Software Requirements Specification (SRS) of R25 -039

Project ID: R25 - 039

Project Title: Data-Privacy Focused Federated Learning Framework for Industrial IoT

Student Details:

Names:	Student IDs:
Nanayakkara Y.D.T. D	IT21826368
Mendis H.R.M	IT21822612
Weerasinghe K.M	IT21831904
Dissanayaka K.D.A.R. A	IT21828348

Supervisor: Mr. Amila Seneratha

External Supervisor: Dr. Sanika Wijesekara

Co-Supervisor: Mr. Tharaniyawarma Kumaralingam

Date of Submission: 1st May 2025

Table of Contents

1 Introduction

1.1 Purpose

The purpose of this document is to specify the comprehensive requirements for the **Data-privacy based Federated Learning Framework for IIoT**. It defines the system's architecture, functionalities, components, and constraints to serve as the foundational guide for the project's development, testing, and future maintenance.

1.2 Scope

This SRS covers the design and requirements for a federated learning framework tailored for Industrial IoT (IIoT) environments. The scope encompasses:

The overall **client-server architecture** designed for collaborative model training.

The **Secure Communication and Protocol Enforcement Module (SCPM)**, responsible for establishing secure and authenticated channels.

The **Attack Detection and Resilience Module (ADRM)**, which identifies and mitigates cyber threats like model poisoning and Byzantine attacks.

The **Privacy Preservation Module (PPM)**, which employs cryptographic techniques to ensure client data privacy.

The **Secure Aggregation Module (SAM)**, which securely combines model updates from clients without revealing individual contributions to the server.

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
ADRM	Attack Detection and Resilience Module
DP	Differential Privacy
FL	Federated Learning
HE	Homomorphic Encryption
HMAC	Hash-based Message Authentication Codes
IIoT	Industrial Internet of Things
PPM	Privacy Preservation Module
SAM	Secure Aggregation Module
SCPM	Secure Communication and Protocol Enforcement Module
TLS	Transport Layer Security

1.4 Overview

The software is a specialized framework designed to implement

Federated Learning (FL) in Industrial IoT (IIoT) environments while prioritizing data privacy and cybersecurity. The main goal is to allow distributed IIoT devices to collaboratively train a shared machine learning model without centralizing their raw data, thus mitigating privacy risks and single points of failure.

Key tasks include establishing secure communication channels, detecting and defending against adversarial attacks, preserving confidentiality with advanced cryptographic techniques, and securely aggregating model updates to improve a global model. The primary users of the system are autonomous **IoT devices (clients)** and the **central server** that orchestrates the entire learning process.

2 Overall Descriptions

The proliferation of Industrial Internet of Things (IIoT) devices has created a challenge where machine learning models require data that is inherently distributed. Traditional centralized approaches introduce significant privacy risks and create single points of failure, which conflict with the nature of industrial operations and data regulations. While Federated Learning (FL) presents a solution by training models locally, existing FL systems are often insufficient for the IIoT context. They typically fail to provide a single, comprehensive framework that simultaneously addresses the critical needs for security, privacy, efficient communication, and the resource limitations of IIoT devices.

This product aims to fill that gap by developing a robust, modular, and integrated FL framework specifically designed to meet these unique industrial demands. It provides an end-to-end solution for collaborative and secure model training by combining a multi-layered security architecture with a robust secure aggregation protocol

2.1 Product perspective

This framework improves upon existing approaches in the field. Compared to other solutions:

- **Foundational FL Frameworks**: Early work focused primarily on communication efficiency but lacked robust mechanisms to handle the unique security threats and resource constraints of industrial environments.

Privacy-Enhancing FL: Other researchers have integrated cryptographic techniques like secure aggregation, Differential Privacy (DP), and Homomorphic Encryption (HE). However, these solutions often introduce significant computational overhead, making them challenging for resource-limited IIoT devices.

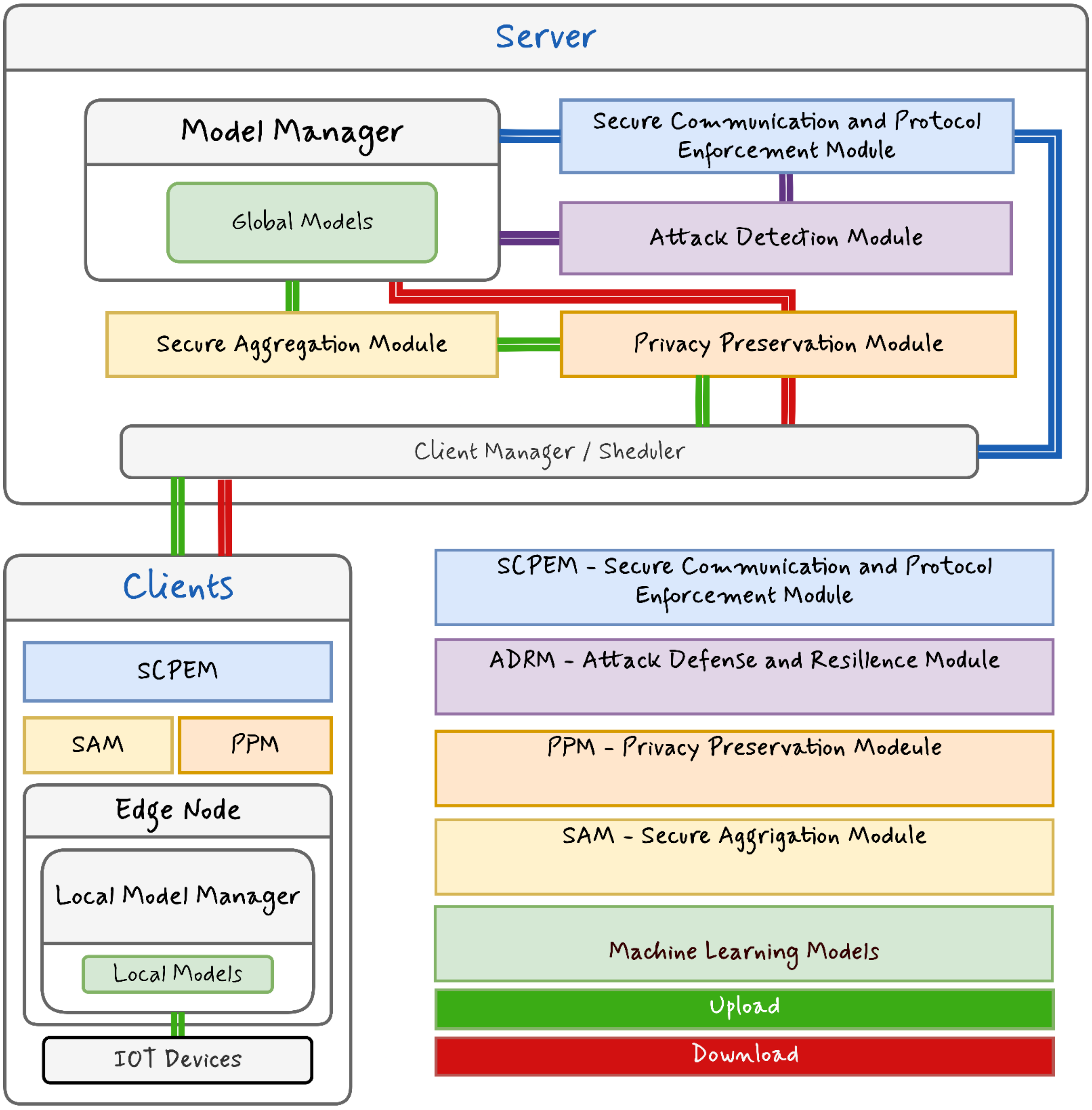
Attack-Resilient FL: Studies have explored defenses against poisoning and Byzantine attacks, but these methods typically do not provide a holistic solution that also accounts for privacy protection and communication efficiency.

To the authors' knowledge, no existing framework integrates robust solutions for all these critical challenges—

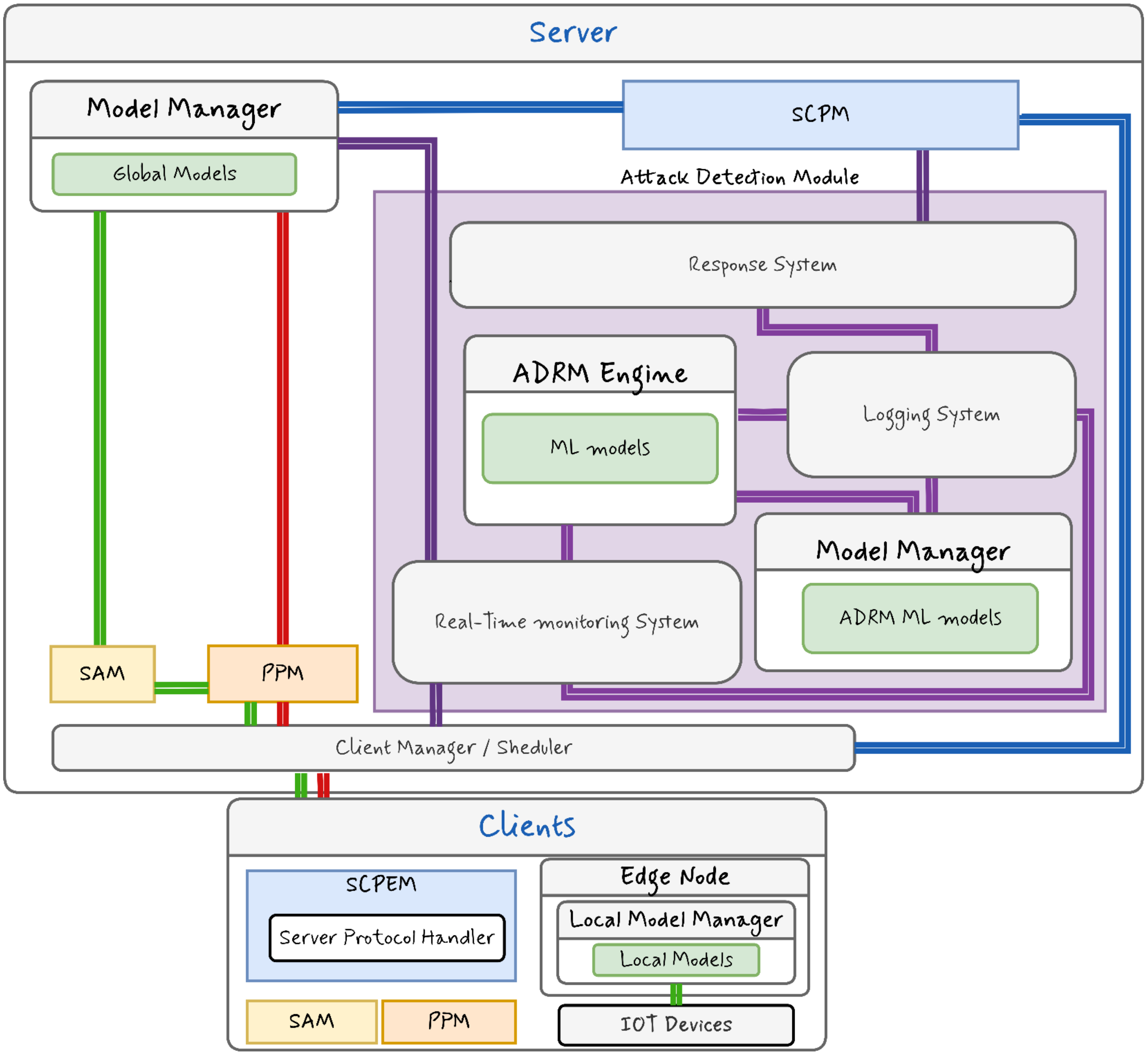
security, privacy, and resource efficiency—in a single, unified system tailored for the IIoT, which is the primary contribution of this product.

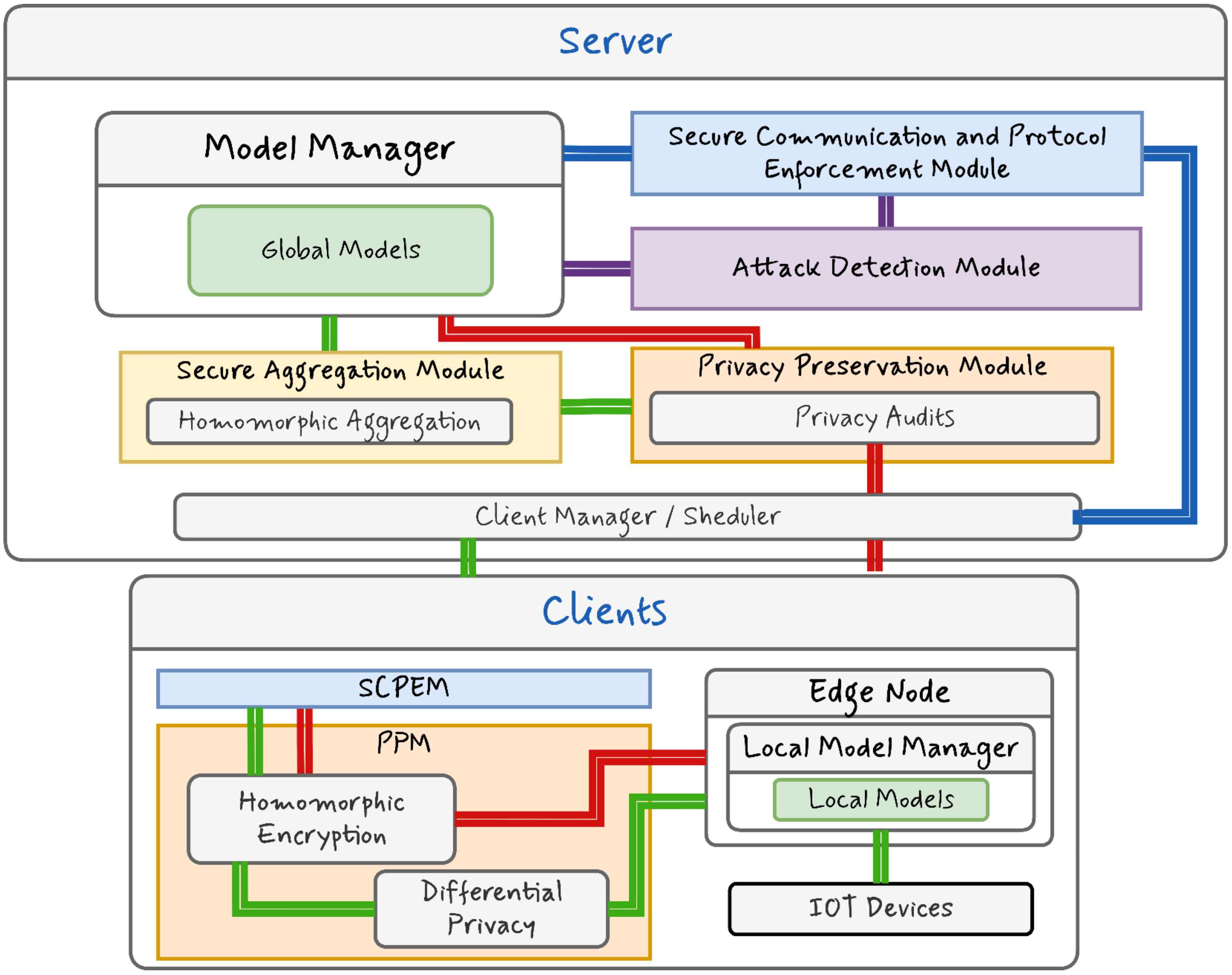
2.1.1 System interfaces

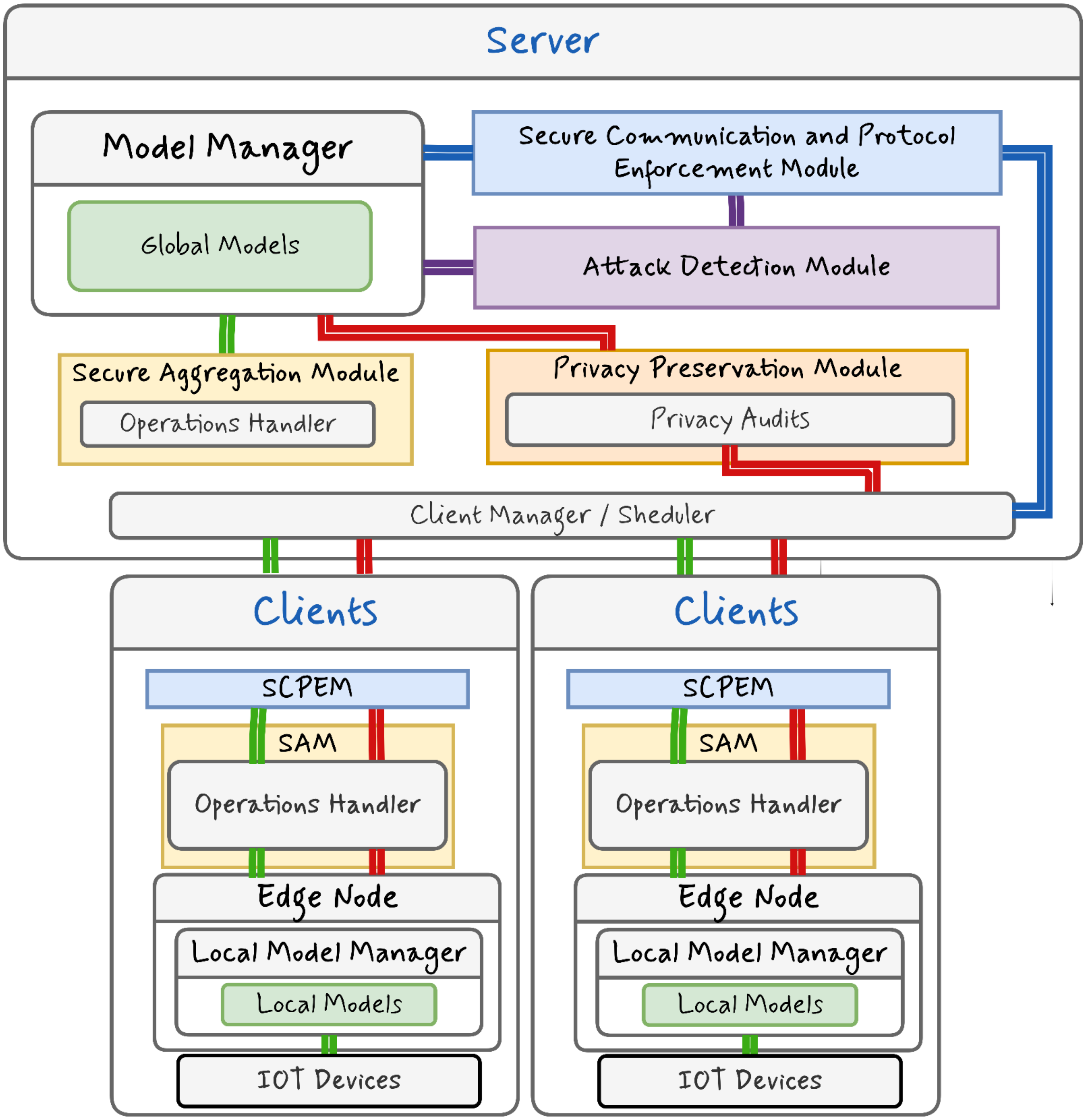
Main System architecture

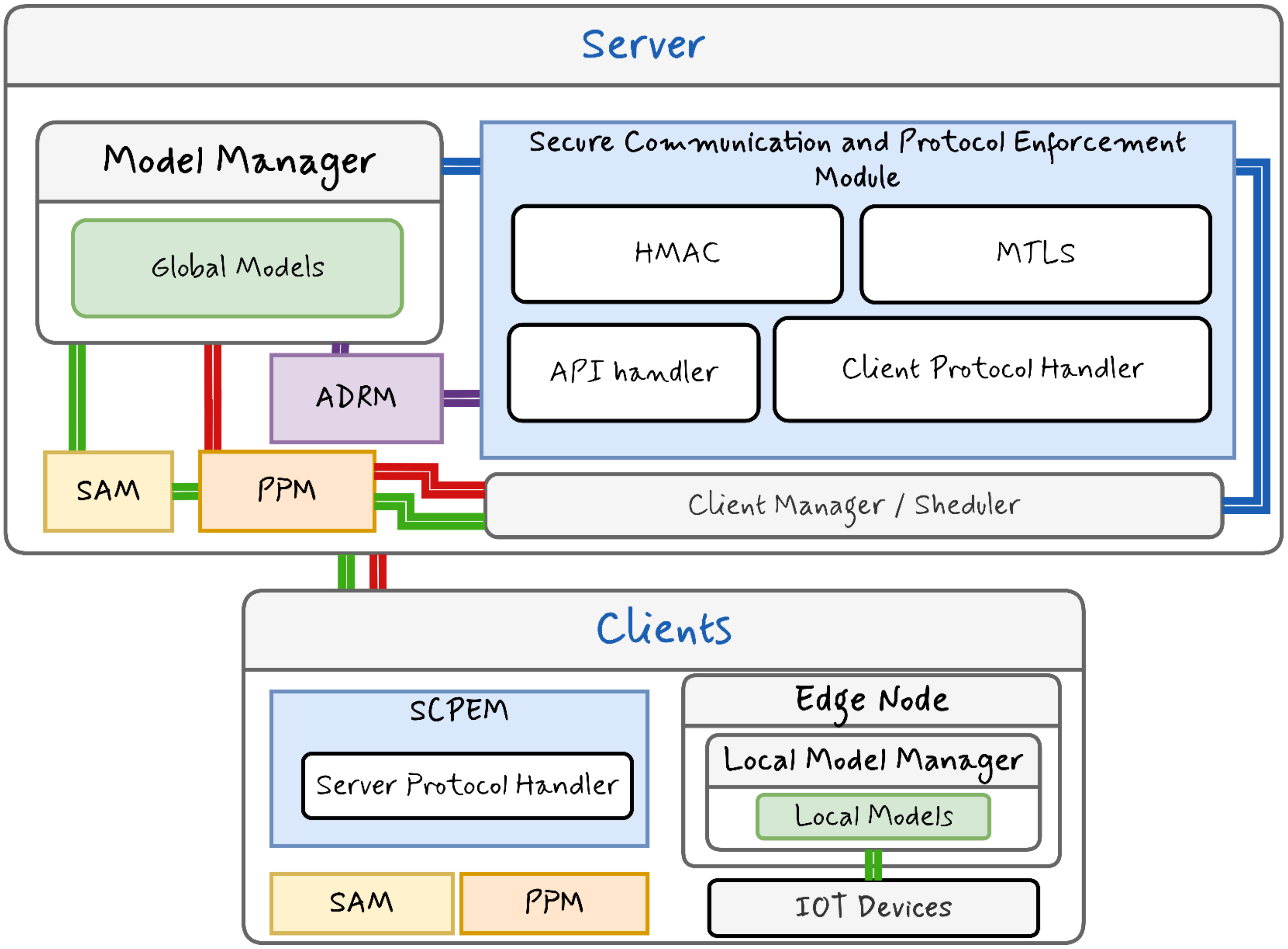

Server
Model Manager
Global Models
Secure Communication and Protocol Enforcement Module
Attack Detection Module
Secure Aggregation Module
Privacy Preservation Module
Client Manager / Shedduler
Clients
SCPEM
SAM
PPM
Edge Node
Local Model Manager
Local Models
IOT Devices
SCPEM - Secure Communication and Protocol Enforcement Module
ADRM - Attack Defense and Resillence Module
PPM - Privacy Preservation Modeule
SAM - Secure Aggrigation Module
Machine Learning Models
Upload
Download

ADRM – Attack Defense and Resilience Module




Server
Model Manager
Global Models
Secure Communication and Protocol Enforcement Module
Attack Detection Module
Secure Aggregation Module
Homomorphic Aggregation
Privacy Preservation Module
Privacy Audits
Client Manager / Sheduler
Clients
SCPEM
PPM
Homomorphic Encryption
Differential Privacy
Edge Node
Local Model Manager
Local Models
IOT Devices


Server
Model Manager
Global Models
Secure Communication and Protocol Enforcement Module
Attack Detection Module
Secure Aggregation Module
Operations Handler
Privacy Preservation Module
Privacy Audits
Client Manager / Sheduler
Clients
SCPEM
SAM
Operations Handler
Edge Node
Local Model Manager
Local Models
IOT Devices
Clients
SCPEM
SAM
Operations Handler
Edge Node
Local Model Manager
Local Models
IOT Devices


Server
Model Manager
Global Models
Secure Communication and Protocol Enforcement Module
HMAC
MTLS
API handler
Client Protocol Handler
ADRM
SAM
PPM
Client Manager / Sheduler
Clients
SCPEM
Server Protocol Handler
SAM
PPM
Edge Node
Local Model Manager
Local Models
IOT Devices

2.1.2 User interfaces

Still being developed

The User interface will be developed on the API calles

2.1.3 Hardware interfaces

The framework is designed to operate on a client-server hardware model:

- **Clients**: The clients are **Industrial IoT (IIoT) devices**, which are often resource-constrained in terms of computation and energy.

Server: A central **server** is required to manage the overall federated learning process, including model aggregation and client coordination.

2.1.4 Software interfaces

The framework operates as a self-contained client-server system. The primary software interface is the communication protocol defined between the server and the IIoT clients. While clients will run local model training, which implies the use of a machine learning library, the specific library is not named in the document.

2.1.5 Communication interfaces

The framework's SCPM module establishes and maintains a secure and trustworthy channel for all data exchanges.

Encryption: It utilizes **Transport Layer Security (TLS) 1.3** to secure all network traffic against eavesdropping and man-in-the-middle attacks.

Authentication: It enforces **mutual authentication**, where both the server and the client must present and validate a digital certificate, ensuring only authorized devices participate.

Data Integrity: It uses **Hash-based Message Authentication Codes (HMAC)** to generate a unique digital signature for each message, ensuring data has not been altered in transit.

2.1.6 Memory constraints

The software is designed for

resource-limited IIoT devices. The document acknowledges that privacy-preserving techniques, especially homomorphic encryption, have significant

computational overhead that can be a limitation. The system design therefore aims to optimize communication and computation to reduce overhead and energy use, implying it must operate within tight memory and processing constraints.

2.1.7 Operations

- **Normal Operations**:
  - The system performs collaborative model training where clients compute updates on their local data and send only those updates, not the raw data, to the server.
  - The server aggregates these updates, creates an improved global model, and distributes it back to clients for the next training round.

- **Special Operations**:
  - **Attack Detection & Mitigation**: The ADRM actively monitors for attacks (e.g., poisoning, Byzantine) and takes measures to mitigate them, such as isolating malicious clients or discarding their updates.
  - **Client Management**: The server manages client participation, including selection and scheduling for training rounds.
  - **Privacy Audits**: The system can conduct audits to verify that the privacy mechanisms are functioning correctly and to check for information leakage.

2.1.8 Site adaptation requirements

Requirements for execution on a particular installation

Server side, A machine contains with Python libaries and a network connection

Client Side, A machine contains with Python libaries and a network conncetion

2.2 Product functions

The framework's functionality is delivered through several interconnected modules:

- **Secure Communication (SCPM)**: Establishes a confidential and integrity-protected channel between clients and the server using TLS 1.3 with mutual authentication and HMAC for message validation.
- **Attack Detection (ADRM)**: Employs anomaly detection and statistical analysis to identify and neutralize malicious client behavior, thereby protecting the integrity of the global model.
- **Privacy Preservation (PPM)**: Implements a dual-layer of privacy using Differential Privacy (DP) to add statistical noise and Homomorphic Encryption (HE) to allow computation on encrypted data, preventing data inference.
- **Secure Aggregation (SAM)**: Uses cryptographic protocols like Shamir's Secret Sharing to aggregate client model updates, ensuring the server cannot inspect any individual contribution.
- **Centralized Orchestration**: The server manages the entire FL lifecycle, including client selection, model distribution, update aggregation, and versioning of the global model.

2.3 User characteristics

The primary "users" of the system are not humans but autonomous system components:

- **Clients**: These are **Industrial IoT (IIoT) devices** that are expected to be resource-constrained and operate at the network edge. They perform local computations and communicate with the server.
- **Server**: A central server that orchestrates the process. This component would be managed by technical personnel (e.g., system administrators, MLOps engineers), though their specific characteristics are not detailed in the paper.

2.4 Constraints

- **Computational Overhead**: The use of advanced cryptography, particularly homomorphic encryption, imposes a significant computational burden that is a key limitation for resource-constrained devices.
- **Resource Limitations**: The framework must be optimized for low energy consumption and efficient computation to be viable on typical IIoT hardware.
- **Network Reliability**: The system must be robust against client dropouts, which are a common issue in distributed IIoT networks.

2.5 Assumptions and dependencies

Sufficient Participants: The secure aggregation process assumes that a minimum number of clients (a threshold) will successfully submit their shares in each round to allow for the reconstruction of the global model update.

Trusted Key Administrators: For decrypting the final model when using Homomorphic Encryption, the framework relies on a master decryption key that is split into shares and distributed among a set of trusted administrators. It is assumed that enough of these administrators will cooperate for decryption and that they will not collude to compromise the key.

Secure Credential Management: The mutual authentication process depends on the assumption that both the server and all legitimate clients have been securely provisioned with valid digital certificates.

2.6 Apportioning of requirements

Order in which requirements are to be implemented.

1. *Setting up the server*
2. *Installing the python libraries*
3. *Setting up Network connection*
4. *Running the server.py*
5. *Running the Frontends, either the Terminal user interface or the Web portal*
6. *Setting up Client side*
7. *Installing the Python liberates*
8. *Running the Client machines which configured the ports and connect to the same network or the network given*
9. *Wait till the registration and running the successful, Select the Version to run either HE or SSE*
10. *Then the server starts the rounds when the server requirements (connected client total) are connected, until the rounds are over it will run again and and again*

Outcome – Trained ML model using Federated Learning

3 Specific requirements(1) (for Software Dev. Oriented Projects – SRS)

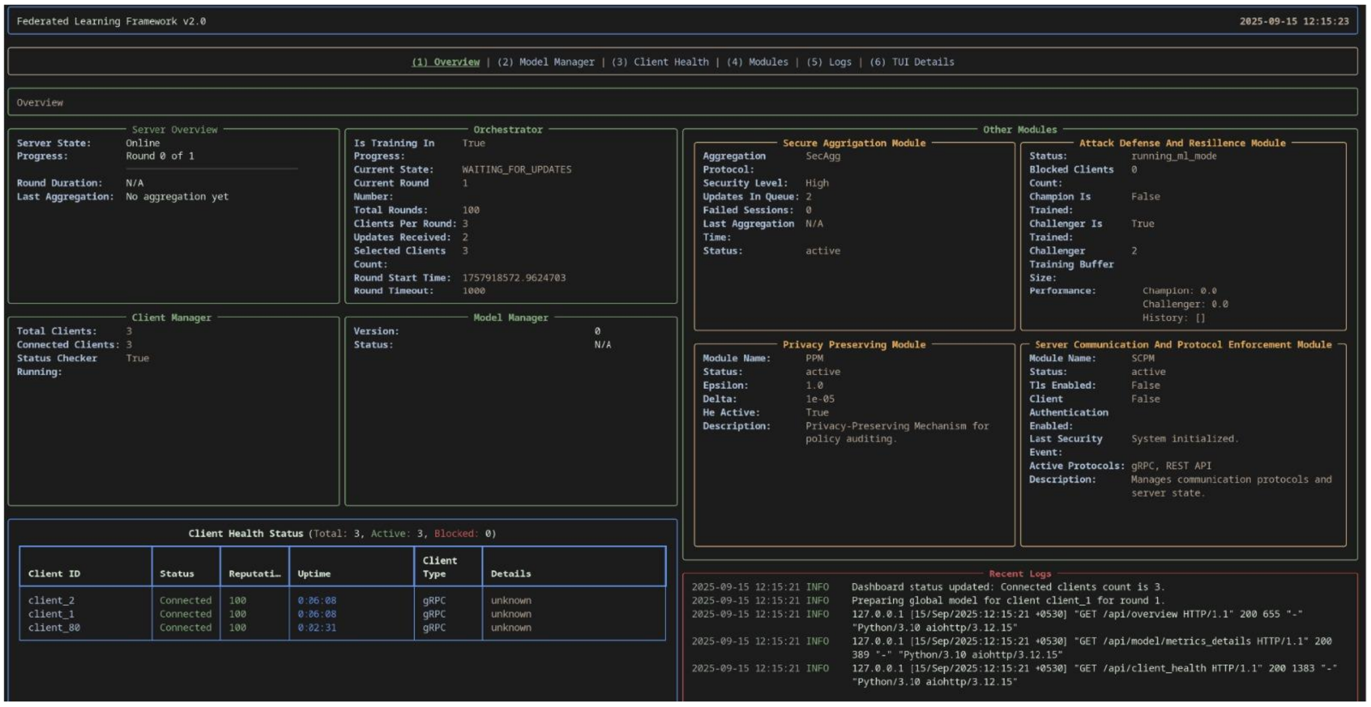
3.1 External interface requirements

No external interface requirements

3.1.1 Software interfaces.

For now, we have a Terminal User interface Which shows all the details

Proposed System – Frontend Terminal User Interface(TUI)



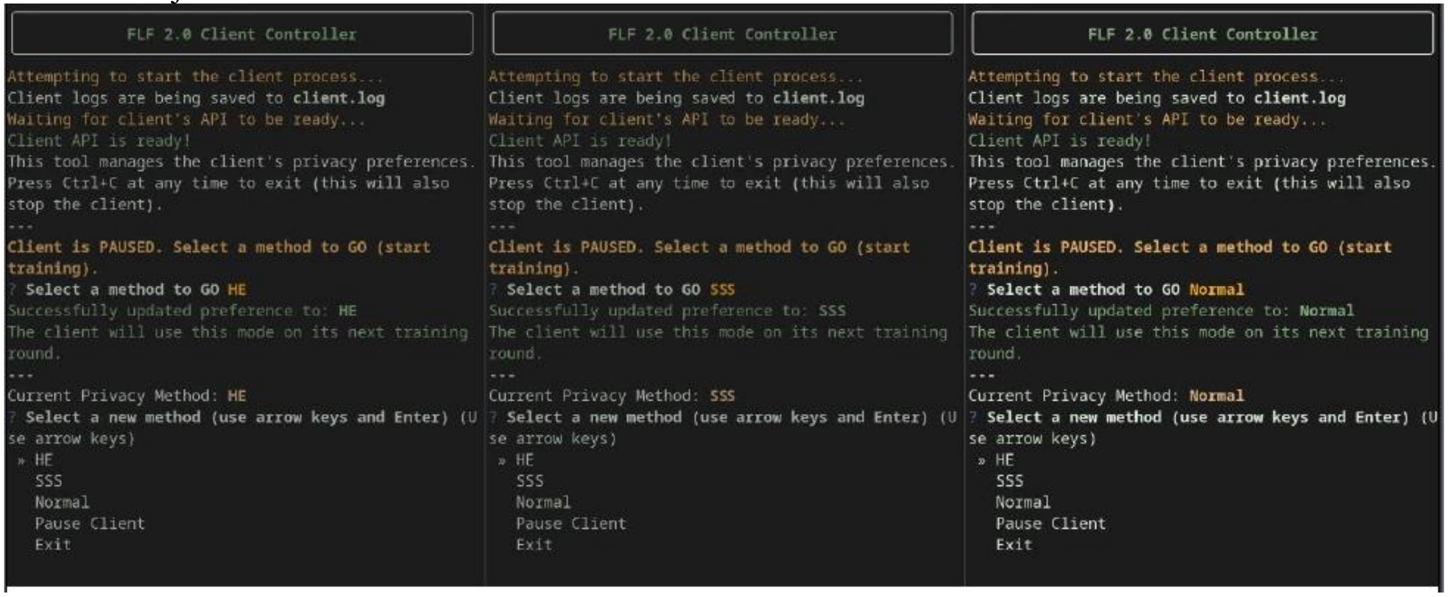
```
Federated Learning Framework v2.0                                                    2025-09-15 12:15:23

         (1) Overview | (2) Model Manager | (3) Client Health | (4) Modules | (5) Logs | (6) TUI Details

Overview

─ Server Overview ─
Server State:       Online
Progress:           Round 0 of 1

Round Duration:     N/A
Last Aggregation:   No aggregation yet

─ Orchestrator ─
Is Training In      True
Progress:
Current State:      WAITING_FOR_UPDATES
Current Round       1
Number:
Total Rounds:       100
Clients Per Round:  3
Updates Received:   2
Selected Clients    3
Count:
Round Start Time:   1757918572.9624703
Round Timeout:      1000

─ Client Manager ─
Total Clients:      3
Connected Clients:  3
Status Checker      True
Running:

─ Model Manager ─
Version:                                     0
Status:                                      N/A

─ Other Modules ─
─ Secure Aggrigation Module ─
Aggregation         SecAgg
Protocol:
Security Level:     High
Updates In Queue:   2
Failed Sessions:    0
Last Aggregation    N/A
Time:
Status:             active

─ Attack Defense And Resillence Module ─
Status:             running_ml_mode
Blocked Clients     0
Count:
Champion Is         False
Trained:
Challenger Is       True
Trained:
Challenger          2
Training Buffer
Size:
Performance:          Champion: 0.0
                      Challenger: 0.0
                      History: []

─ Privacy Preserving Module ─
Module Name:        PPM
Status:             active
Epsilon:            1.0
Delta:              1e-05
He Active:          True
Description:        Privacy-Preserving Mechanism for
                    policy auditing.

─ Server Communication And Protocol Enforcement Module ─
Module Name:        SCPM
Status:             active
Tls Enabled:        False
Client              False
Authentication
Enabled:
Last Security       System initialized.
Event:
Active Protocols:   gRPC, REST API
Description:        Manages communication protocols and
                    server state.

Client Health Status (Total: 3, Active: 3, Blocked: 0)
┌──────────────┬───────────┬───────────┬─────────┬─────────┬──────────┐
│              │           │           │         │ Client  │          │
│ Client ID    │ Status    │ Reputati… │ Uptime  │ Type    │ Details  │
├──────────────┼───────────┼───────────┼─────────┼─────────┼──────────┤
│ client_2     │ Connected │ 100       │ 0:06:08 │ gRPC    │ unknown  │
│ client_1     │ Connected │ 100       │ 0:06:08 │ gRPC    │ unknown  │
│ client_80    │ Connected │ 100       │ 0:02:31 │ gRPC    │ unknown  │
└──────────────┴───────────┴───────────┴─────────┴─────────┴──────────┘

─ Recent Logs ─
2025-09-15 12:15:21 INFO    Dashboard status updated: Connected clients count is 3.
2025-09-15 12:15:21 INFO    Preparing global model for client client_1 for round 1.
2025-09-15 12:15:21 INFO    127.0.0.1 [15/Sep/2025:12:15:21 +0530] "GET /api/overview HTTP/1.1" 200 655 "-"
                            "Python/3.10 aiohttp/3.12.15"
2025-09-15 12:15:21 INFO    127.0.0.1 [15/Sep/2025:12:15:21 +0530] "GET /api/model/metrics_details HTTP/1.1" 200
                            389 "-" "Python/3.10 aiohttp/3.12.15"
2025-09-15 12:15:21 INFO    127.0.0.1 [15/Sep/2025:12:15:21 +0530] "GET /api/client_health HTTP/1.1" 200 1383 "-"
                            "Python/3.10 aiohttp/3.12.15"
```

3.1.2 Hardware interfaces

No hardware interfaces all software based

3.1.3 User software interfaces

3 Clients running separate terminals split



```
FLF 2.0 Client Controller

Attempting to start the client process...
Client logs are being saved to client.log
Waiting for client's API to be ready...
Client API is ready!
This tool manages the client's privacy preferences.
Press Ctrl+C at any time to exit (this will also
stop the client).
---
Client is PAUSED. Select a method to GO (start
training).
? Select a method to GO HE
Successfully updated preference to: HE
The client will use this mode on its next training
round.
---
Current Privacy Method: HE
? Select a new method (use arrow keys and Enter) (U
se arrow keys)
 » HE
   SSS
   Normal
   Pause Client
   Exit
```

```
FLF 2.0 Client Controller

Attempting to start the client process...
Client logs are being saved to client.log
Waiting for client's API to be ready...
Client API is ready!
This tool manages the client's privacy preferences.
Press Ctrl+C at any time to exit (this will also
stop the client).
---
Client is PAUSED. Select a method to GO (start
training).
? Select a method to GO SSS
Successfully updated preference to: SSS
The client will use this mode on its next training
round.
---
Current Privacy Method: SSS
? Select a new method (use arrow keys and Enter) (U
se arrow keys)
 » HE
   SSS
   Normal
   Pause Client
   Exit
```

```
FLF 2.0 Client Controller

Attempting to start the client process...
Client logs are being saved to client.log
Waiting for client's API to be ready...
Client API is ready!
This tool manages the client's privacy preferences.
Press Ctrl+C at any time to exit (this will also
stop the client).
---
Client is PAUSED. Select a method to GO (start
training).
? Select a method to GO Normal
Successfully updated preference to: Normal
The client will use this mode on its next training
round.
---
Current Privacy Method: Normal
? Select a new method (use arrow keys and Enter) (U
se arrow keys)
 » HE
   SSS
   Normal
   Pause Client
   Exit
```

3.1.4 Communication interfaces

Socket programming, Demonstration will be done in a local network with a normal router.

3.2 Classes/Objects < For Software Dev. Oriented Projects>

User python as the programming language

These final requirements classification is based on all the provided Python modules, which represent the distinct functional components of the Federated Learning Server (FLS) application.

These classes are designed to categorize all requirements effectively, from the core training loop to advanced security and management features.

Final Requirements Classification Classes

Class Name	**Modules/Components Represented**	**Description**	**Priority**
Federated Learning Orchestrator	orchestrator.py	Manages the **full training lifecycle**: round transitions, state control, client selection, and dispatching tasks to other components. It is the central coordination unit.	**Essential**
Client Registry & Manager	client_manager.py	Manages the **client-side state and metadata**: connection status, registration, selection logic,	**Essential**

		reputation tracking, and persistent storage of client information.	
Global Model Manager	model_manager.py, sam.py	Manages the **central machine learning model**: initialization, storage of global weights, evaluation on the test set, and integration with the Secure Aggregation Mechanism (SAM).	**Essential**
Server Control Plane Manager (SCPM)	scpm.py, server.py	Manages all **external communications and APIs**: gRPC handlers for client communication, REST API for dashboard/admin interfaces, and life-cycle of communication listeners.	**Essential**
Adaptive Decentralized Response Mechanism (ADRM)	adrm_engine.py, anomaly_model.py, adrm/config.py	The sophisticated **multi-stage defense system**: detects malicious/anomalous client updates (using ML-based and statistical checks) and triggers administrative responses like blocking clients or applying penalties.	**Desirable**
Secure Aggregation Mechanism (SAM)	sam.py	Implements advanced, **trustworthy aggregation methods** like FedAdam and homomorphic encryption aggregation to ensure robust and privacy-preserving model updates.	**Desirable**
Privacy Policy Mechanism (PPM)	ppm.py	Audits and validates the **privacy protocol compliance** of received client updates against the active policies (e.g., Homomorphic Encryption (HE) policy).	**Desirable**
System Configuration & Logging	config.py, log_manager modules (referenced throughout)	Manages all **non-domain-specific parameters**: network settings, hyper-parameters, file paths, and centralized logging mechanisms for system health and audit trails.	**Optional**
Core Data & Model Definition	common_model.py, data_loader.py	Defines the **base data structure and ML architecture**: Specifies the neural network (e.g., SimpleFCN) and the data loading/preprocessing logic for the FL task.	**Optional**

Priority Definitions

- **Essential:** A mandatory requirement for the core Federated Learning loop to function. The system cannot operate as an FLS without it.
- **Desirable:** Requirements that implement advanced functionality, security, or robustness, adding significant value and fulfilling key non-functional requirements.
- **Optional:** Features or components that, while useful, could be omitted, postponed, or are implementation-specific details that do not impact the minimal viable product's core functionality.

3.3 Performance requirements

100 rounds for the demonstration, it can be configured as the ML architecture and the Dataset

Runtime also can be changed for now it is 100000ms

For one client, according to the training ML a CPU or GPU and the memory requirements can be changed

We have tested the Whole setup in one laptop (server and clients) in the CNN ML model and the CIFAR-10 dataset, with 3 clients.

Then going to move on to a bigger dataset and the ML architecture which will be scaled the system with more than one laptop (one for server and others for clients) used WUSTL IIOT dataset

Based on the provided Python modules and the nature of a modern Federated Learning Server (FLS), here are the design constraints and required software system attributes for the SRS.

3.4 Design Constraints

Constraint	**Description**	**Source/Reason**
Asynchronous Architecture	The server MUST be designed using an **asynchronous (asyncio)** programming model to handle thousands of concurrent client connections without thread blocking, as evidenced by the use of asyncio and await in server.py, orchestrator.py, and client_manager.py.	**Performance, Scalability** (Required by high-throughput network operations).
Communication Protocol	All external client-server communication MUST be implemented via a robust, high-performance protocol, specifically **gRPC** (based on scpm.py imports), to support efficient, bi-directional, and defined data transfer (model weights, updates, commands).	**Interoperability, Efficiency** (Protocol is defined by the scpm module's purpose).
Framework Dependence	The core model management and aggregation logic MUST rely on the **PyTorch** deep learning framework, as shown by imports (torch, torch.nn) in common_model.py, model_manager.py, and sam.py.	**Implementation Dependency** (The choice of ML framework is fixed).
Persistent Storage	Critical client state data (e.g., reputation, history) and ADRM model data (Champion/Challenger) MUST be managed using **file-based persistence** (e.g., JSON, PKL) and protected by synchronization primitives (asyncio.Lock in client_manager.py) to ensure data integrity across asynchronous operations.	**Data Integrity, Persistence** (Evident from file I/O and locking in client_manager.py and ADRM modules).

Security Mechanism Integration	The final model aggregation step MUST integrate and support at least two security/privacy mechanisms: **Secure Aggregation Mechanism (SAM)** (e.g., FedAdam) and an **Adaptive Decentralized Response Mechanism (ADRM)** for anomaly detection.	**Functionality Requirement** (Directly implemented by orchestrator.py and model_manager.py imports).

3.5 Software System Attributes

3.5.1 Reliability

The system must operate continuously and correctly, particularly regarding the core federated learning loop.

- **Failure Isolation:** Errors occurring within a single client connection or a non-critical module (e.g., logging) must **not** crash the central Orchestrator or disrupt the main training loop. Exception handling must be robust (as seen with try...except asyncio.CancelledError in server.py).
- **Checkpointing:** The Global Model and key Orchestrator state must be checkpointed after every successful aggregation to allow for fast recovery from a total system failure with minimal loss of training progress.
- **Data Integrity:** All state-modifying operations on the Client Registry must use **locks** (async with self._lock) to prevent race conditions and ensure data consistency in the asynchronous environment.

3.5.2 Availability

The server must be highly available to all registered clients to minimize downtime and maximize participation.

- **99.9% Uptime Goal:** The system should target 99.9% uptime (less than 9 hours of downtime per year) for core operational features (model dispatch and update reception).
- **Fast Restart:** The server must be able to load persistent data (ClientInfo, ADRM models, GlobalModel state) and restart the Orchestrator loop within **60 seconds** to minimize disruption.
- **Live Status:** The **Server Control Plane Manager (SCPM)** must provide a real-time status API (evidenced by scpm.py and its API handler reference) for immediate diagnostics and health checks.

3.5.3 Security

Security is paramount due to the decentralized nature of Federated Learning.

- **Authentication & Authorization:** The communication protocol (gRPC) must implement a secure channel (e.g., **TLS/SSL**) to authenticate clients and encrypt all data-in-transit (model updates, global model).
- **Malicious Client Mitigation:** The system **MUST** actively detect and respond to malicious updates using the **ADRM Engine** (adrm_engine.py), which involves:
  - ML-based anomaly detection.
  - Dynamic blocking/penalty mechanisms via the ResponseSystem in client_manager.py.
- **Privacy Auditing:** The system must validate the privacy claims of incoming updates using the **Privacy Policy Mechanism (PPM)** (ppm.py) to ensure they comply with the configured policies (e.g., Homomorphic Encryption).

3.5.4 Maintainability

The codebase must be structured to facilitate long-term maintenance, feature addition, and bug fixes.

- **Modular Design:** The system is strictly divided into distinct, loosely coupled modules (e.g., Orchestrator, ClientManager, ModelManager, ADRM, SCPM), which must be preserved to isolate concerns and allow parallel development.
- **Logging:** The system must utilize structured, context-aware logging (as seen in the use of ContextAdapter in multiple modules) to allow maintainers to filter and trace system events by component, round, and client ID.
- **Configuration Management:** All operational parameters (e.g., total_rounds, clients_per_round, ADRM thresholds) must be managed through a centralized **configuration file** (config.py in ADRM and referenced in ModelManager) that can be modified without altering source code.

3.6 Other Requirements

- **Scalability:** The system must be capable of supporting **over 5,000 registered clients** and actively communicating with at least **500 clients simultaneously** per training round.
- **Performance (Aggregation Time):** The total time required for model update aggregation (including SAM and ADRM checks) should not exceed **30 seconds** per training round to maintain an efficient training pace.
- **Test Data Loading:** The system must successfully load and utilize the WUSTL-IIoT-2021 test dataset for model evaluation, as indicated by data_loader.py.

5. References

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS), 2017[1].

[2] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Khan, and A. T. Suresh, "Federated learning: Strategies for communication cost reduction," arXiv preprint arXiv: 1610.05492, 2016[2].

[3] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," IEEE Signal Processing Magazine, vol. 37, no. 4, pp. 115-125, 2020[3].

[4] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, et al., "Practical secure aggregation for federated learning on user-held data," arXiv preprint arXiv: 1712.07119, 2017[4].

[5] Y. Aono, T. Hayashi, T. Ohara, and S. Sasaki, "Privacy-preserving deep learning via homomorphic encryption," in Proceedings of the 2017 Asia Conference on Computer and Communications Security (ASIACCS), 2017[5].

[6] J. Bell and R. Lai, "Private federated learning using homomorphic encryption," in Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), 2019[6].

[7] M. Abadi, A. Chu, I. Goodfellow, H. B.. "Deep learning with differential privacy." in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS), 2016[7].

[8] Y. Wei, W. Chen, J. Zhang, and S. Guo, "DP-FedAvg: Differential Privacy Enhanced Federated Averaging for Industrial IoT," Sensors, vol. 21, no. 3, p. 856, 2021[8].

[9] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing the robustness of federated learning against poisoning attacks," in Proceedings of the 36th International Conference on Machine Learning (ICML), 2019[9].

[10] C. Xie, S. Wang, T. Long, and X. Wang, "FL-GA: A Robust and Secure Federated Learning Framework against Poisoning Attacks," in Proceedings of the 2018 IEEE International Conference on Big Data, 2018[10].

[11] Y. Cao, R. Gu, Y. Wang, and G. Chen, "Byzantine-Resilient Federated Learning for Industrial IoT with Robust Aggregation," IEEE Transactions on Industrial Informatics, 2022[11].

[12] V. A. Nguyen and N. T. Binh, "A Hierarchical Federated Learning Framework for Industrial IoT in Edge Computing," Journal of Network and Computer Applications, vol. 182, p. 103038, 2021[12].

[13] S. Zeng, Y. Li, and D. Gao, "A Survey on Federated Learning in Industrial IoT: Challenges, Applications, and Future Directions," IEEE Internet of Things Journal, 2022[13].

[14] M. Wang, J. Wang, J. Liu, and Z. Li, "Resource-Efficient Federated Learning for Industrial IoT with Data Reduction and Model Compression," IEEE Transactions on Industrial Informatics, 2020[14].