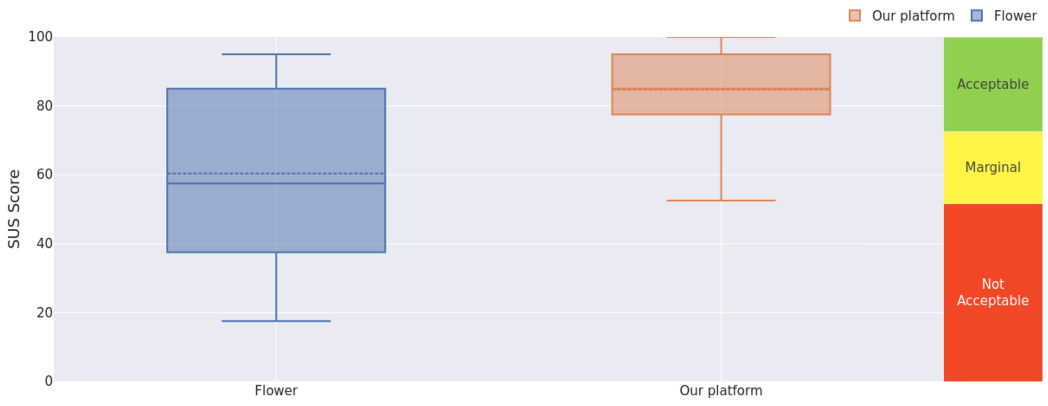

Figure 6. SUS scores for both systems, Flower and our platform.

Table 5. SUS questionnaire results comparison.

Parameter	Flower	Our Platform
SUS Score (mean)	60.36	84.64
SD	25.39	12.49
Min	17.5	52.5
Max	95	100
Percentile value	28.18	95.15
1. Quartile	36.25	76.87
Median	57.5	85
3. Quartile	85.62	95
Quartile scale	1st	4th
Acceptability Scale	Marginal	Acceptable
NPS Scale	Detractor	Promoter
Industry Benchmark	Below average	Above industry Standard

Taking into account the results obtained, we can claim that our platform presents a high level of perceived usability, as expected. The SUS score rates our solution at 84.64 (above the average of 68), which means that it has a perceived usability level higher than 95.15% of all products considered to establish the percentile scale. Moreover, with regard to the comparison with Flower, the SUS score of our platform is 40% higher and users' opinions are less dispersed (they offer a lower standard deviation). Users consider our platform as an acceptable system and, according to their responses, they can be considered as promoters of it, contrary to their opinion about Flower, which obtains only a marginal rate in the acceptability scale, and users may act as detractors of this option. Finally, the score obtained for the industry benchmark analysis suggests that our platform is above industry standards.

5. Conclusions and Future Work

Distributed, privacy-preserved machine learning is a hot topic at the moment as regulation is shifting towards private, strictly governed data environments (GDPR, PIPL, etc.), and the impact is even higher in the case of sensitive datasets, as in the clinical field. One of the ways to comply with these constraints is to leverage the use of federated learning systems, or federated systems more broadly, where the data reside in private and secured data storage centres, and the users never interact with it directly, but in an indirect way. For example, a researcher can conduct statistical analysis (aggregations) of the datasets without knowing the individualities, and thus violating the policies; another application consists of introducing algorithms which directly interact with data and learn patterns from it, and even consistently leveraging this feature in a multinode multidata setting to improve the dataset availability.

In this regard, there have been several attempts to build and introduce federated learning systems to the research community, keeping away the bridge between testing environments, where researchers have direct access to data, and production-ready systems, where researchers need to be allowed to specific data sources. We have designed and implemented FLIP, where privacy and security are taken into account as mandatory requirements of the software, while keeping the entry barrier for research as simple as possible. This way, we provide an easy to use web-based platform that can differentiate between two clear users, administrator and researcher, where the first is the unique person that can manage the system and may have direct contact with the data.

FLIP has demonstrated a low entry barrier, as shown in the SUS results, while keeping the performance almost in sync with respect to isolated training processes. We have proved that RAM usage keeps almost as low as in the traditional approach and that time is dependent on the hardware resources of the worst node. One way to potentially solve this issue could be to train the least powerful nodes once every several rounds, but this and other alternatives are kept for future studies.

FLIP is currently ongoing further changes within H2020 projects, and thus we have decided to keep it as a private repository, also to comply with IPR and project needs. We are planning to have a public release once we can show a high TRL (Technology Readiness Level).

Finally, within the scope of future work, even though the platform provides a functional tool, the set of production-ready services is just a few. We have aimed towards a platform covering the basic functionalities and performing correctly, rather than trying to achieve a complex, full-featured design, both because we have considered an incremental delivery process, and because decisions should happen carefully in order to ensure a steady step. In this way, we have established the basis for potential use cases that enhance the true potential of federated learning through an easy-to-use GUI. Furthermore, there are several possibilities to extend current state, which are, among others, the adoption of current standard tools to deliver new functionalities in a production-ready environment, the integration of ETL (Extract, Transform, and Load) and data storage services, and promoting the user experience to an even higher level.

Author Contributions: Conceptualization, B.A.G.; methodology, B.A.G., S.U.M., F.M.G. and S.B.L.; software, B.A.G., F.M.G. and S.B.L.; validation, B.A.G., S.U.M. and F.M.G.; formal analysis, B.A.G.; investigation, B.A.G. and S.U.M.; writing—original draft preparation, B.A.G., S.U.M., F.M.G. and S.B.L.; writing—review and editing, B.A.G. and S.U.M.; supervision, S.U.M.; project administration, S.U.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by European Union's Horizon 2020 COVID-X project under grant agreement no. 101016065.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the SUS questionnaire.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
CRUD	Create, Read, Update, and Delete
ETL	Extract, Transform, and Load
FL	Federated Learning
FLIP	Federated Learning Interactive Platform
GDPR	General Data Protection Regulation

GUI	Graphic User Interface
IoT	Internet of Things
ML	Machine Learning
NN	Neural Network
PIPL	Personal Information Protection Law
REST	Representational State Transfer
SUS	System Usability Scale
TRL	Technology Readiness Levels

References

1. Sasaki, Y. A Survey on IoT Big Data Analytic Systems: Current and Future. *IEEE Internet Things J.* **2022**, *9*, 1024–1036. [CrossRef]
2. Gourd, E. GDPR obstructs cancer research data sharing. *Lancet Oncol.* **2021**, *22*, 592. [CrossRef] [PubMed]
3. Devi, A.S.; Chinnasamy, A. Privacy Preservation of Sensitive Data in Big Data Analytics—A Survey. In Proceedings of the 2021 10th International Conference on Internet of Everything, Microwave Engineering, Communication and Networks (IEMECON), Jaipur, India, 1–2 December 2021; pp. 1–5. [CrossRef]
4. Liang, P.P.; Liu, T.; Ziyin, L.; Allen, N.B.; Auerbach, R.P.; Brent, D.; Salakhutdinov, R.; Morency, L.P. Think Locally, Act Globally: Federated Learning with Local and Global Representations. *arXiv* **2020**, arXiv:2001.01523.
5. Patel, V.A.; Bhattacharya, P.; Tanwar, S.; Gupta, R.; Sharma, G.; Bokoro, P.N.; Sharma, R. Adoption of Federated Learning for Healthcare Informatics: Emerging Applications and Future Directions. *IEEE Access* **2022**, *10*, 90792–90826. [CrossRef]
6. Pfohl, S.R.; Dai, A.M.; Heller, K. Federated and Differentially Private Learning for Electronic Health Records. *arXiv* **2019**, arXiv:1911.05861.
7. Chen, Y.; Wang, J.; Yu, C.; Gao, W.; Qin, X. FedHealth: A Federated Transfer Learning Framework for Wearable Healthcare. *arXiv* **2019**, arXiv:1907.09173.
8. Cosgriff, C.V.; Ebner, D.K.; Celi, L.A. Data sharing in the era of COVID-19. *Lancet Digit. Health* **2020**, *2*, e224. [CrossRef]
9. Zhuang, W.; Wen, Y.; Zhang, X.; Gan, X.; Yin, D.; Zhou, D.; Zhang, S.; Yi, S. Performance Optimization for Federated Person Re-identification via Benchmark Analysis. In Proceedings of the 28th ACM International Conference on Multimedia, Seattle, WA, USA, 12–16 October 2020. [CrossRef]
10. Zhou, J.; Zhang, S.; Lu, Q.; Dai, W.; Chen, M.; Liu, X.; Pirttikangas, S.; Shi, Y.; Zhang, W.; Herrera-Viedma, E. A Survey on Federated Learning and its Applications for Accelerating Industrial Internet of Things. *arXiv* **2021**, arXiv:2104.10501.
11. Muhammad, K.; Wang, Q.; O'Reilly-Morgan, D.; Tragos, E.; Smyth, B.; Hurley, N.; Geraci, J.; Lawlor, A. FedFast: Going Beyond Average for Faster Training of Federated Recommender Systems. In Proceedings of the P26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual Event, CA, USA, 6–10 August 2020; pp. 1234–1242. [CrossRef]
12. Zhuang, W.; Gan, X.; Wen, Y.; Zhang, S. EasyFL: A Low-code Federated Learning Platform for Dummies. *IEEE Internet Things J.* **2022**, *9*, 13740–13754. [CrossRef]
13. Aftab, U.; Siddiqui, G.F. Big Data Augmentation with Data Warehouse: A Survey. In Proceedings of the 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 10–13 December 2018; pp. 2785–2794. [CrossRef]
14. McMahan, H.B.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B.A. Communication-Efficient Learning of Deep Networks from Decentralized Data. *arXiv* **2017**, arXiv:1602.05629.
15. Caldas, S.; Duddu, S.M.K.; Wu, P.; Li, T.; Konečný, J.; McMahan, H.B.; Smith, V.; Talwalkar, A. LEAF: A Benchmark for Federated Settings. *arXiv* **2019**, arXiv:1812.01097.
16. The TensorFlow Federated Authors. TensorFlow Federated. 2018. Available online: https://www.tensorflow.org/federated (accessed on 7 March 2023).
17. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. TensorFlow: A system for large-scale machine learning. In Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation, OSDI'16, Savannah, GA, USA, 2–4 November 2016; pp. 265–283.
18. Chollet, F. Keras. 2015. Available online: https://keras.io/ (accessed on 7 March 2022).
19. Liu, Y.; Fan, T.; Chen, T.; Xu, Q.; Yang, Q. FATE: An Industrial Grade Platform for Collaborative Learning with Data Protection. *J. Mach. Learn. Res.* **2022**, *22*, 10320–10325.
20. Ma, Y.; Yu, D.; Wu, T.; Wang, H. PaddlePaddle: An Open-Source Deep Learning Platform from Industrial Practice. *Front. Data Comput.* **2019**, *1*, 105–115. [CrossRef]
21. Silva, S.; Altmann, A.; Gutman, B.; Lorenzi, M. Fed-BioMed: A General Open-Source Frontend Framework for Federated Learning in Healthcare. In *Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning*; Albarqouni, S., Bakas, S., Kamnitsas, K., Cardoso, M.J., Landman, B., Li, W., Milletari, F., Rieke, N., Roth, H., Xu, D., et al., Eds.; Lecture Notes in Computer Science Series; Springer International Publishing: Cham, Switzerland, 2020; Volume 12444, pp. 201–210. [CrossRef]
22. Beutel, D.J.; Topal, T.; Mathur, A.; Qiu, X.; Fernandez-Marques, J.; Gao, Y.; Sani, L.; Li, K.H.; Parcollet, T.; de Gusmão, P.P.B.; et al. Flower: A Friendly Federated Learning Research Framework. *arXiv* **2022**, arXiv:2007.14390.
23. Hasell, J.; Mathieu, E.; Beltekian, D.; Macdonald, B.; Giattino, C.; Ortiz-Ospina, E.; Roser, M.; Ritchie, H. A cross-country database of COVID-19 testing. *Sci. Data* **2020**, *7*, 345. [CrossRef] [PubMed]

24. Lewis, J.R. The System Usability Scale: Past, Present, and Future. *Int. J. Hum.-Comput. Interact.* **2018**, *34*, 577–590. [CrossRef]
25. Turner, C.W.; Lewis, J.R.; Nielsen, J. Determining usability test sample size. *Int. Encycl. Ergon. Hum. Factors* **2006**, *3*, 3084–3088.
26. Blattgerste, J.; Behrends, J.; Pfeiffer, T. A Web-Based Analysis Toolkit for the System Usability Scale. In Proceedings of the 15th International Conference on PErvasive Technologies Related to Assistive Environments, Corfu, Greece, 29 June–1 July 2022; pp. 237–246. [CrossRef]
27. Brooke, J. SUS: A quick and dirty usability scale. *Usability Eval. Ind.* **1995**, *189*, 4–7.
28. Bangor, A.; Kortum, P.; Miller, J. Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale. *J. Usability Stud.* **2009**, *4*, 114–123.
29. Bangor, A.; Kortum, P.T.; Miller, J.T. An Empirical Evaluation of the System Usability Scale. *Int. J. Hum.-Comput. Interact.* **2008**, *24*, 574–594. [CrossRef]
30. Reichheld, F.F. The one number you need to grow. *Harv. Bus. Rev.* **2003**, *81*, 46–54. 124. [PubMed]
31. Lewis, J.R.; Sauro, J. Item benchmarks for the system usability scale. *J. Usability Stud.* **2018**, *13*, 158–167.