Dashboard Module Specifications

Objective :

- Maintain session data such as session users, session passwords, user and IP mappings, and so on.
- Provide a session manager who will manage all essential session details on both the server and client sides.
- Provide a summarizer that will use suitable filtering or alrgorithm to summarise the meeting.
- Provide a telemetry analyzer that will give data analysis of the meet for further improvement of the software.
- Provide a persistence interface for storing files generated by the telemetry analyzer and summarizer.

Team Configuration:

Team Lead and Persistence:

Handled By: Hrishi Raaj Singh Chauhan (111901054)

Role:

- To design the abstract design for the dashboard module.
- Handle the deadlines for the dashboard module and also make a time schedule for the team members.
- Integrate the module with other modules.
- Handle internal team conflict if arises.
- Regression testing and E2E testing for the module.
- Store the summary for the telemetrical analysis at the end of the meet.
- Create graph for the telemetrical analysis to store it for future reference.

Session Manager:

Handled By: Saurabh Kumar

Role:

- Runs as a controller both on the server and on the client.
- Initialize all managers in other modules with the necessary dependencies.
- Serve as a interface between all of the Dashboard submodules.
- Maintain the session data for all users Set the users list to the Networking module to broadcast
- Also maintain the type of the session whether the session is lab mode or exam mode and notify other modules regarding the change.

Telemetry and UX:

Handled By: Rupesh Kumar

Role:

- Runs statistical anlaysis on each session's dicussions.
- Provid session analytics and give brief summary of the session when it ended like number of chats per session etc.
- Provide the frontend for the dashboard.

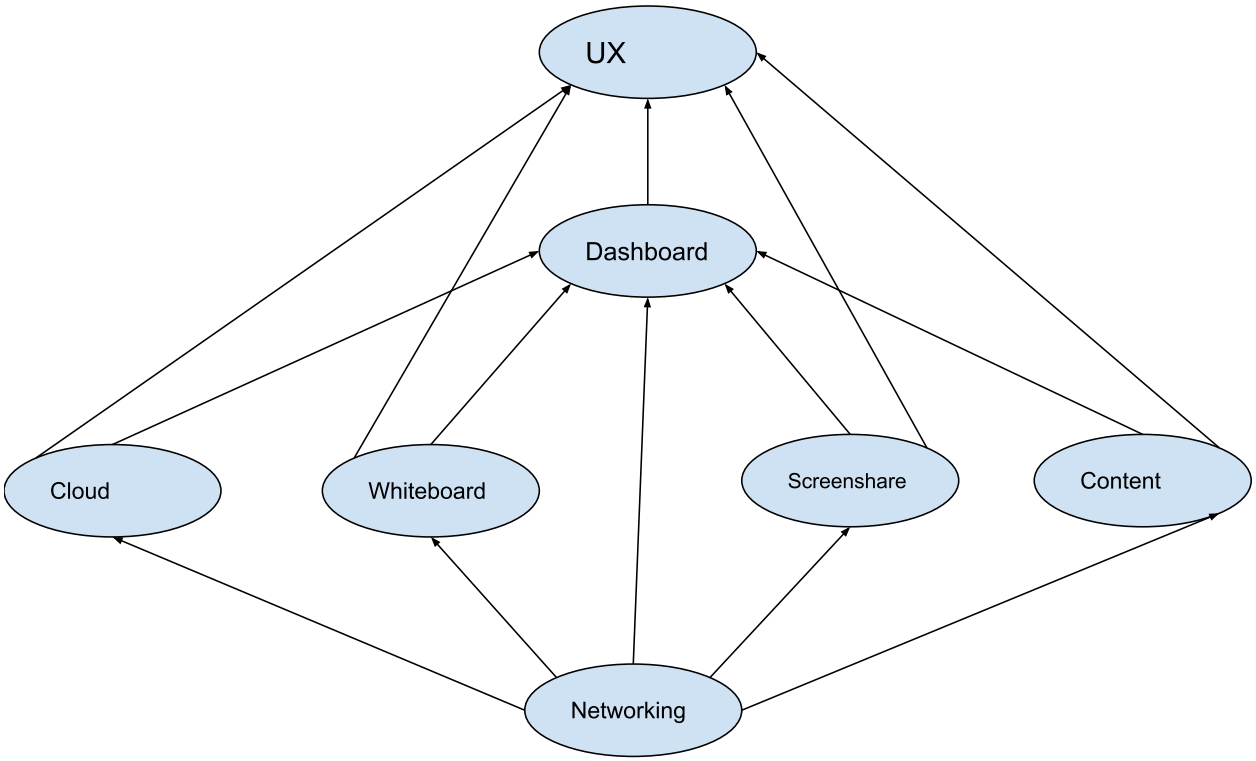
Summary:

Handled By: Morem Jayant Kumar

Role:

- To design an algorithm/model to summarize the content that has been discussed in the discussion and then using persistence module to save it.

Dependeny Diagram:

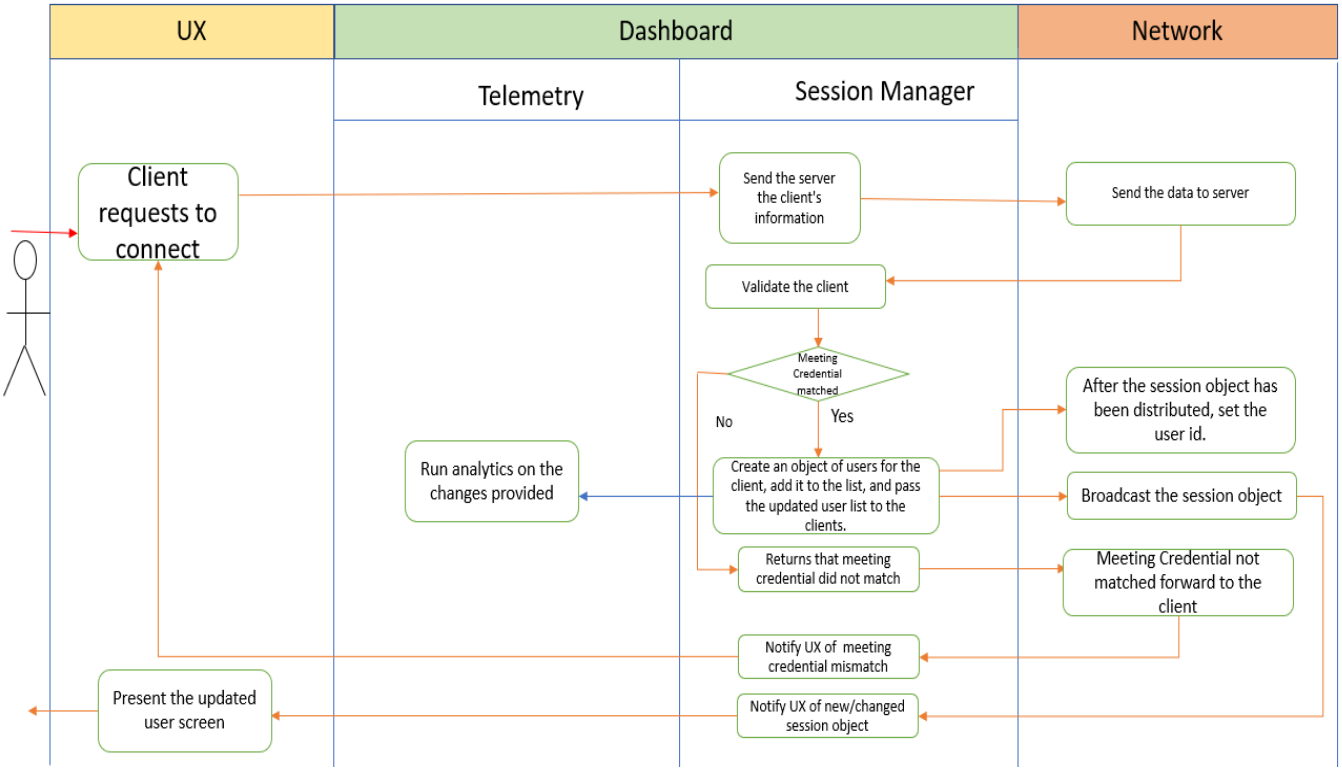


Server Boots up:

The session manager initialises the session object on the server-side when the server is started, and the session manager also regulates the flow of IP address and port to connect to the server.

Client Joins the meet:

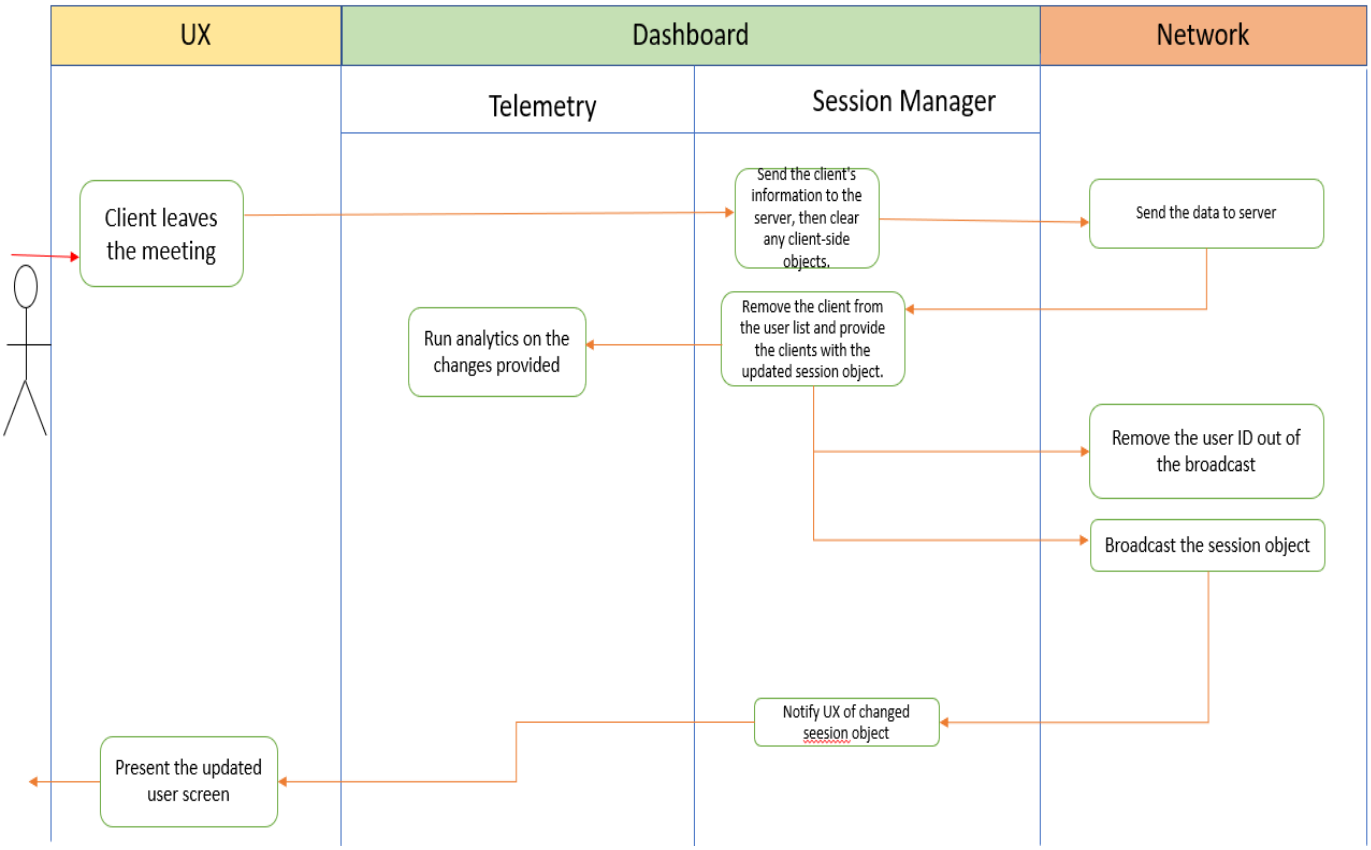
When the client joins the meet the client side dashboard will receive the AddClient call from the UX which it will transfer to the server side dashboard module using network module. The server side then creates a user object based on the cient details which is received from the UX and then add it to the session object. The telemetry module will run its analysis on the changed data.This change in the session object is then broadcast to all the other modules. The UX will also get notified about the change.



Client Leaves the meet:

When the client leaves the meet the client side dashboard will receive the RemoveClient call from the UX which it will transfer to the server side dashboard module using network module.

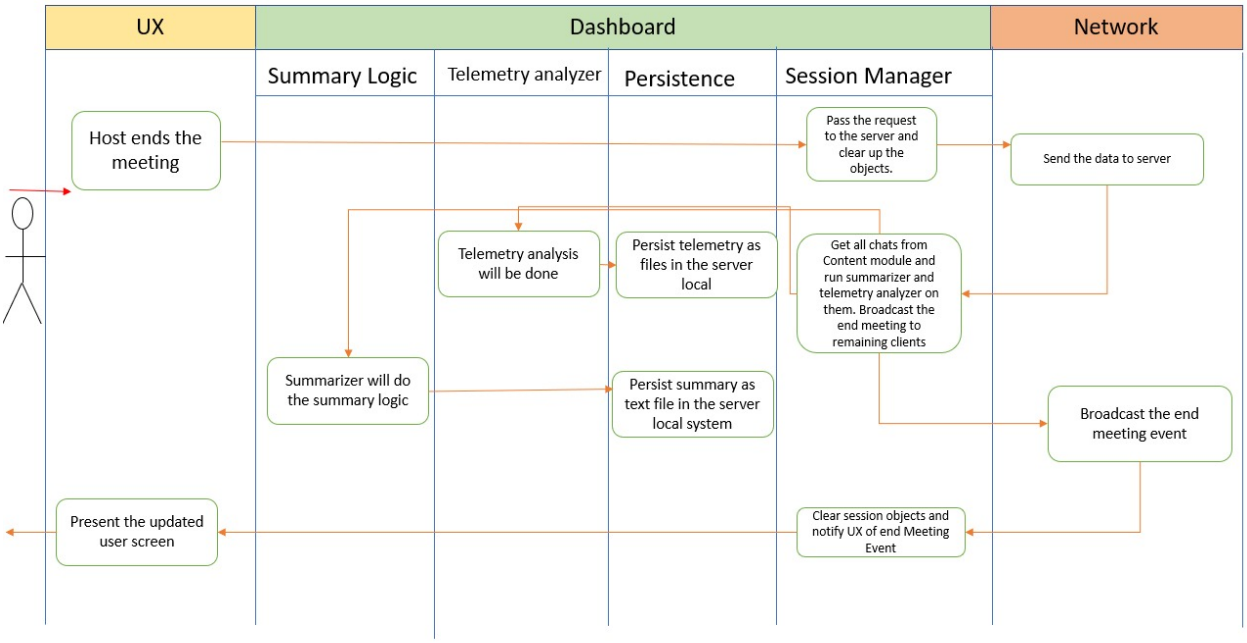
The server side then removes the user from the session object. The telemetry module will run its analysis on the changed data.This change in the session object is then broadcast to all the other modules. The UX will also get notified about the change.



When Host Ends the Meeting:

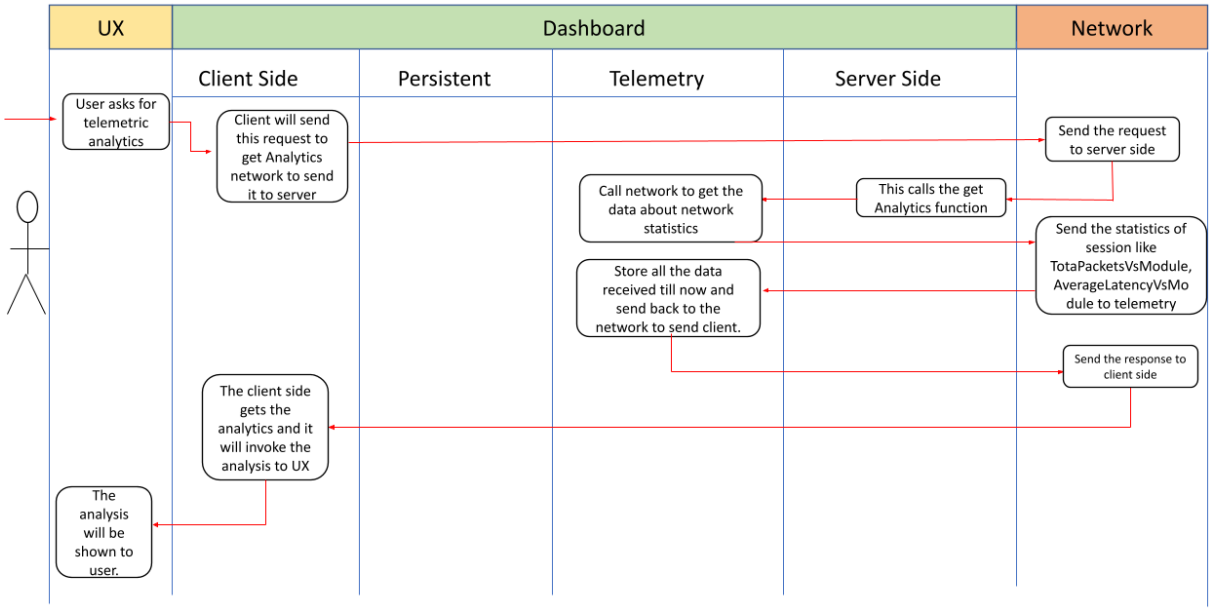
When host ends the meet then session manager should clear up the session object at the client side. Then the session manager should gather all the chats on which summary logic will be

applied to make a summary of the discussion which is then passed to the persistence module to save it as a text file. Also session manager will provide chat data to the telemetry module which will complete its analysis and trigger the persistence module to save its analysis in an understandable format.



Telemetry Analytics Retrieval During The Session:

During the session if user asked for the telemetrical analysis during a ongoing session ,Then the dashboard will transmit the request from the client to the server, where it will already have the data that it would have received from the other modules to create analytics for the client who requested it. This data will be provided then to the client side to display.



Persistence:

Handled By: Hrishi Raaj Singh Chauhan (111901054)

Overview:

Persistence module is an important module of the dashboard whose functionality is to save the analytics and graphs related to the analytical summary of each session which will be provided by the telemetry module when the meeting ends. Also this module will be used to save the chat summary of the session. Basically this module will interact with two othe modules namely summary and telemetry, these modules will call the persistence module for the save functionality.

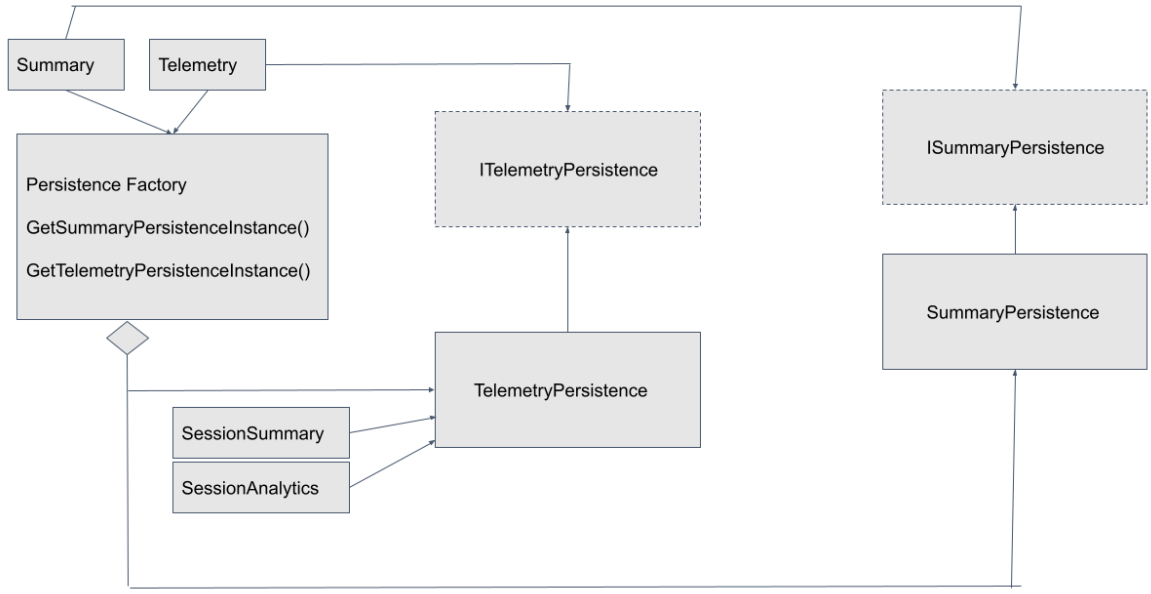
Objective:

- To store the analytical graphs in a .png format
- To store a document of the session analytics in a understandable format.
- Stores the chat summary in the .txt format at the end of the meeting.

Design Analysis:

- The module will follow factory design pattern to create instance of the objects in a go. Basically the persistence factory will provide two persistence instance creator function like GetSummary and Gettelemetry. These will be exposed to other modules as well.
- To create graphs like histogram etc we will get UserCountVsTimeStamp ,UserIdVsChatCount,TotalPacketsVsModule,LatencyVsModule from the telemetry module.
- We will also get the summary of telemetrical analysis in a string format maybe and we will store them in a file at the end of the session using a save function.
- There should be several save methods to store distinct files, like as when storing the summaryChat, which requires different inputs than when storing the graphs.
- To serve this purpose, it is better to create two different Interfaces like ISummaryPersistence and ITelemetryPersistence. \

Class Diagram:



Factory:

We are using factory pattern and singleton design pattern for Persistence.

```
public static class PersistenceFactory
{
    private static TelemetryPersistence telemetryPersistence = null;
    private static SummaryPersistence summaryPersistence = null;

    public static SummaryPersistence GetSummaryPersistenceInstance()
    {
        if(summaryPersistence == null)
        {
            summaryPersistence = new SummaryPersistence();
        }
        return summaryPersistence;
    }
    public static TelemetryPersistence GetTelemetryPersistenceInstance()
    {
        if(telemetryPersistence == null)
        {
            telemetryPersistence = new TelemetryPersistence();
        }
        return telemetryPersistence
    }
}
```

Interfaces:

ITelemetryPersistence:
This interface is required by the telemetry so that they can call Save function to store Analytics data represented as graph in .png format and using SaveSessionSummary to save the summary file of the analytics.

```
using Dashboard.Server.Telemetry;

namespace Dashboard.Server.Persistence
{
    public interface ITelemetryPersistence
    {
        public void Save(SessionAnalytics sessionAnalyticsData);

        public void SaveSessionSummary(SessionSummary sessionsummaryData);
    }
}
```

ISummaryPersistence:
This interface is required by the Summary module which will call SaveSummary function to store the summary in .txt format at the server.

```
namespace Dashboard.Server.Persistence
{
    public interface ISummaryPersistence
    {
        public bool SaveSummary(string message);
    }
}
```