Meet.me

Design Specifications

CS5617 Software Engineering

By Yogesh R (111801047)

Technical Architect

Instructor: Ramaswamy Krishnan Chittur

Table of Contents

1 Overview

The recent pandemic has changed the ways of working collaboratively: it has moved to online services such as video conferencing. We aim to facilitate the collaboration experience with our application: meet.me.

Meet.me has the features of chats, a whiteboard, screen-sharing, summary generation and telemetry. The whiteboard is a canvas with tools such as pen, brush, shapes, eraser, and features such as undo and redo. The summary generation uses the chats from the room to provide a succinct summary. The telemetry provides statistical plots & information.

1.1 Application Obligations

1. Goals
    a. A client-server architecture-based application.
    b. The application runs on a local area network using sockets.
    c. Users can send global chats/files, and screen share. The chats/files will have the features of promote and reply.
    d. The whiteboard supports standard shapes such as circles, rectangles, etc., on the whiteboard. It has features like select, delete, save, undo/redo, and check pointing.
    e. It has basic password-based authentication.
    f. It provides summary, telemetry and snapshots taken by users.
2. Extended Goals
    a. The application runs over the internet using web sockets.
    b. The application allows freestyle drawing.
    c. Restoring information about archived meetings.
    d. Emoji, private text, chat threading, and file previewer
    e. To use google OAuth-based authentication.
    f. Add the application to the MS store.

1.2 Organizing Modules

The organizing modules are to perform specific functionally of meet.me. We have five core modules: networking, dashboard, content, whiteboard & screen share, and UX. Each core module will communicate with a testing module to validate the same.

1.3 Key architecture

Meet.me works based on client-server architecture [1]. It runs on a local area network. The server holds the state of the room, and all the clients connect to it. The connections happen via sockets in the “IP: Port” of the server. The host of the meeting boots the server, and the members use the client program to join the discussion.

It will be made using C# programming language, .NET framework v4.8, and visual studio 2019. Version control is done using GitHub, and project tracking is done using azure DevOps.

2 Modular Structure

Meet.me has five core modules: Networking, Dashboard, Content, Screen Share & Whiteboard, and UX.

2.1 Networking

The networking module deals with the communication of the client and the server. It provides functions such as sending different objects, serialization & deserialization of objects, and prioritizing real-time whiteboard and screen-share over chats/files. The networking module provides one function in client side to send to server, and two functions send and broadcast in server side to send to clients. The serializing is done by each class itself. This module will have a map from IP to streams. The removal of data from this map is done by the dashboard via a function defined in the networking module. Refer to networking spec.pdf for detailed information.

2.2 Content

The content module deals with all the operations pertaining to chats and files. The chat operations are global-broadcasting, private messaging, prioritizing chats, emoticons support, rich text support, replying, user-tagging, etc.

The state management in content will handle the storage and broadcasting of chats/files. The content module will store all the chats in both client and server. The files will be stored in server and a link to download will be send to all clients. Refer to content spec.pdf for detailed information.

2.3 Screen Share & Whiteboard

The screen share & whiteboard module implements the screen-sharing and whiteboard logic. The screen-sharing will be consecutive images send to the server for broadcast with optimizations such as discarding images similar to the previous image.

The whiteboard will be a list of Shape objects such as circles, rectangles, etc. The whiteboard will have a toolbox with standard shape, eraser, brush, pen, etc. It will also support features such as undo/redo and checkpointing. The user can have unlimited number of checkpoints (constrained by server storage).

The state management in screen share & whiteboard module will handle the storage and broadcasting of shapes/images. The whiteboard state will be stored in both client and server. At any given time, only one user can share the screen. Refer to screen share & whiteboard spec.pdf for detailed information.

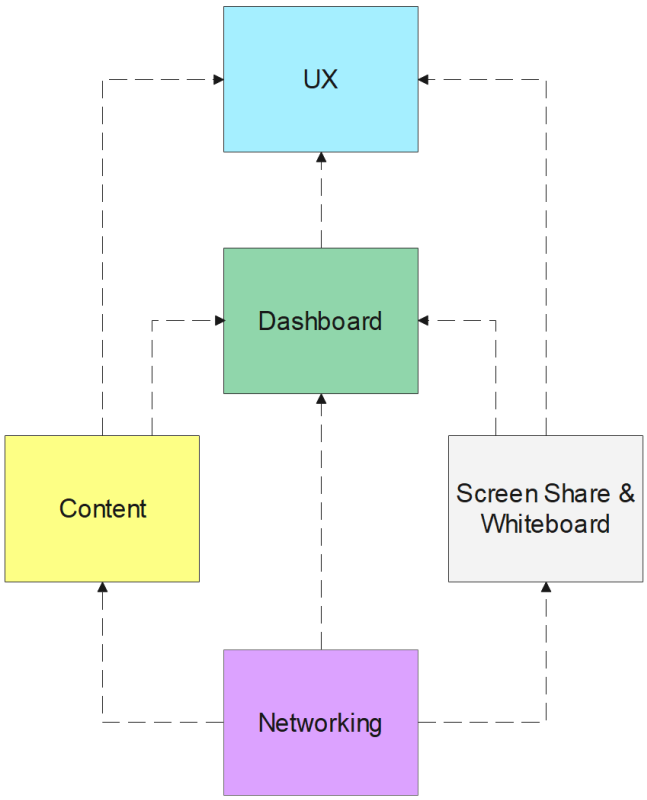
2.4 Dashboard

The dashboard module handles the session and other required processing pertaining to it. The client and server sessions store all the users in the room. The server session additionally stores the IP addresses and password of the room. It also provides additional features such as telemetry and summary. The summary will be computed and send to users who request it during the meeting. The summary & telemetry is computed and persisted in the end of the meeting. Refer to dashboard spec.pdf for detailed information.

2.5 UX

The UX module provides the façade to meet.me. It deals with the aesthetic appearance and the intuitive user experience. We will be using the model-view-ViewModel software architecture pattern as it facilitates the separation of the GUI code from the development of business logic, thus making it easier for unit testing. There will be three pages: Dashboard, Whiteboard, and Screen-sharing. The chat window will overlay from the right side in all the pages. Refer to UX spec.pdf for detailed information.

2.6 Dependency Diagram



The UX module renders the UI based on the data from dashboard, content, screen share and whiteboard. The dashboard provides the users and summary. The content provides chats and files. The dashboard depends on content, screen share and whiteboard for summary generation and persistence. The dashboard, content, screen

share and whiteboard depend on the networking to communicate between client or server.

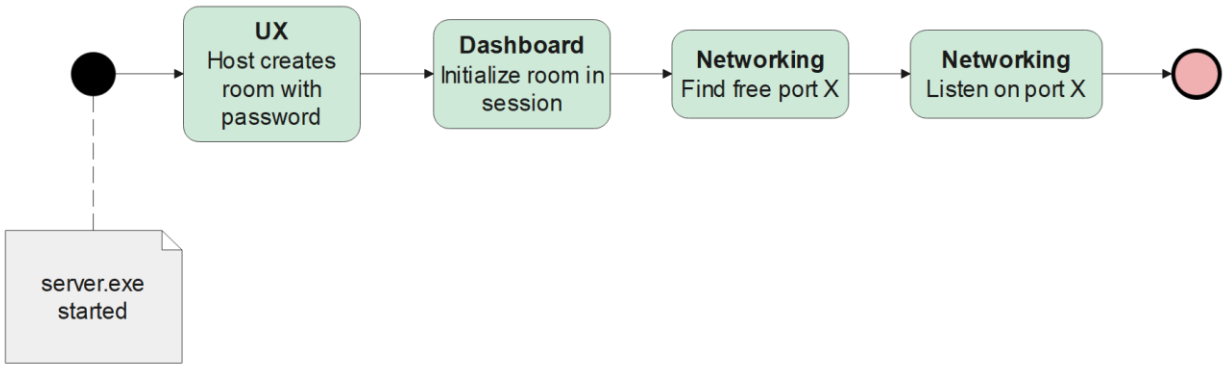
The testing module will be dependent on all the modules for unit testing.

3 Activity Diagrams

The blue and green boxes imply client and server, respectively.

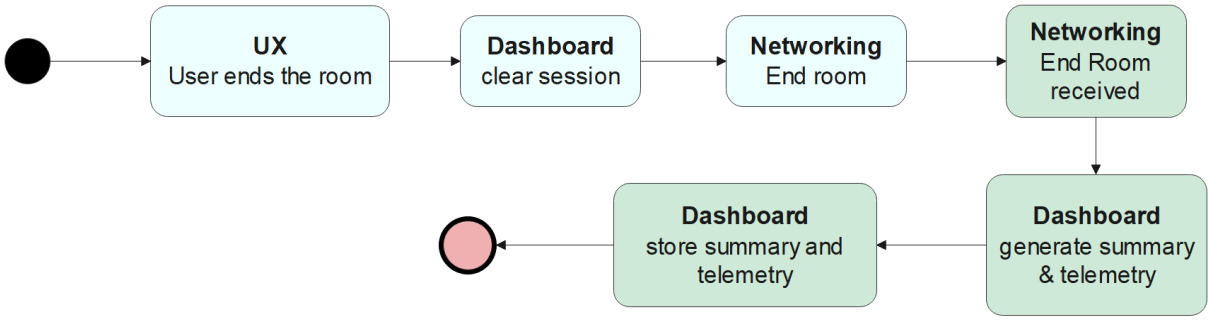
3.1 Room Activities

3.1.1 Room Creation



The host runs server.exe, which brings up a CLI. The host creates a room with a password. The CLI returns the IP and port X at which it is listening. An empty folder with current data time is created to persist data pertaining to the room. Now the host has to share their IP:X and password with all the members of the meeting.

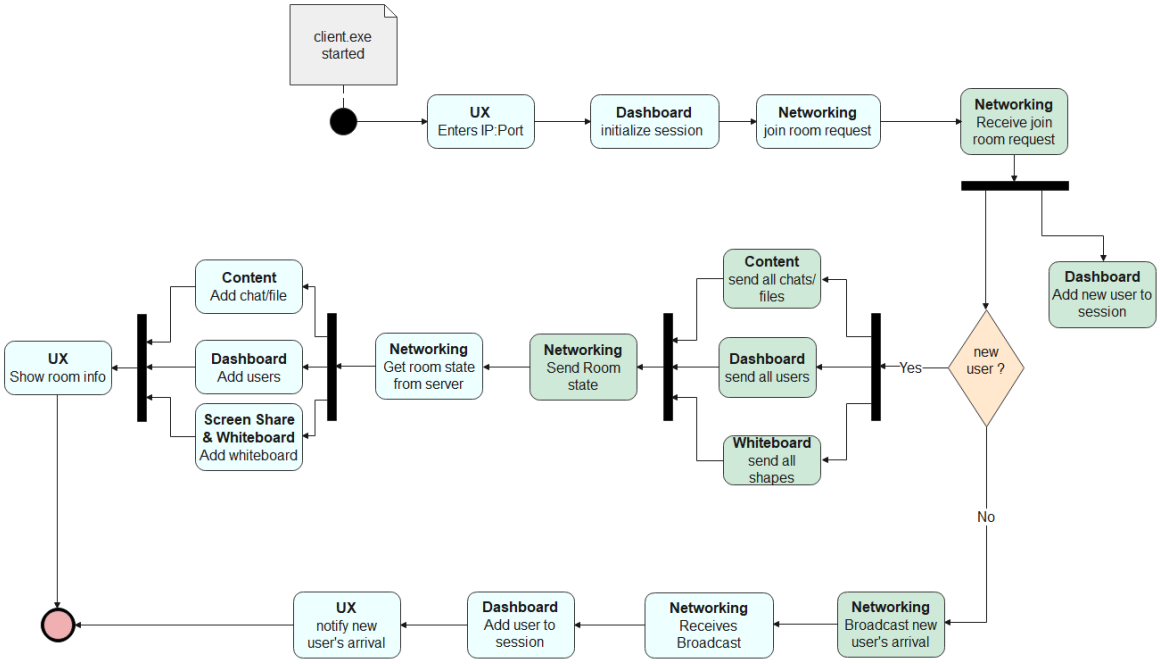
3.1.2 Room Completion



When the room is completed, the summary & telemetry is computed and persisted. The summary will be a text or pdf and telemetry will be some images showing statistical plots.

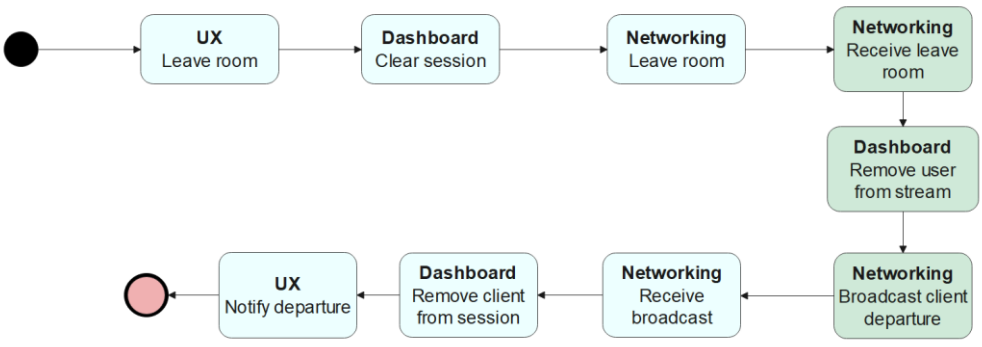
3.2 Client Activities

3.2.1 Client Arrival



The meeting member will run the client.exe that will bring a user interface with a join room button. Clicking on the button will prompt the user to enter the IP: Port given by the host and a username (to display others). Then the user will receive the room state (all chats/files, shapes, users) from the server.

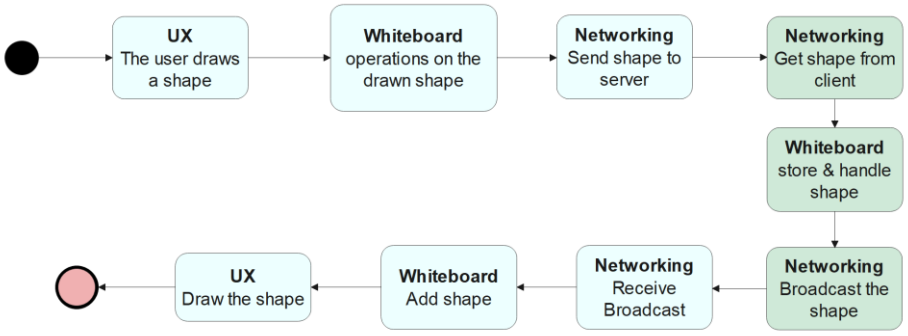
3.2.2 Client Departure



When a user leaves the meeting, the session removes the user from the server, and send a broadcast. All the other clients' session will also remove this user. When the last user leaves the room, it is considered as room completion.

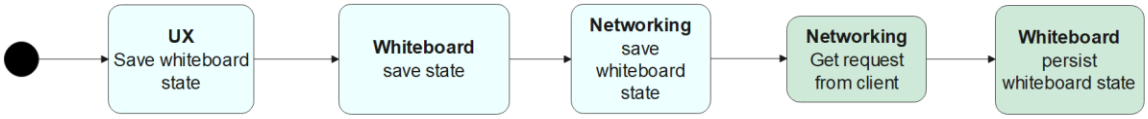
3.3 Whiteboard Activities

3.3.1 Draw



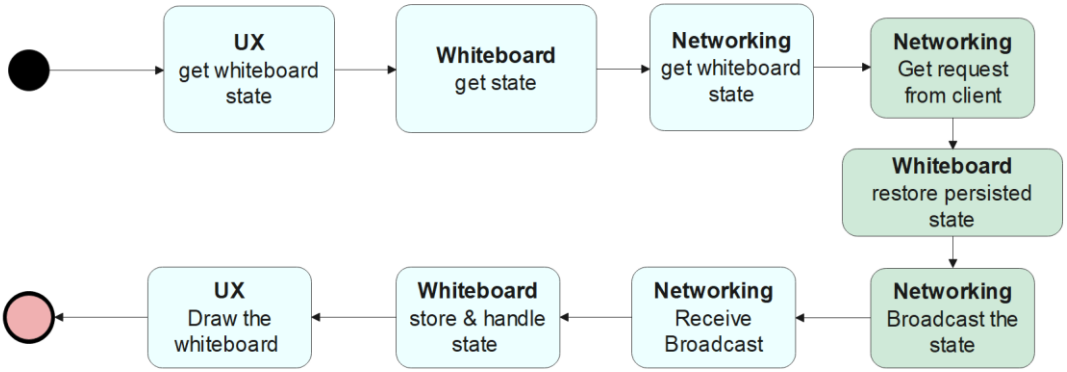
When a user draws a shape, the newly drawn shape is broadcasted to all the clients and also saved in the server. All the clients receive the broadcast and update theirs chats.

3.3.2 Save Whiteboard



The whiteboard is saved as a list of Shape objects. The current state is saved in the server for future retrieval. The user can store any number of states.

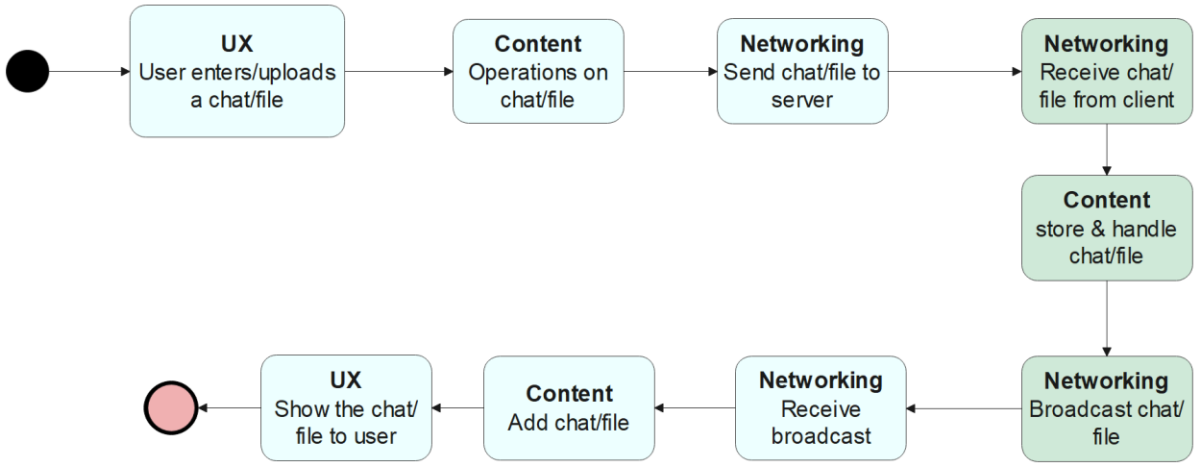
3.3.3 Restore Whiteboard



The stored whiteboard can be retrieved by the user. The UI will have a dropdown where the user can choose their checkpoint to retrieve from.

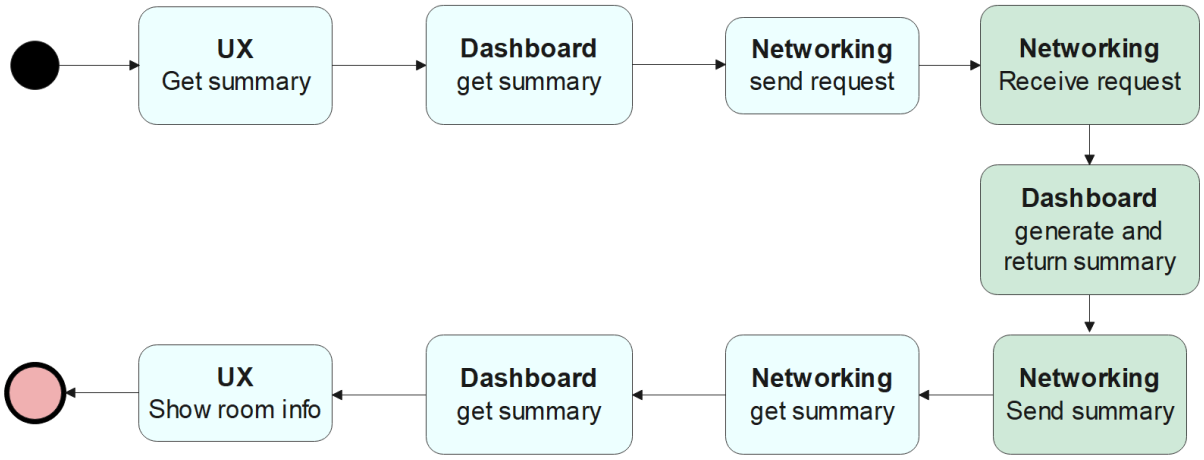
3.4 Other Activities

3.4.1 Chat/File



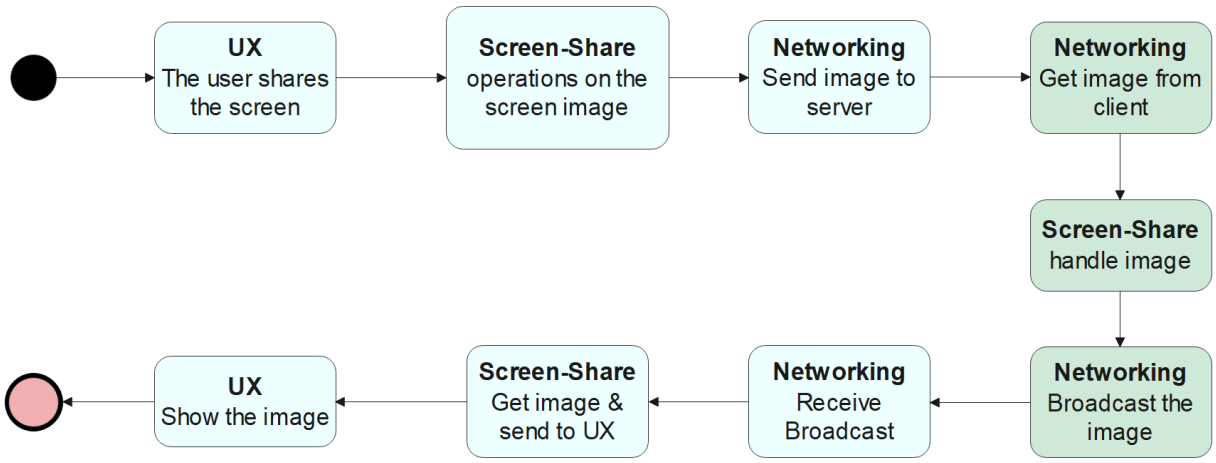
The client can add a message or upload a file in the chat panel. It will be sent to all other clients in the room.

3.4.2 Summary Acquisition



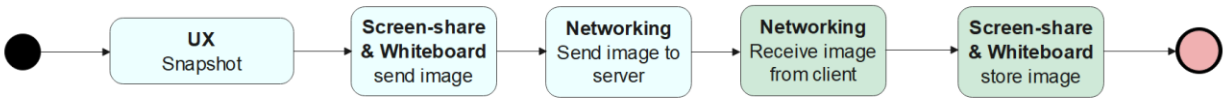
The client can request the summary during the meeting. The summary is generated in the server and return to the user. The user can see it in the summary panel of dashboard page in the UX.

3.4.3 Screen Share



Screen sharing is done by sending multiple images of the client's screen. The newly captured image is broadcasted to all the clients. A null image object is sent to indicate end of screen sharing.

3.4.4 Snapshot



The client can take a snapshot of the whiteboard or the screen-share at any moment. The image taken is stored in the server.

4 Conclusion

Meet.me can be used by teams or friends to brainstorm an idea or discuss any topic in general. It provides features like chat, file sharing, whiteboard, and screen-sharing, thus making communication among teams easier.

The aesthetically pleasing UI with simple, intuitive user experience and robust server makes meet.me a superior choice for teams.

5 Appendix

1. Why client-server architecture over peer-to-peer architecture?

   We realized that peer-to-peer architecture implementation has higher programming complexity than client-server. So, with a notable time constraint of one semester, we decided to use client-server architecture. ref.

2. Why are files used for persistence over database?

   Since the server is running on a host machine, who is an end-user we can't expect them to setup our database with required migrations, ports, drivers, passwords, etc. Moreover, using ubiquitous files such as txt will be easily accessible by all types of users.