Logbook – Secure Communication and Protocol Enforcement of R25 -039

Project ID: R25 - 039

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

Student Details:

Names:	Student IDs:
Dissanayaka K.D.A.R. A	IT21828348

Supervisor: Mr. Amila Seneratha

Co-Supervisor: Mr. Tharaniyawarma Kumaralingam

Date of Submission: 2025

Contents

1. Group Details

Student Details:

Names:	Student IDs:	Research Component
Nanayakkara Y.D.T. D	IT21826368	Attack Defense and Resillence
Mendis H.R.M	IT21822612	Privacy Preseravation
Weerasinghe K.M	IT21831904	Secure Aggrigation
Dissanayaka K.D.A.R. A	IT21828348	Secure Communicaiton and Protocol Enforcement

2. Project Details

Topic - Data-Privacy Focused Federated Learning Framework for Industrial IoT

Aim – To develop a product that going to full fill the research

Deliverables – Federated Learning Framework designed for industrial internet of things

This project was initiated to develop a secure and private **Federated Learning (FL) framework** specifically for **Industrial IoT (IIoT)** environments.

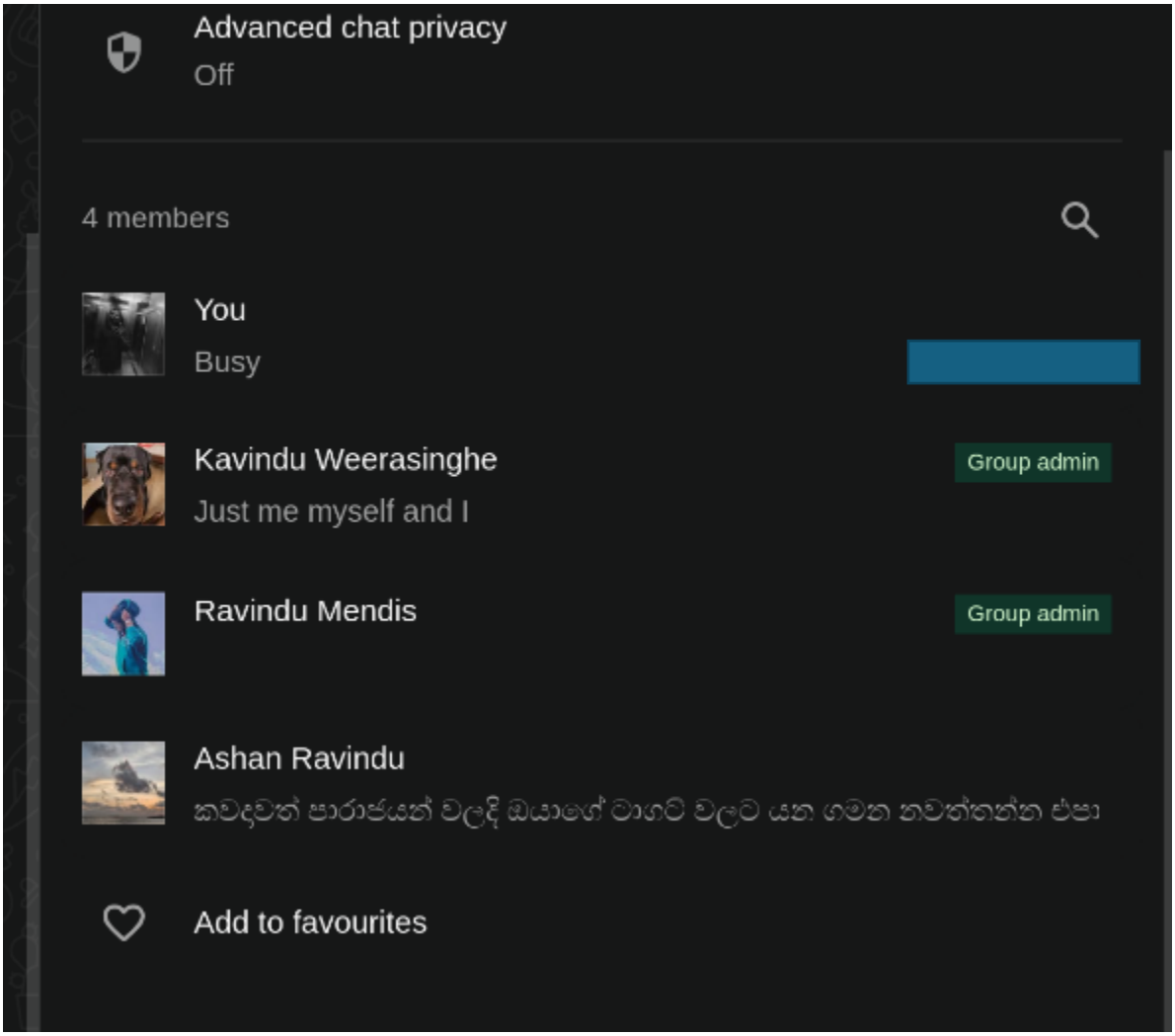
The Challenge: Traditional AI methods require centralizing sensitive factory data, which poses major **privacy risks** and clashes with the distributed nature of industrial operations. Existing FL solutions are insufficient because they fail to simultaneously provide robust security, data privacy, and efficient operation on **resource-limited IIoT devices**.

The Solution: The developed framework is a multi-layered system that provides **end-to-end protection**.

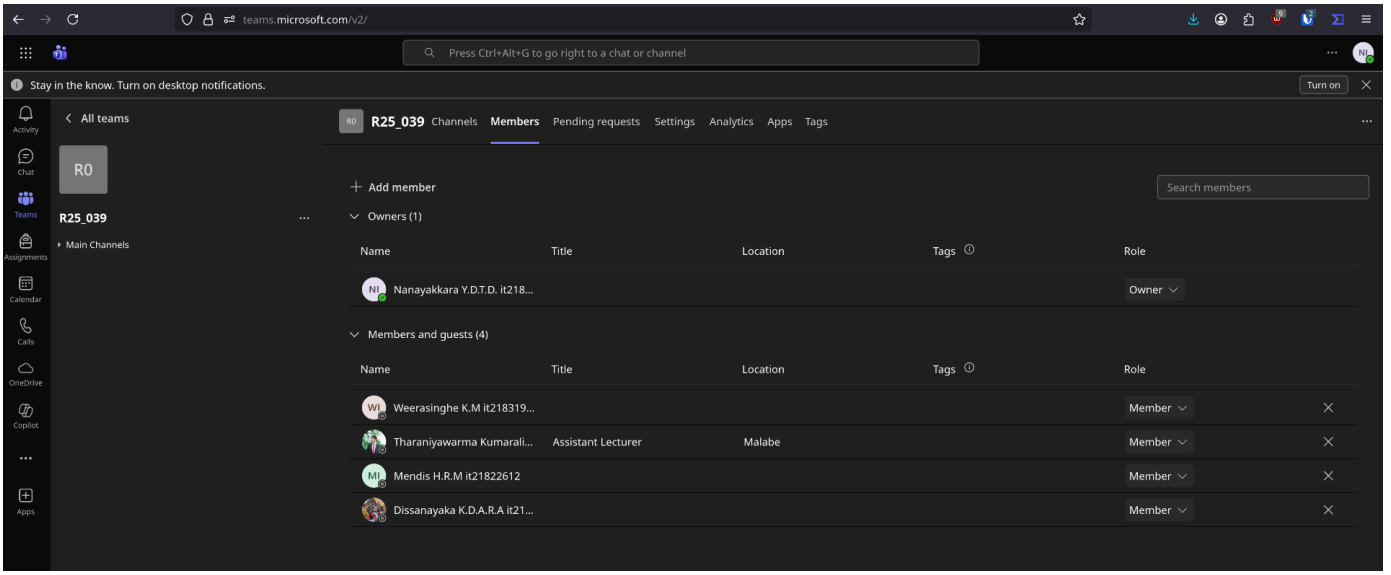
- It uses techniques like **Differential Privacy (DP)** and **Homomorphic Encryption (HE)** to guarantee data confidentiality.
- It implements a robust protocol that uses **client/server validation** to actively block cyber threats such as **Model Poisoning and Byzantine Attacks**.
- The system is optimized for **efficiency** to reduce overhead on IIoT devices.

3. Communication Methods

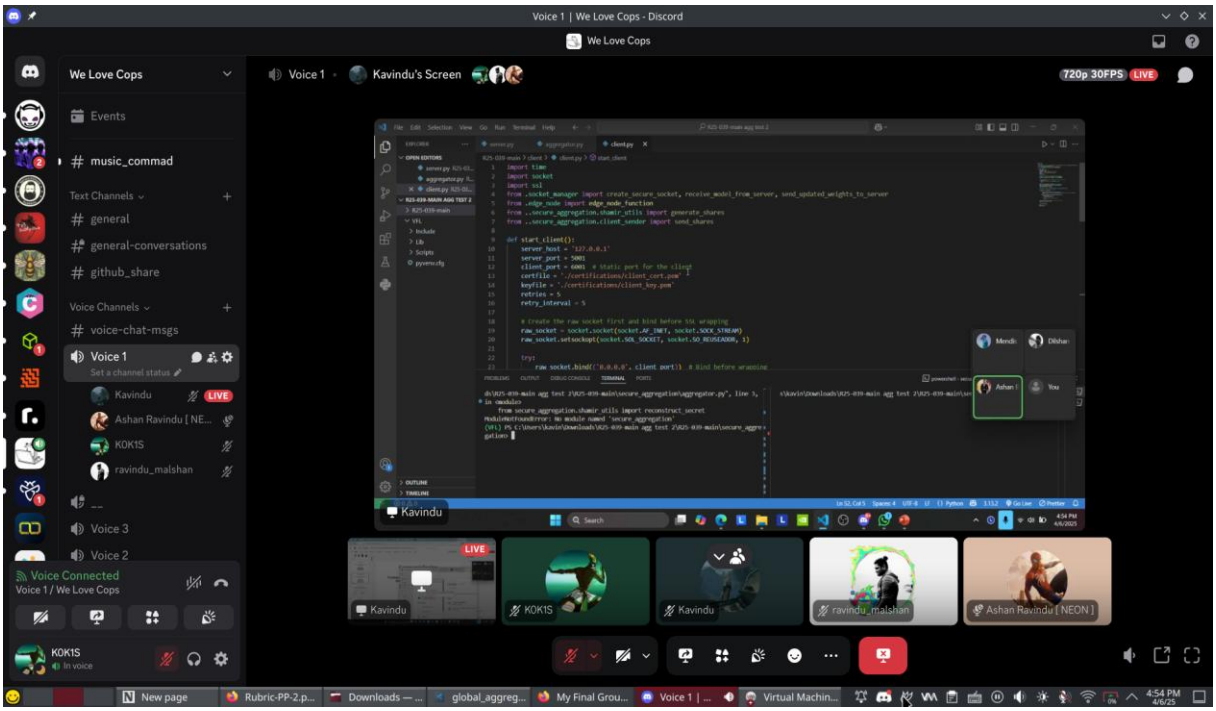
WhatsApp Group – Team



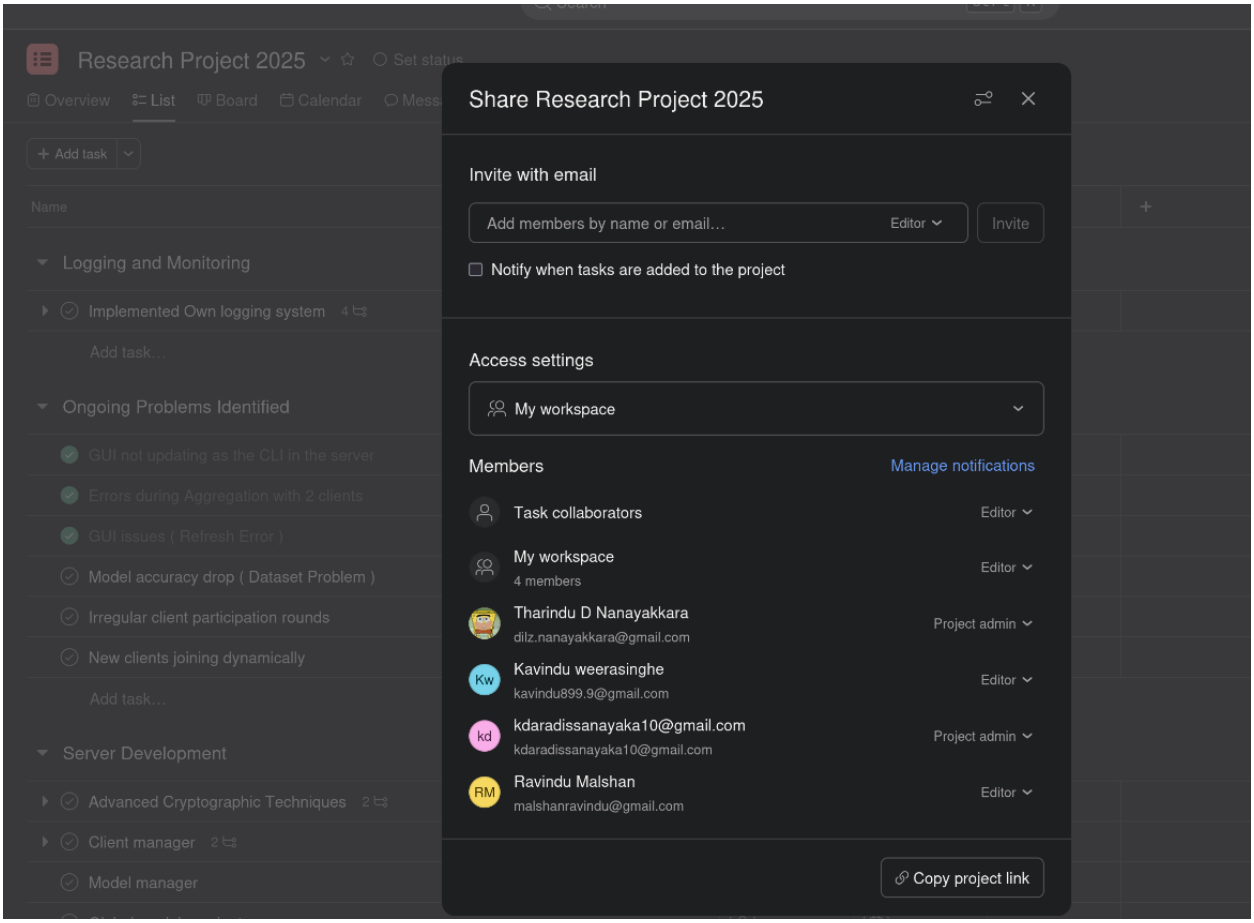
Microsoft Teams - All



Group Meetings – Discord – Team



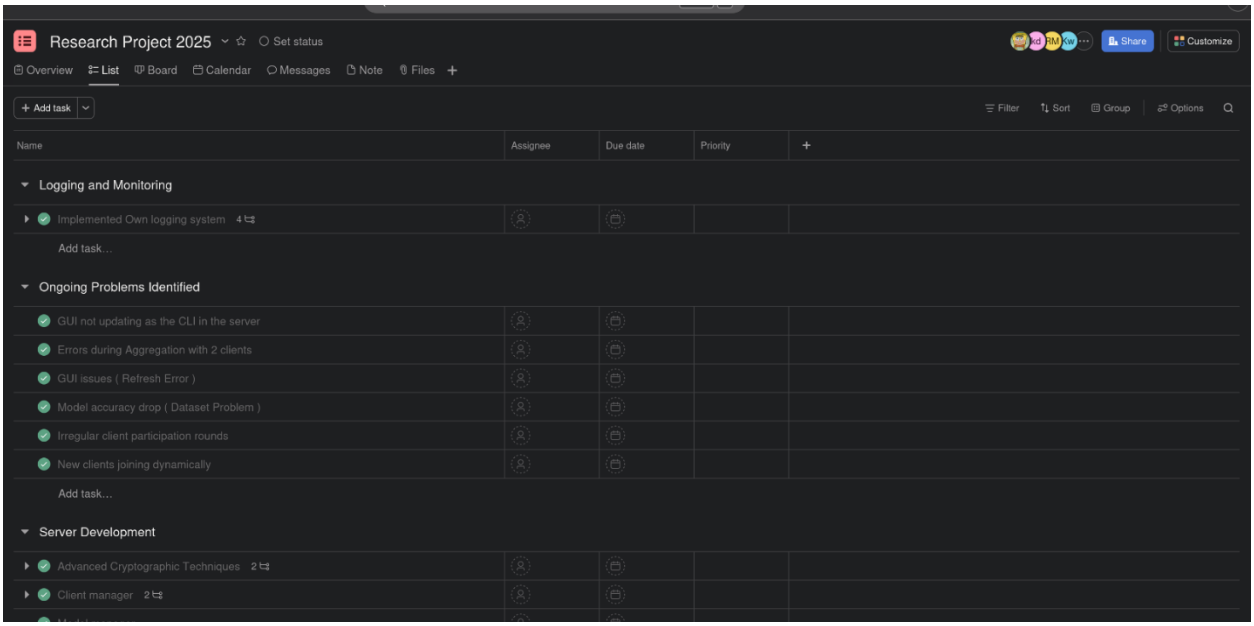
Asana – Task Assigning – Team



4. Meetings With Supervisors

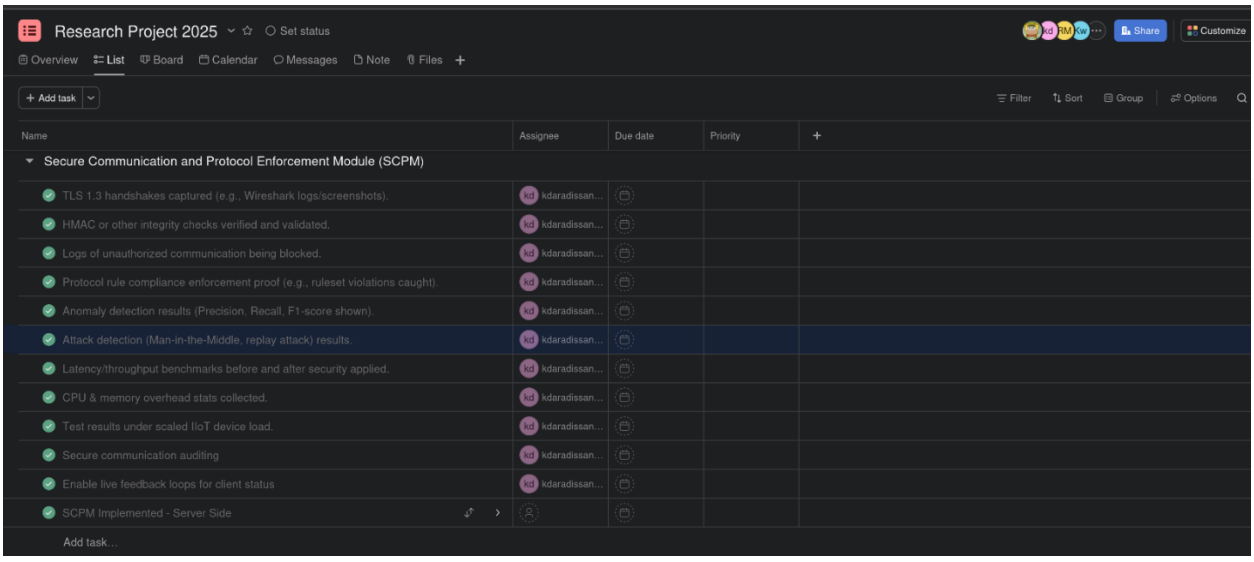
All the meetings were conducted in person and only WhatsApp calls were taken to organize the meeting

5. Task Details



Secure communication and protocol enforcement module were assigned to me.

5.1 Personal task Assigning and Completion



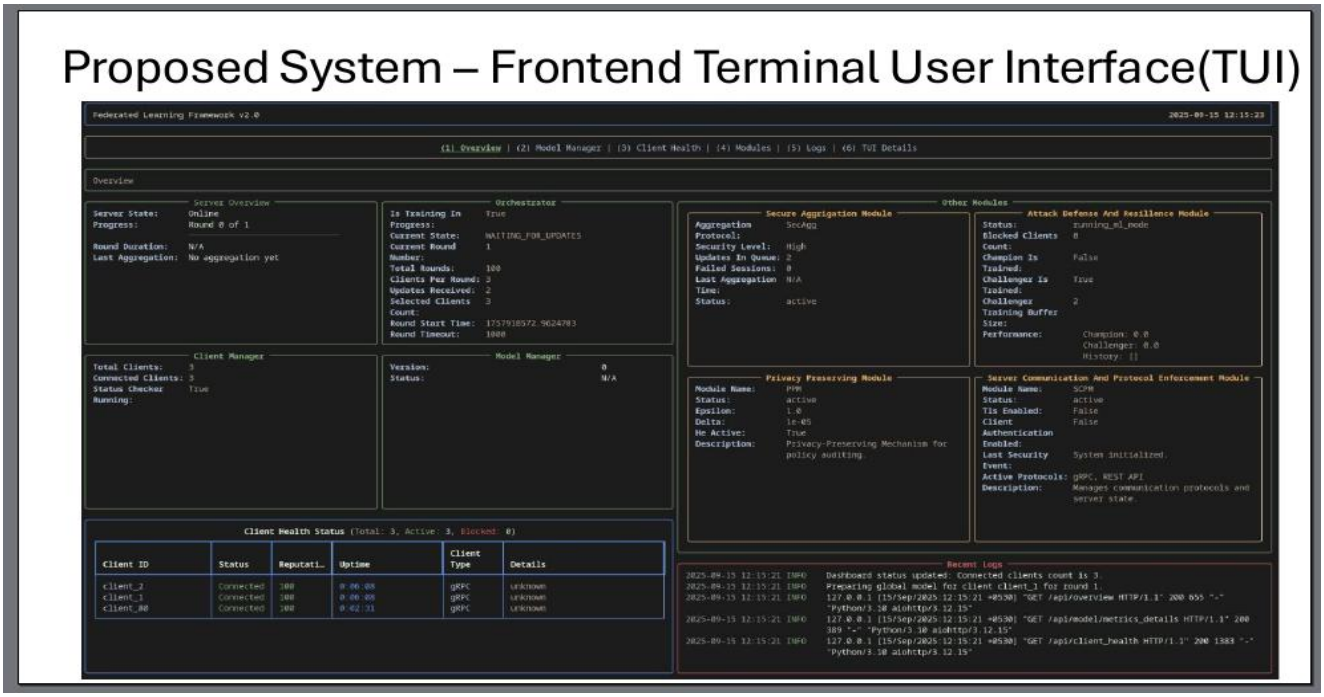
6. System Details

6.1 System completion status

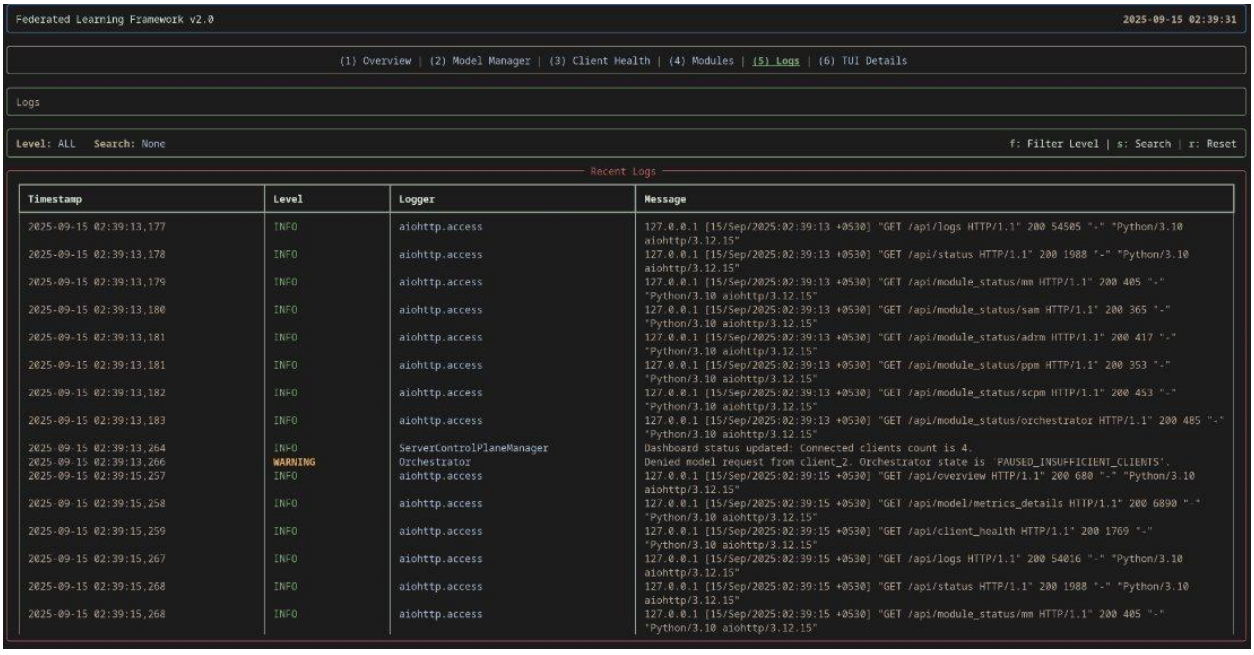
Finished

SCPM TUI

Whole system is related

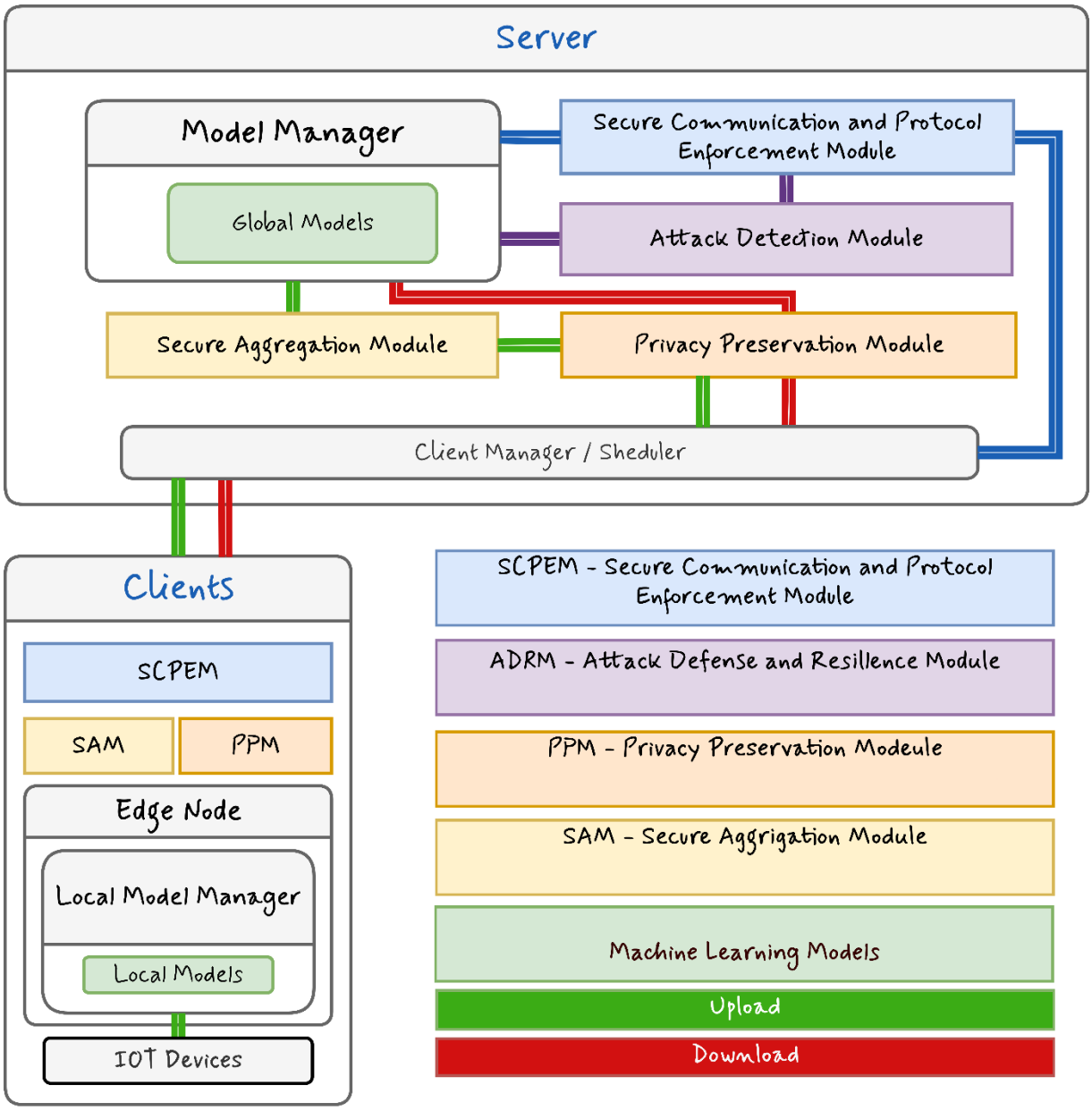


Log - Manager


Federated Learning Framework v2.0
2025-09-15 02:39:31
(1) Overview | (2) Model Manager | (3) Client Health | (4) Modules | (5) Logs | (6) TUI Details
Logs
Level: ALL Search: None
f: Filter Level | s: Search | r: Reset
Recent Logs
Timestamp
Level
Logger
Message
2025-09-15 02:39:13,177 INFO aiohttp.access 127.0.0.1 [15/Sep/2025:02:39:13 +0530] "GET /api/logs HTTP/1.1" 200 54505 "-" "Python/3.10 aiohttp/3.12.15"
2025-09-15 02:39:13,178 INFO aiohttp.access 127.0.0.1 [15/Sep/2025:02:39:13 +0530] "GET /api/status HTTP/1.1" 200 1988 "-" "Python/3.10 aiohttp/3.12.15"
2025-09-15 02:39:13,179 INFO aiohttp.access 127.0.0.1 [15/Sep/2025:02:39:13 +0530] "GET /api/module_status/mm HTTP/1.1" 200 405 "-" "Python/3.10 aiohttp/3.12.15"
2025-09-15 02:39:13,180 INFO aiohttp.access 127.0.0.1 [15/Sep/2025:02:39:13 +0530] "GET /api/module_status/sam HTTP/1.1" 200 365 "-" "Python/3.10 aiohttp/3.12.15"
2025-09-15 02:39:13,181 INFO aiohttp.access 127.0.0.1 [15/Sep/2025:02:39:13 +0530] "GET /api/module_status/adrm HTTP/1.1" 200 417 "-" "Python/3.10 aiohttp/3.12.15"
2025-09-15 02:39:13,181 INFO aiohttp.access 127.0.0.1 [15/Sep/2025:02:39:13 +0530] "GET /api/module_status/ppm HTTP/1.1" 200 353 "-" "Python/3.10 aiohttp/3.12.15"
2025-09-15 02:39:13,182 INFO aiohttp.access 127.0.0.1 [15/Sep/2025:02:39:13 +0530] "GET /api/module_status/scpm HTTP/1.1" 200 453 "-" "Python/3.10 aiohttp/3.12.15"
2025-09-15 02:39:13,183 INFO aiohttp.access 127.0.0.1 [15/Sep/2025:02:39:13 +0530] "GET /api/module_status/orchestrator HTTP/1.1" 200 485 "-" "Python/3.10 aiohttp/3.12.15"
2025-09-15 02:39:13,264 INFO ServerControlPlaneManager Dashboard status updated: Connected clients count is 4.
2025-09-15 02:39:13,266 WARNING Orchestrator Denied model request from client_2. Orchestrator state is 'PAUSED_INSUFFICIENT_CLIENTS'.
2025-09-15 02:39:15,257 INFO aiohttp.access 127.0.0.1 [15/Sep/2025:02:39:15 +0530] "GET /api/overview HTTP/1.1" 200 680 "-" "Python/3.10 aiohttp/3.12.15"
2025-09-15 02:39:15,258 INFO aiohttp.access 127.0.0.1 [15/Sep/2025:02:39:15 +0530] "GET /api/model/metrics_details HTTP/1.1" 200 6890 "-" "Python/3.10 aiohttp/3.12.15"
2025-09-15 02:39:15,259 INFO aiohttp.access 127.0.0.1 [15/Sep/2025:02:39:15 +0530] "GET /api/client_health HTTP/1.1" 200 1769 "-" "Python/3.10 aiohttp/3.12.15"
2025-09-15 02:39:15,267 INFO aiohttp.access 127.0.0.1 [15/Sep/2025:02:39:15 +0530] "GET /api/logs HTTP/1.1" 200 54016 "-" "Python/3.10 aiohttp/3.12.15"
2025-09-15 02:39:15,268 INFO aiohttp.access 127.0.0.1 [15/Sep/2025:02:39:15 +0530] "GET /api/status HTTP/1.1" 200 1988 "-" "Python/3.10 aiohttp/3.12.15"
2025-09-15 02:39:15,268 INFO aiohttp.access 127.0.0.1 [15/Sep/2025:02:39:15 +0530] "GET /api/module_status/mm HTTP/1.1" 200 405 "-" "Python/3.10 aiohttp/3.12.15"

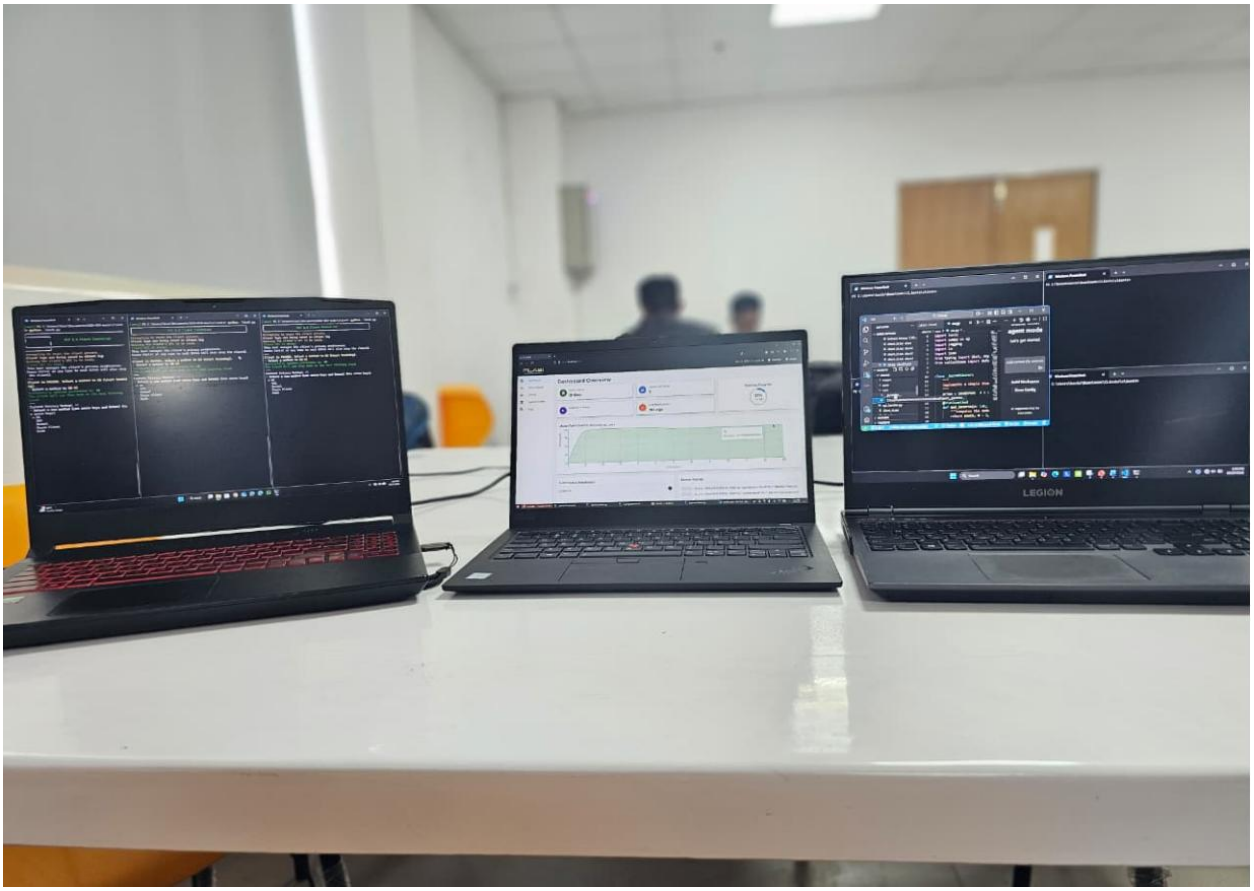
6.2 System Design

I. System Architecture



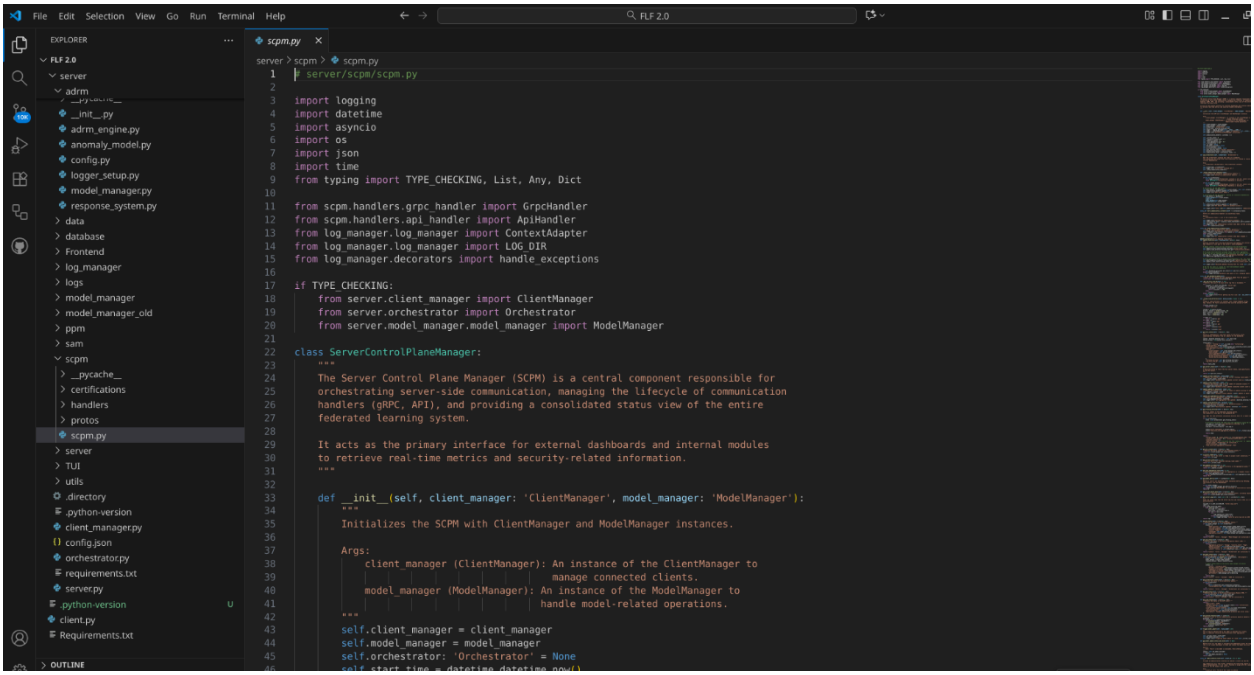
II. Module Architecture

6.3 System Testing



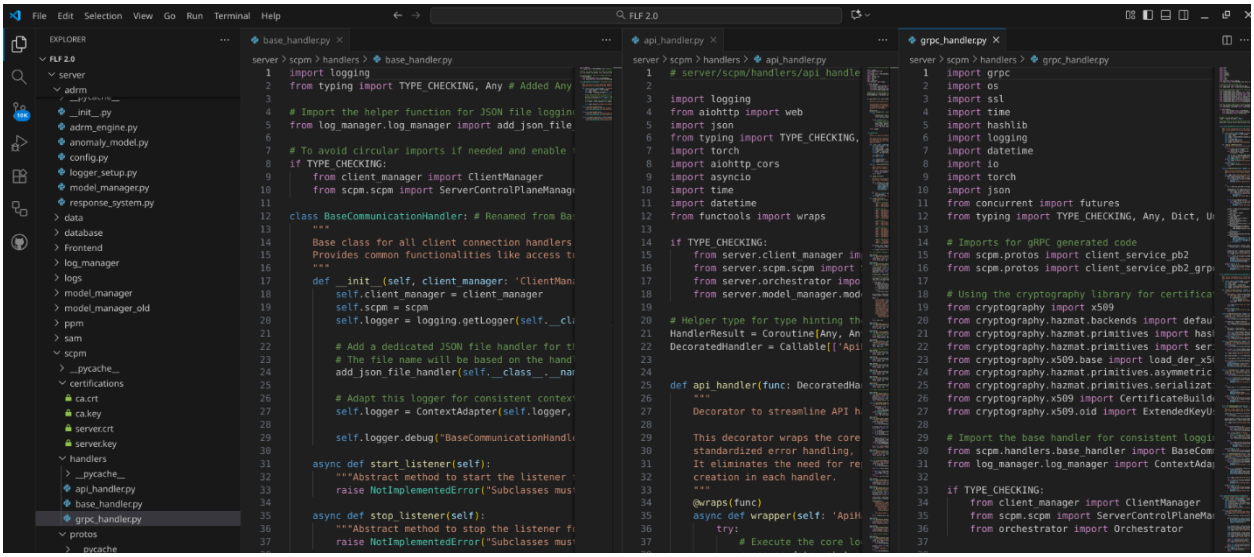
6.4 System Codes

SCPM Server



```python
# server/scpm/scpm.py

import logging
import datetime
import asyncio
import os
import json
import time
from typing import TYPE_CHECKING, List, Any, Dict

from scpm.handlers.grpc_handler import GrpcHandler
from scpm.handlers.api_handler import ApiHandler
from log_manager.log_manager import ContextAdapter
from log_manager.log_manager import LOG_DIR
from log_manager.decorators import handle_exceptions

if TYPE_CHECKING:
    from server.client_manager import ClientManager
    from server.orchestrator import Orchestrator
    from server.model_manager.model_manager import ModelManager

class ServerControlPlaneManager:
    """
    The Server Control Plane Manager (SCPM) is a central component responsible for
    orchestrating server-side communication, managing the lifecycle of communication
    handlers (gRPC, API), and providing a consolidated status view of the entire
    federated learning system.

    It acts as the primary interface for external dashboards and internal modules
    to retrieve real-time metrics and security-related information.
    """

    def __init__(self, client_manager: 'ClientManager', model_manager: 'ModelManager'):
        """
        Initializes the SCPM with ClientManager and ModelManager instances.

        Args:
            client_manager (ClientManager): An instance of the ClientManager to
                                            manage connected clients.
            model_manager (ModelManager): An instance of the ModelManager to
                                          handle model-related operations.
        """
        self.client_manager = client_manager
        self.model_manager = model_manager
        self.orchestrator: 'Orchestrator' = None
```

Handlers – Communication

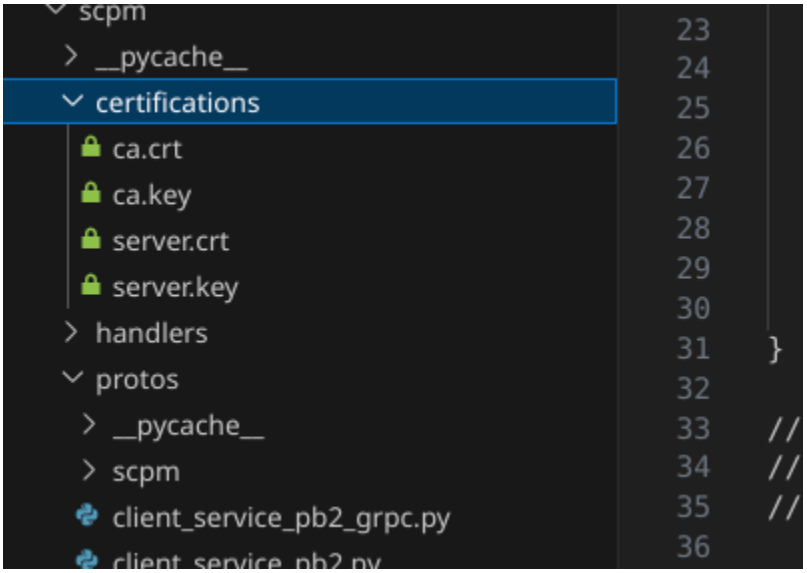


```python
import logging
from typing import TYPE_CHECKING, Any # Added Any

# Import the helper function for JSON file logging
from log_manager.log_manager import add_json_file

# To avoid circular imports if needed and enable
if TYPE_CHECKING:
    from client_manager import ClientManager
    from scpm.scpm import ServerControlPlaneManag

class BaseCommunicationHandler: # Renamed from Ba
    """
    Base class for all client connection handlers
    Provides common functionalities like access t
    """
    def __init__(self, client_manager: 'ClientMan
        self.client_manager = client_manager
        self.scpm = scpm
        self.logger = logging.getLogger(self.__cl

        # Add a dedicated JSON file handler for t
        # The file name will be based on the hand
        add_json_file_handler(self.__class__.__na

        # Adapt this logger for consistent contex
        self.logger = ContextAdapter(self.logger,

        self.logger.debug("BaseCommunicationHandl

    async def start_listener(self):
        """Abstract method to start the listener
        raise NotImplementedError("Subclasses mus

    async def stop_listener(self):
        """Abstract method to stop the listener f
        raise NotImplementedError("Subclasses mus
```

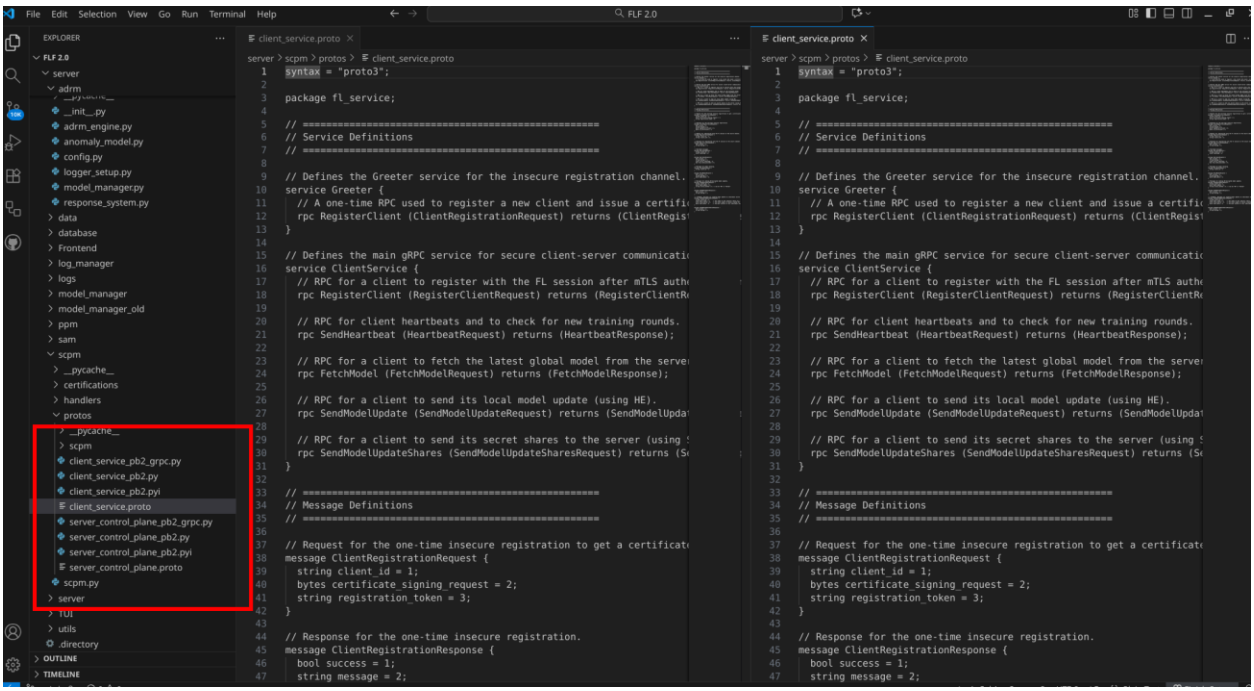
```python
# server/scpm/handlers/api_handle

import logging
from aiohttp import web
import json
from typing import TYPE_CHECKING,
import torch
import aiohttp_cors
import asyncio
import time
import datetime
from functools import wraps

if TYPE_CHECKING:
    from server.client_manager im
    from server.scpm.scpm import
    from server.orchestrator impo
    from server.model_manager.mod

# Helper type for type hinting th
HandlerResult = Coroutine[Any, An
DecoratedHandler = Callable[['Api

def api_handler(func: DecoratedHa
    """
    Decorator to streamline API h

    This decorator wraps the core
    standardized error handling,
    It eliminates the need for re
    creation in each handler.
    """
    @wraps(func)
    async def wrapper(self: 'ApiH
        try:
            # Execute the core lo
```

```python
import grpc
import os
import ssl
import time
import hashlib
import logging
import datetime
import io
import torch
import json
from concurrent import futures
from typing import TYPE_CHECKING, Any, Dict, U

# Imports for gRPC generated code
from scpm.protos import client_service_pb2
from scpm.protos import client_service_pb2_grp

# Using the cryptography library for certifica
from cryptography import x509
from cryptography.hazmat.backends import defau
from cryptography.hazmat.primitives import hash
from cryptography.hazmat.primitives import ser
from cryptography.x509.base import load_der_x5
from cryptography.hazmat.primitives.asymmetric
from cryptography.hazmat.primitives.serializat
from cryptography.x509 import CertificateBuild
from cryptography.x509.oid import ExtendedKeyU

# Import the base handler for consistent loggi
from scpm.handlers.base_handler import BaseCom
from log_manager.log_manager import ContextAda

if TYPE_CHECKING:
    from client_manager import ClientManager
    from scpm.scpm import ServerControlPlaneMa
    from orchestrator import Orchestrator
```

Certificaiton – Server

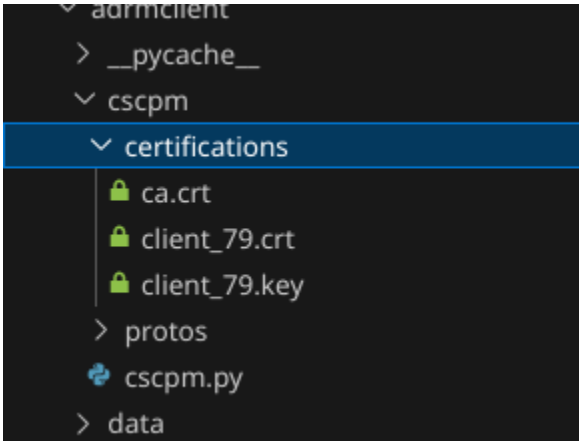


Protos – server

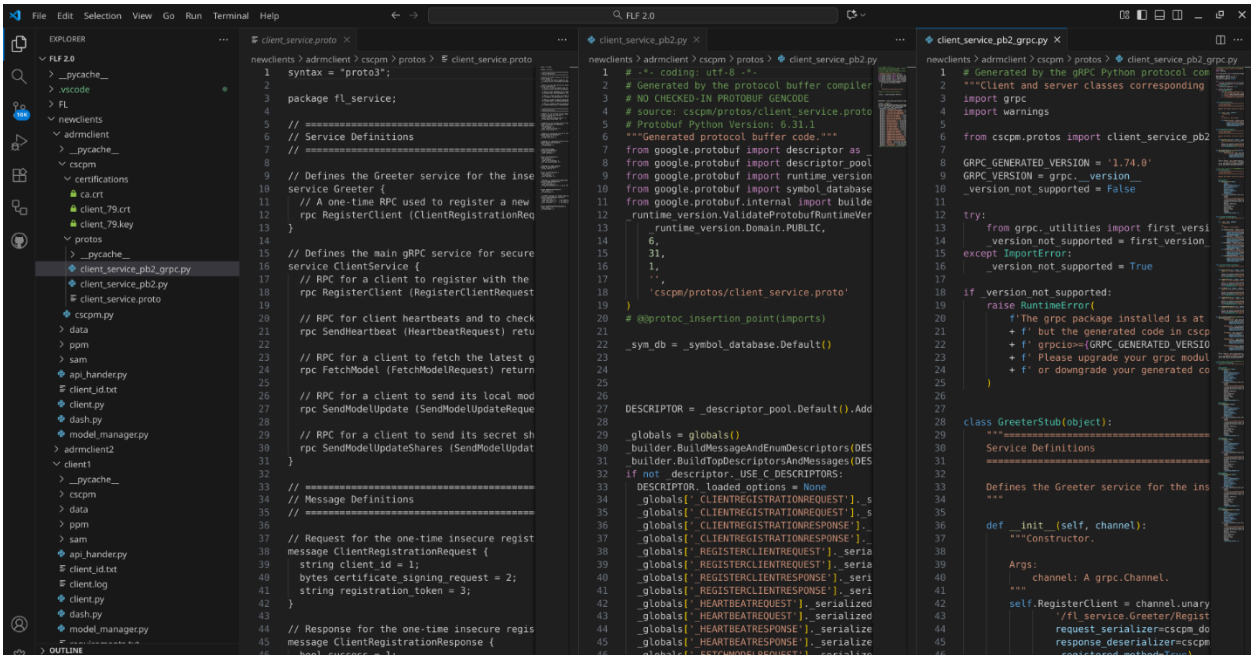


SCPM (Client SCPM) – Client

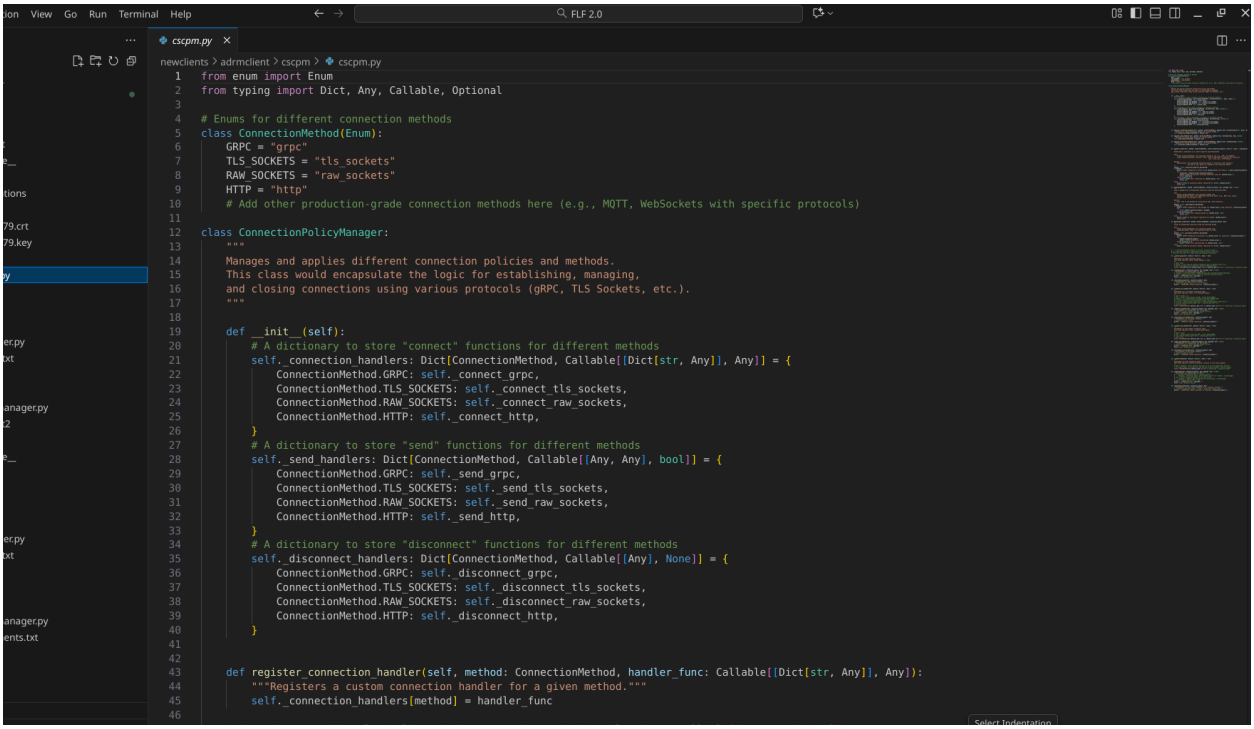
Certifications Create – Registered Client



Protos

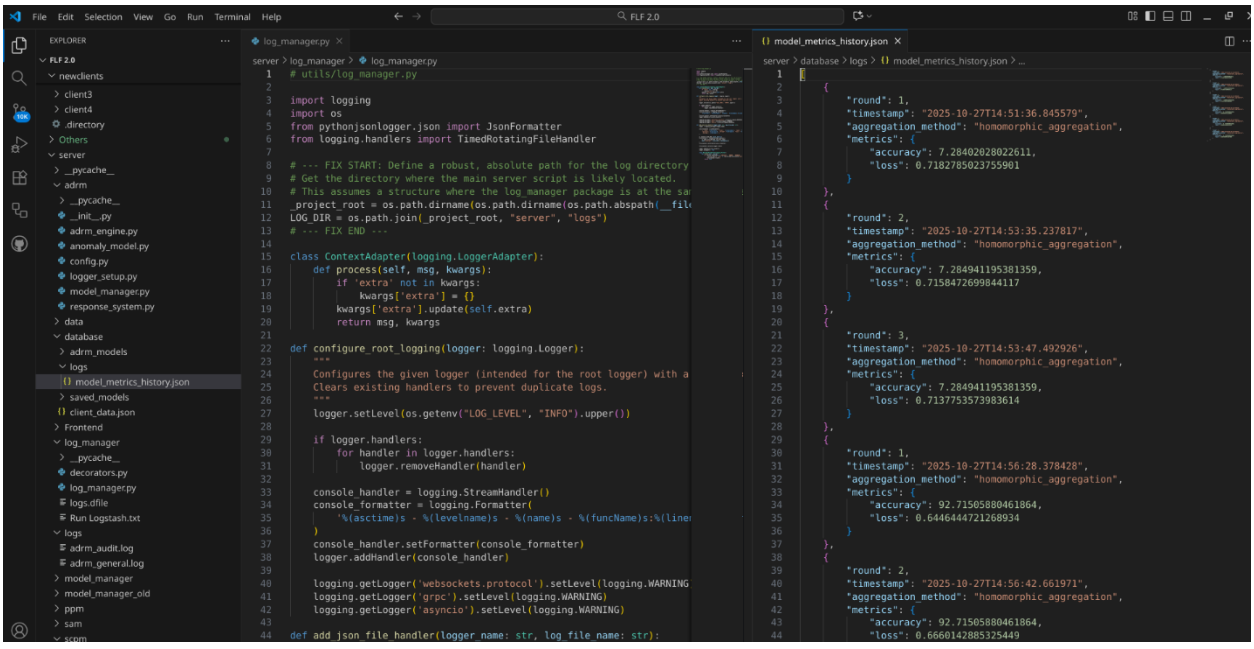


CSCPM



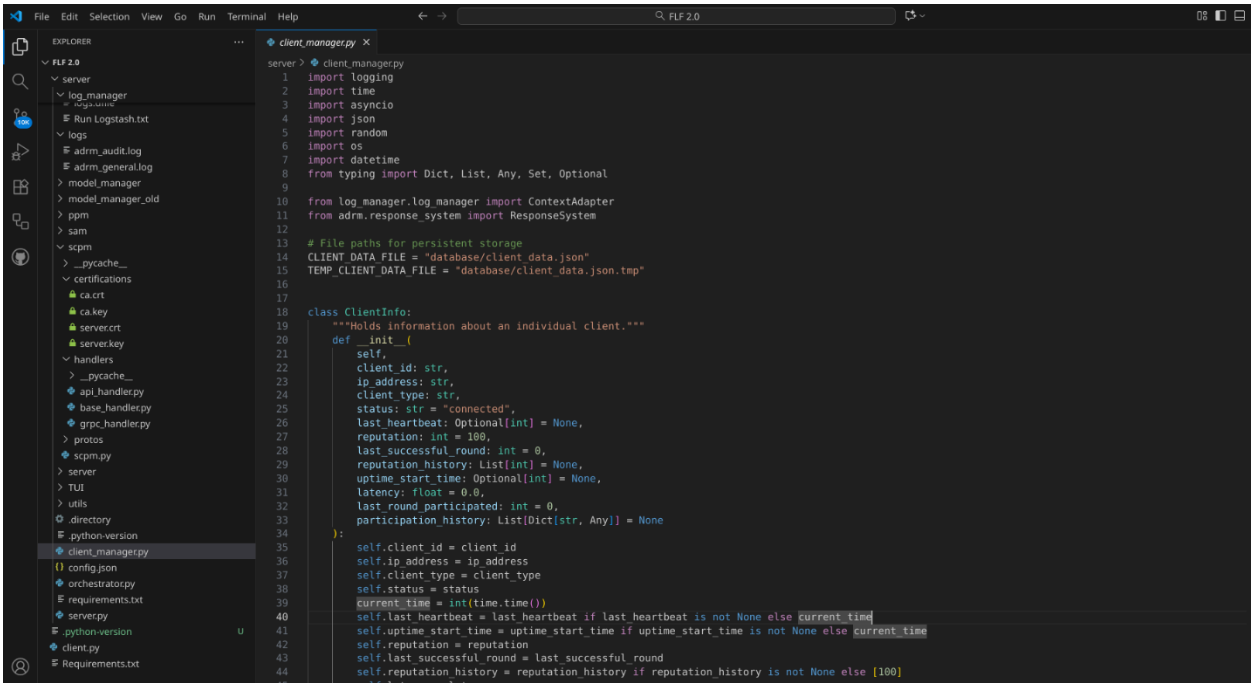
```python
from enum import Enum
from typing import Dict, Any, Callable, Optional

# Enums for different connection methods
class ConnectionMethod(Enum):
    GRPC = "grpc"
    TLS_SOCKETS = "tls_sockets"
    RAW_SOCKETS = "raw_sockets"
    HTTP = "http"
    # Add other production-grade connection methods here (e.g., MQTT, WebSockets with specific protocols)

class ConnectionPolicyManager:
    """
    Manages and applies different connection policies and methods.
    This class would encapsulate the logic for establishing, managing,
    and closing connections using various protocols (gRPC, TLS Sockets, etc.).
    """

    def __init__(self):
        # A dictionary to store "connect" functions for different methods
        self._connection_handlers: Dict[ConnectionMethod, Callable[[Dict[str, Any]], Any]] = {
            ConnectionMethod.GRPC: self._connect_grpc,
            ConnectionMethod.TLS_SOCKETS: self._connect_tls_sockets,
            ConnectionMethod.RAW_SOCKETS: self._connect_raw_sockets,
            ConnectionMethod.HTTP: self._connect_http,
        }
        # A dictionary to store "send" functions for different methods
        self._send_handlers: Dict[ConnectionMethod, Callable[[Any, Any], bool]] = {
            ConnectionMethod.GRPC: self._send_grpc,
            ConnectionMethod.TLS_SOCKETS: self._send_tls_sockets,
            ConnectionMethod.RAW_SOCKETS: self._send_raw_sockets,
            ConnectionMethod.HTTP: self._send_http,
        }
        # A dictionary to store "disconnect" functions for different methods
        self._disconnect_handlers: Dict[ConnectionMethod, Callable[[Any], None]] = {
            ConnectionMethod.GRPC: self._disconnect_grpc,
            ConnectionMethod.TLS_SOCKETS: self._disconnect_tls_sockets,
            ConnectionMethod.RAW_SOCKETS: self._disconnect_raw_sockets,
            ConnectionMethod.HTTP: self._disconnect_http,
        }


    def register_connection_handler(self, method: ConnectionMethod, handler_func: Callable[[Dict[str, Any]], Any]):
        """Registers a custom connection handler for a given method."""
        self._connection_handlers[method] = handler_func
```

Log Manager

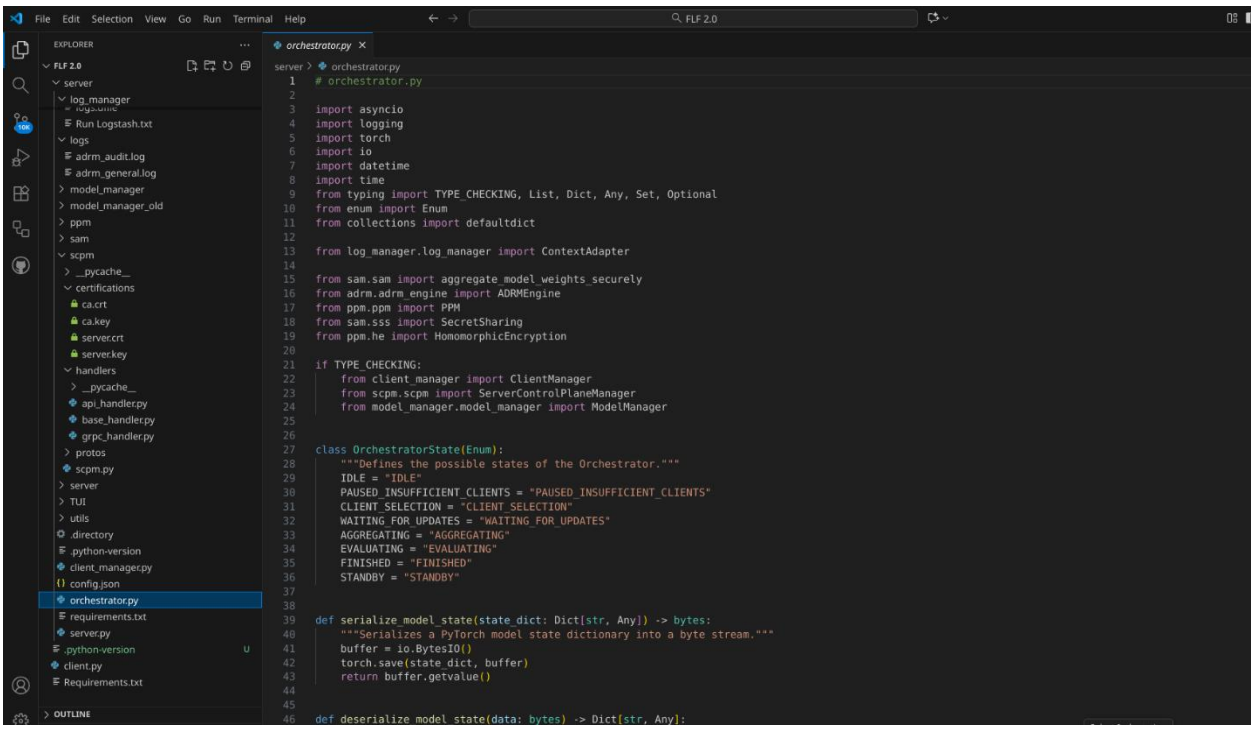


```python
# utils/log_manager.py

import logging
import os
from pythonjsonlogger.json import JsonFormatter
from logging.handlers import TimedRotatingFileHandler

# --- FIX START: Define a robust, absolute path for the log directory ---
# Get the directory where the main server script is likely located.
# This assumes a structure where the log_manager package is at the sa
_project_root = os.path.dirname(os.path.dirname(os.path.abspath(__fil
LOG_DIR = os.path.join(_project_root, "server", "logs")
# --- FIX END ---

class ContextAdapter(logging.LoggerAdapter):
    def process(self, msg, kwargs):
        if 'extra' not in kwargs:
            kwargs['extra'] = {}
        kwargs['extra'].update(self.extra)
        return msg, kwargs

def configure_root_logging(logger: logging.Logger):
    """
    Configures the given logger (intended for the root logger) with a
    Clears existing handlers to prevent duplicate logs.
    """
    logger.setLevel(os.getenv("LOG_LEVEL", "INFO").upper())

    if logger.handlers:
        for handler in logger.handlers:
            logger.removeHandler(handler)

    console_handler = logging.StreamHandler()
    console_formatter = logging.Formatter(
        '%(asctime)s - %(levelname)s - %(name)s - %(funcName)s:%(line
    )
    console_handler.setFormatter(console_formatter)
    logger.addHandler(console_handler)

    logging.getLogger('websockets.protocol').setLevel(logging.WARNING
    logging.getLogger('grpc').setLevel(logging.WARNING)
    logging.getLogger('asyncio').setLevel(logging.WARNING)

def add_json_file_handler(logger_name: str, log_file_name: str):
```

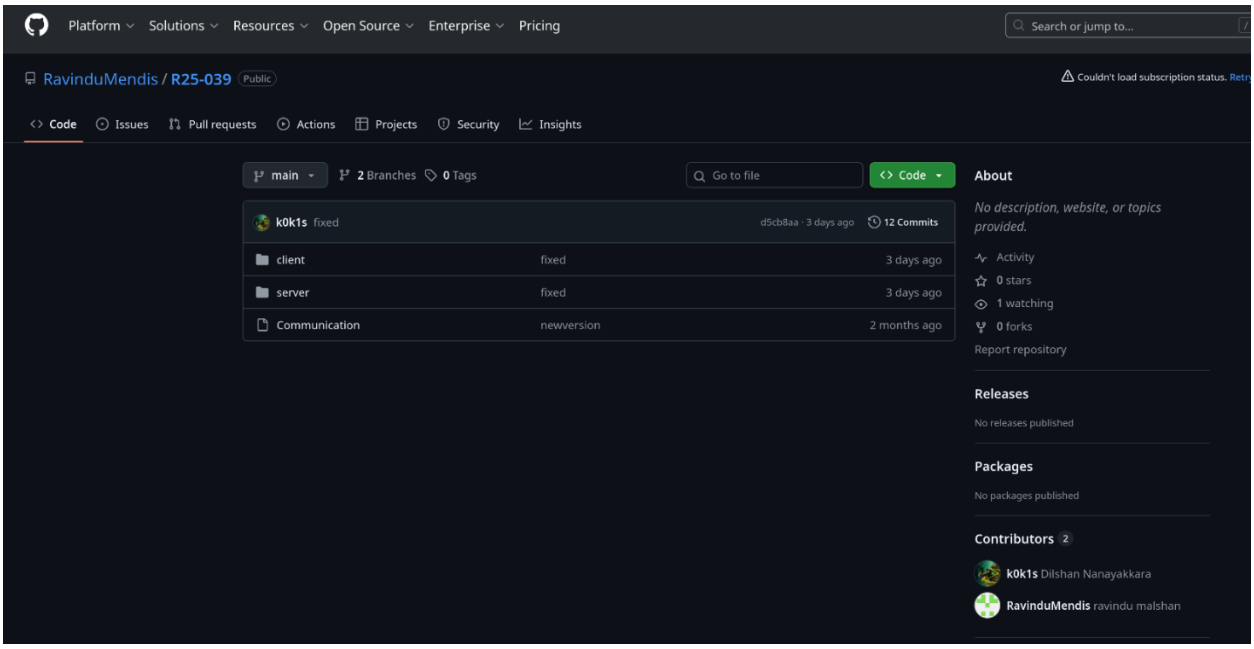
Client Manager



Orchestrator

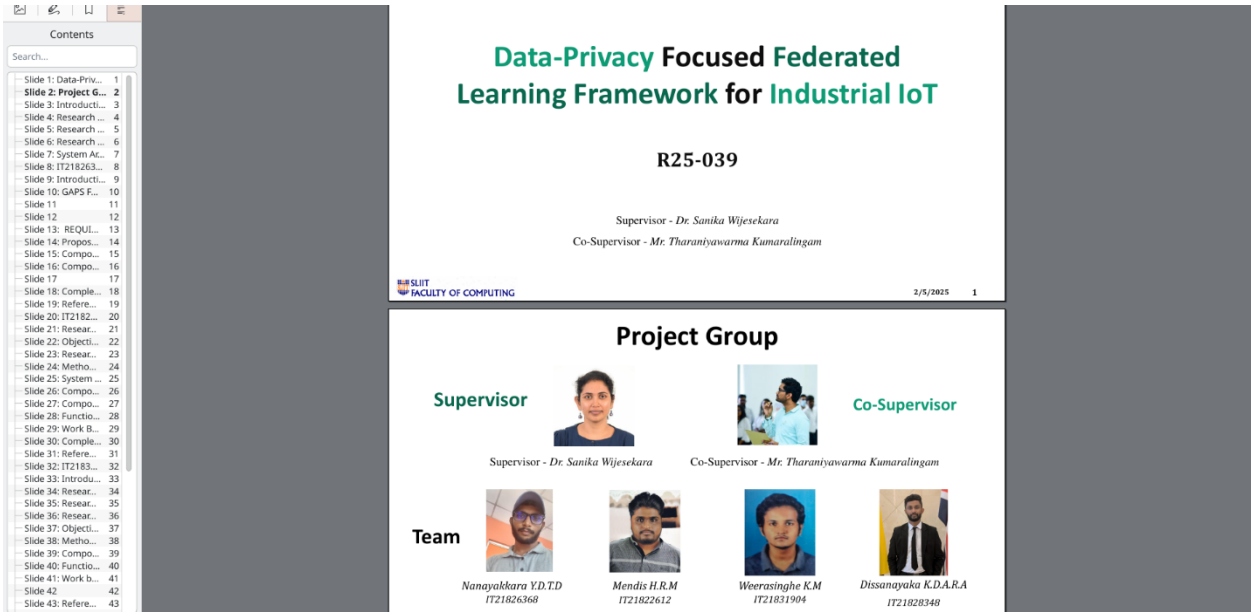


7. GitHub Upload

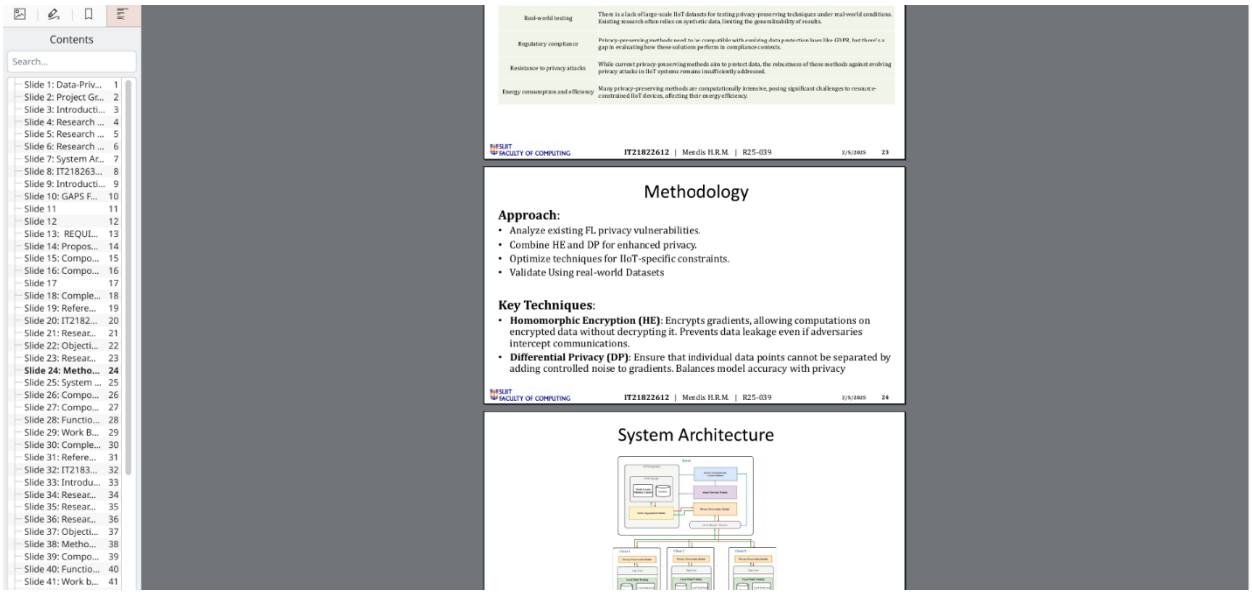


8. Documentation

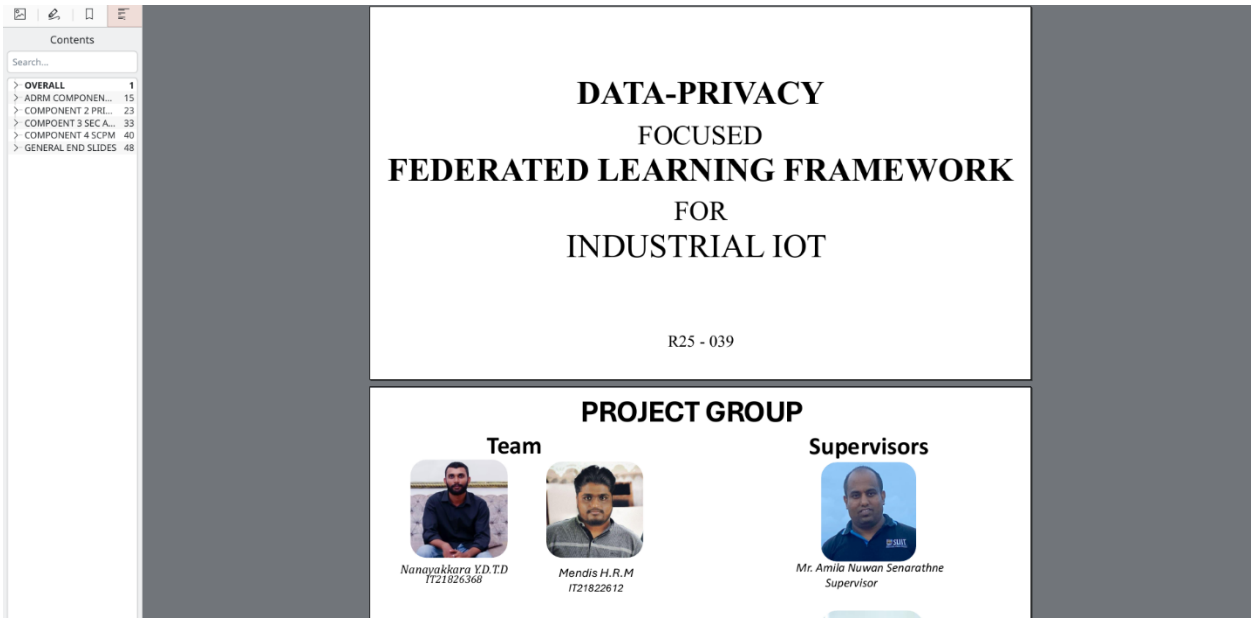
8.1 Proposal



8.2 Presentation 1



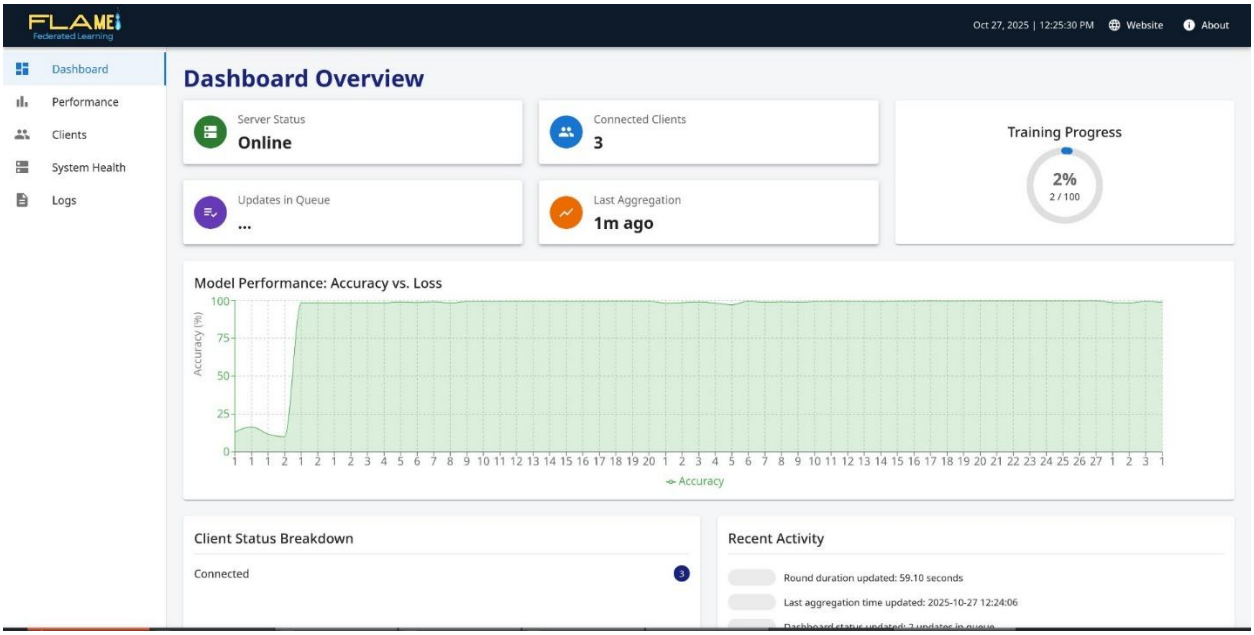
8.3 Presentation 2



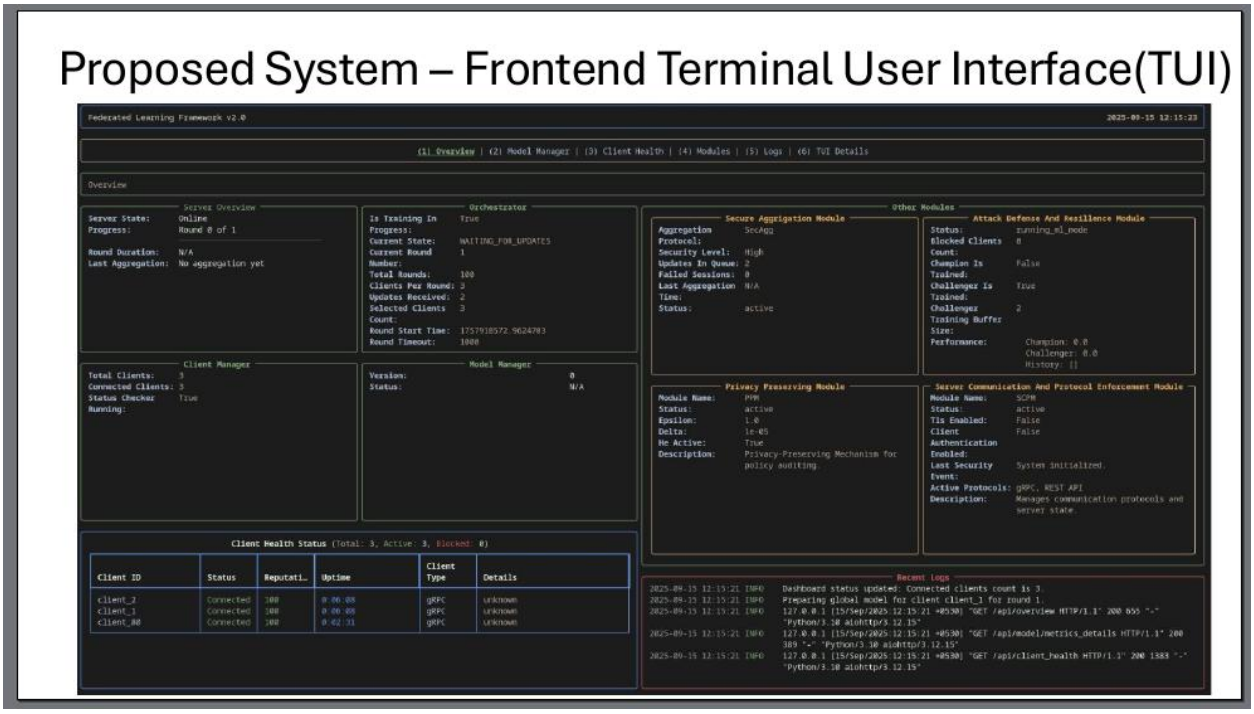
8.4 Final Presentation

8.5 Final Product

Web Portal Frontend



Terminal User interface (Frontend)



8.6 Research Paper

III. Conference Appetence

To Tharindu D Nanayakkara <dilz.nanayakkara@gmail.com> 10/29/25, 11:27 AM

Acceptance Notification

Dear Tharindu D Nanayakkara,

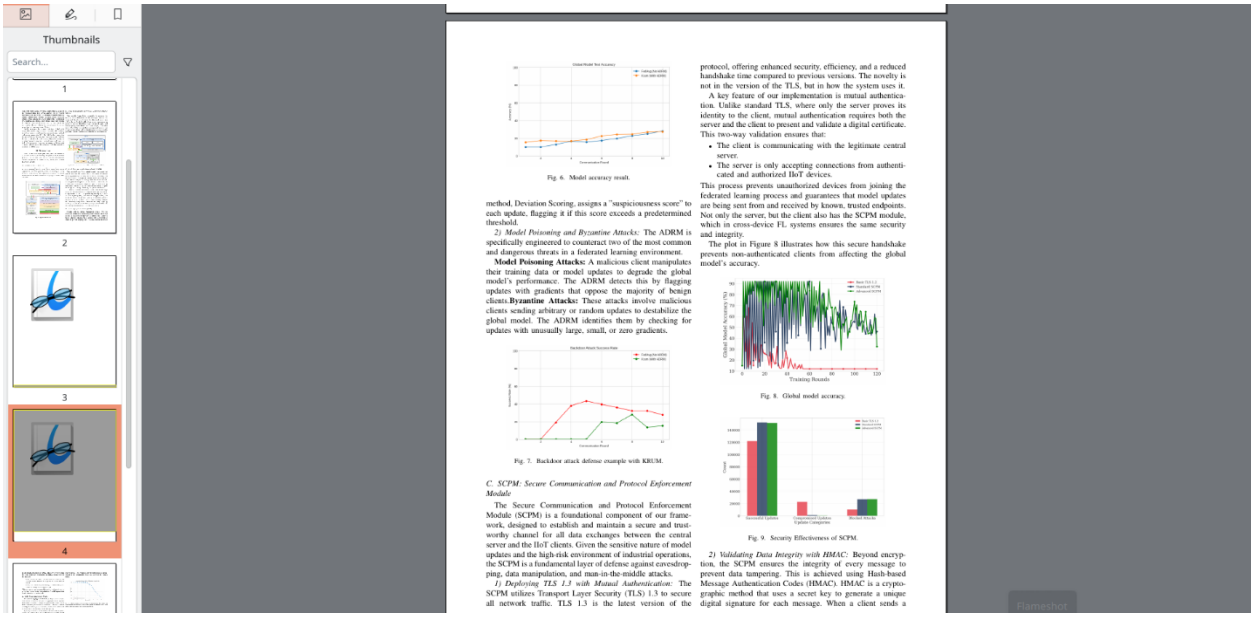
Congratulations! We are pleased to inform you that your paper has been accepted as a regular paper to be presented at the 7th International Conference on Advancements in Computing 2025.

Paper ID: 469
Paper Title: Data-privacy based Federated Learning Framework for Industrial IOT

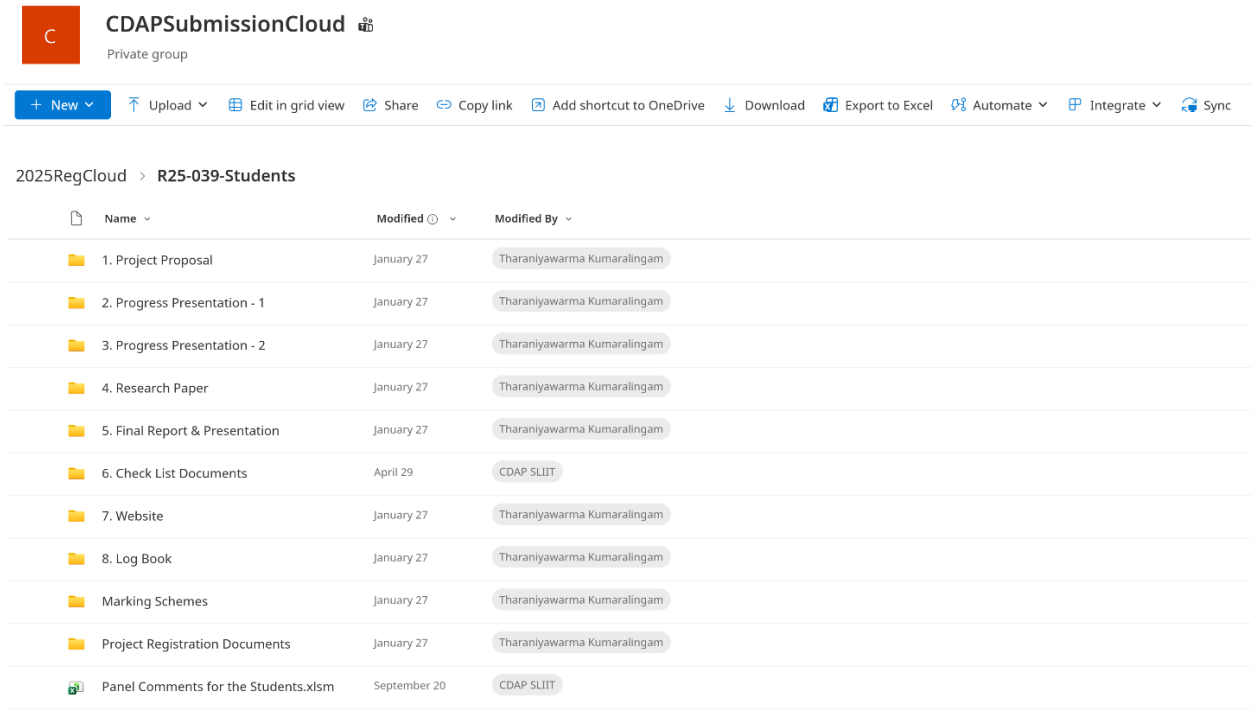
Please visit https://cmt3.research.microsoft.com/7ICAC2025/Submission/Index to view the reviews given during the double-blind review process.

When preparing the camera-ready version of your paper, please address all the review comments and follow the camera-ready guidelines given in the https://icac.lk/for-authors

Please note that the camera-ready deadline is 10th November 2025 and camera-ready submission portal on CMT will be available starting from 22nd October 2025.

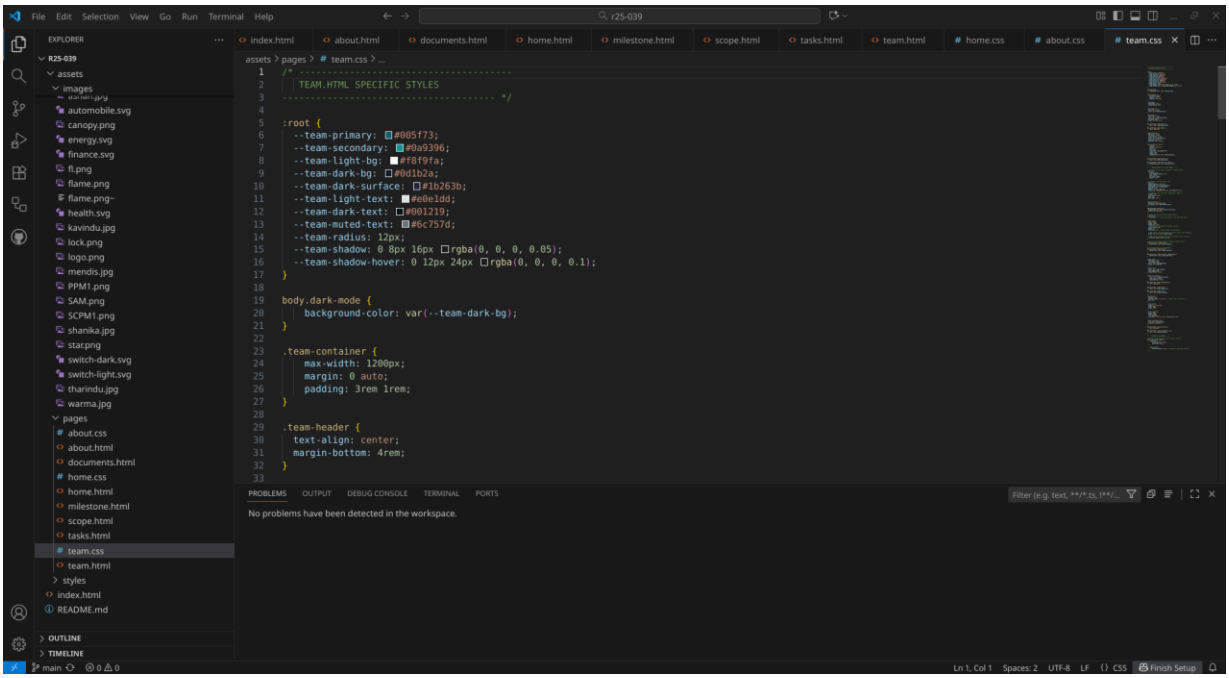


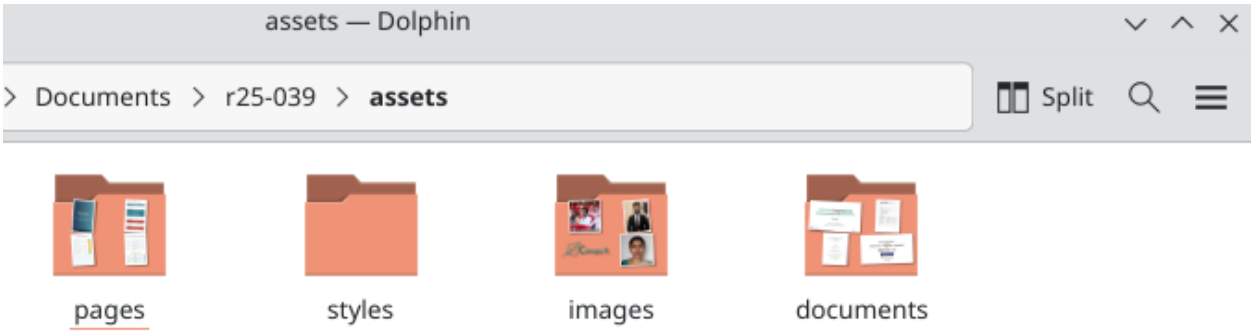
9. CDAP upload

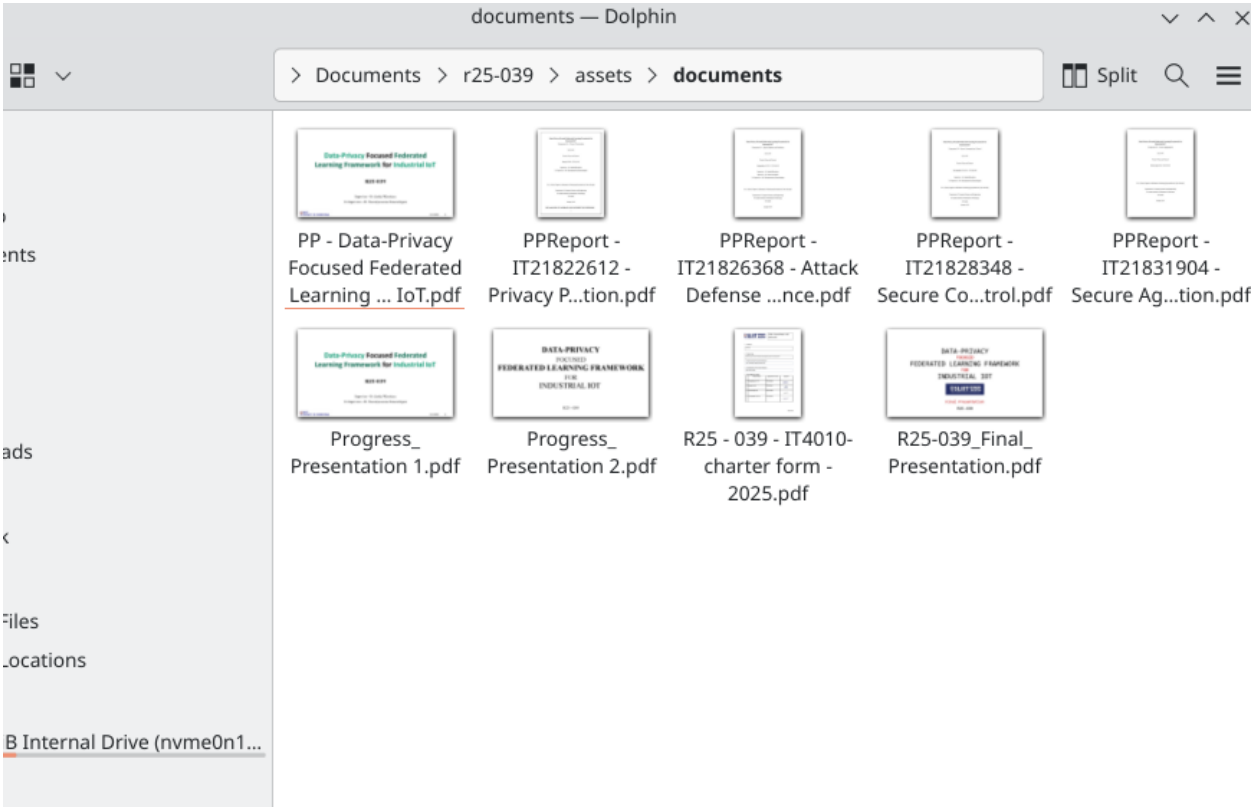


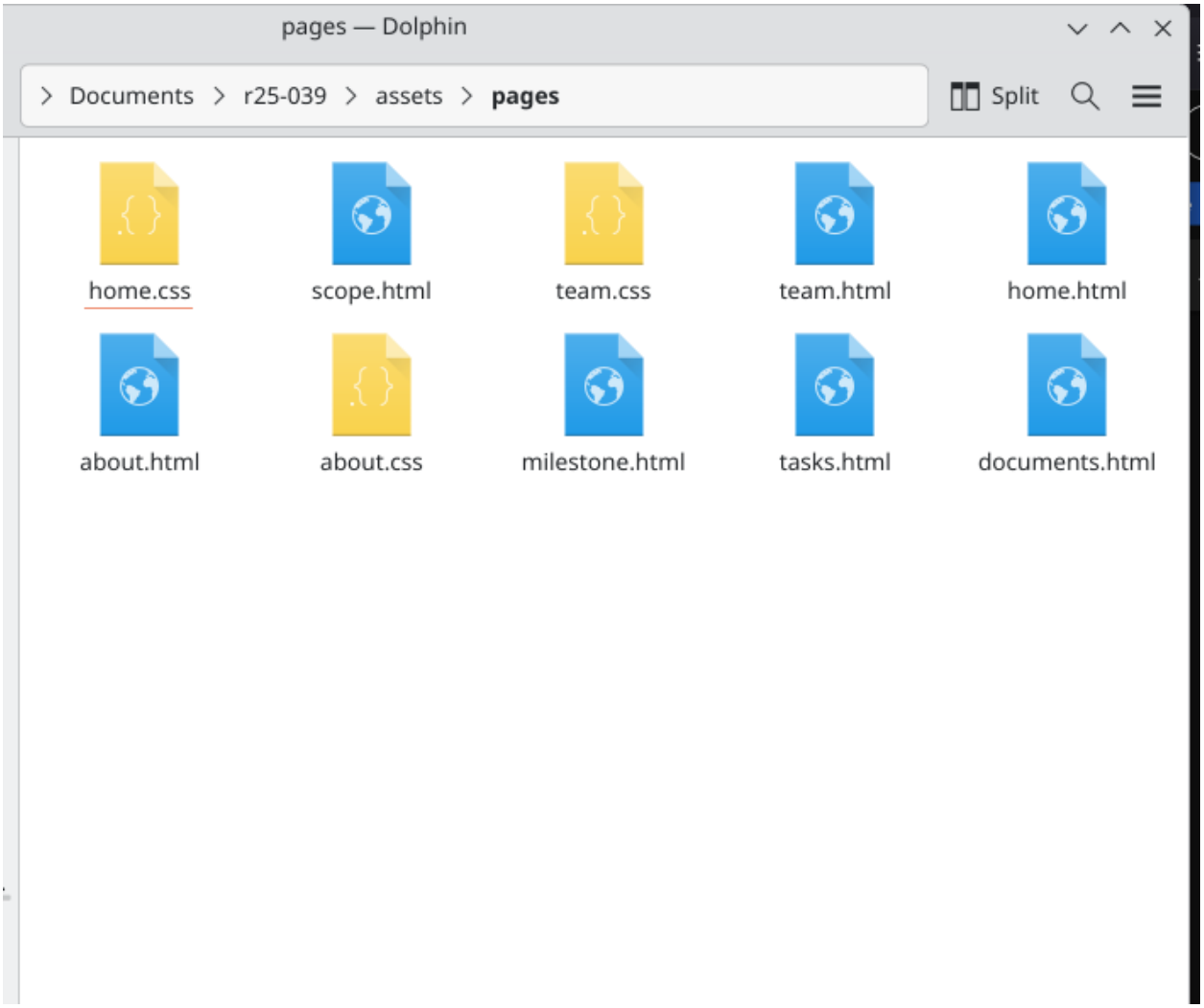
10.Website

10.1Development

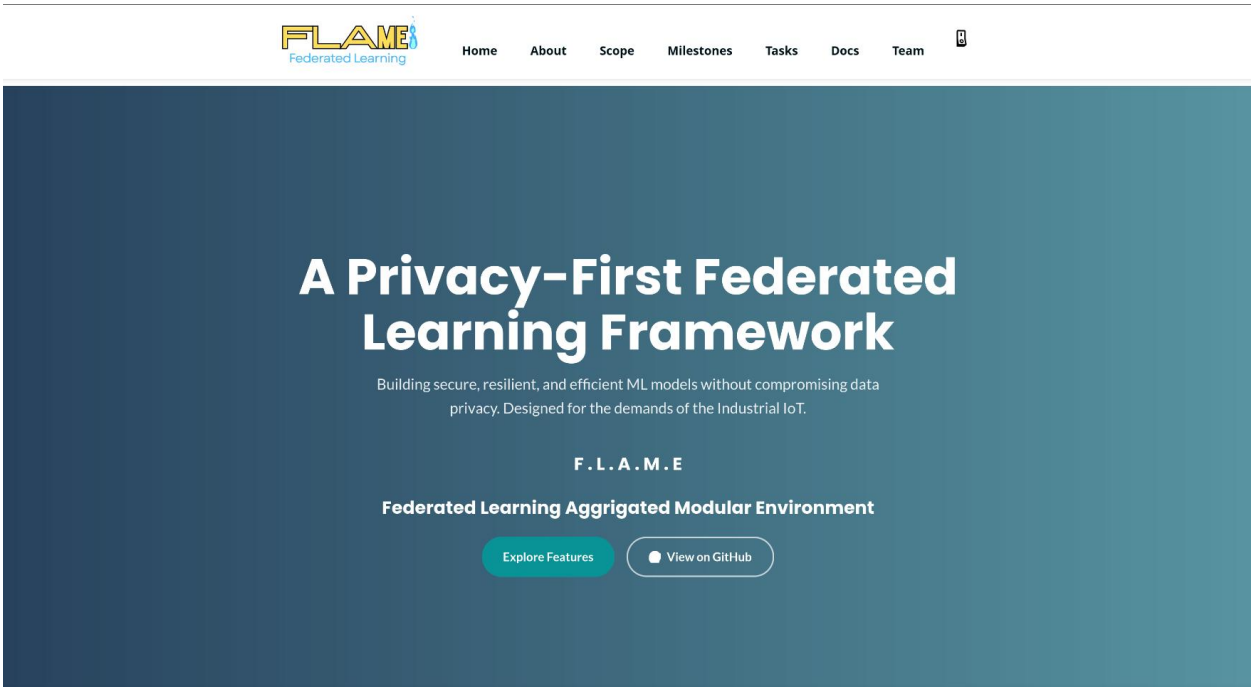



assets — Dolphin
Documents
r25-039
assets
Split
pages
styles
images
documents


documents — Dolphin
Documents
r25-039
assets
documents
Split
PP - Data-Privacy Focused Federated Learning ... IoT.pdf
PPReport - IT21822612 - Privacy P...tion.pdf
PPReport - IT21826368 - Attack Defense ...nce.pdf
PPReport - IT21828348 - Secure Co...trol.pdf
PPReport - IT21831904 - Secure Ag...tion.pdf
Progress_ Presentation 1.pdf
Progress_ Presentation 2.pdf
R25 - 039 - IT4010-charter form - 2025.pdf
R25-039_Final_ Presentation.pdf


pages — Dolphin
Documents
r25-039
assets
pages
Split
home.css
scope.html
team.css
team.html
home.html
about.html
about.css
milestone.html
tasks.html
documents.html

10.2 Finalize



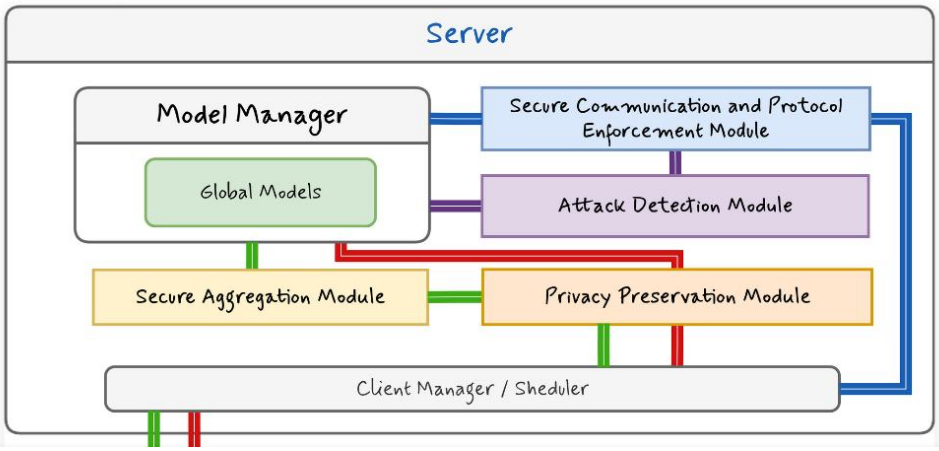
FLAME Federated Learning

Home | About | Scope | Milestones | Tasks | Docs | Team

Privacy-Enhanced Federated Learning Framework

Our comprehensive Federated Learning (FL) System Framework is engineered to significantly augment the privacy, security, and operational resilience of machine learning models deployed in decentralized and distributed environments. The framework is composed of four interconnected core modules, collectively guaranteeing data integrity, defense against adversarial attacks, and authenticated inter-component communication.

Framework Overview and Architecture



Mr. Amila Senerathne

Supervisor

Dr. Sanika Wijesekara

External Supervisor

Mr. T. Kumaralingam

Co-Supervisor