60004210159
Dhruvin Chawda
C22

SE Exp 3

Aim: Identify scenarios and develop UML Use case and Class Diagram for project (online meeting software).

Theory:

Use Case Diagram:

→ Actors and their roles:
① Host: User who schedules and manages online meeting
② Participant/Attendee: User who attends the meeting
③ Admin: Manages user accounts, system configuration, makes report.

→ Use Cases:

- User Management (Admin)
→ Create/Edit/delete User Account
→ Manage User Roles
- Meeting Management (Host User)
→ Schedule/edit/cancel/Manage Meetings
→ Start/end meeting
→ Record meeting
- Meeting Participation (Attendee User)
→ Join/leave meeting
→ Use Audio/Video chat
→ Turn on/off camera/microphone
→ share screen/files

Class Diagram:

Classes in class Diagram:

① User: Represent system user which can have role (Host, attendee, admin). This class captures user account and their access level

② Meeting: Represent scheduled online meeting and have operations scheduling, editing, cancelling, starting, ending meeting. This class is central to system managing meeting, participants.

③ Participant: This class inherits from User. This class focuses on user behaviour with specific meeting context.

④ Recording: This class stores info and data of Recording.

⑤ Communication, content, External Application are generalized classes of Meeting.

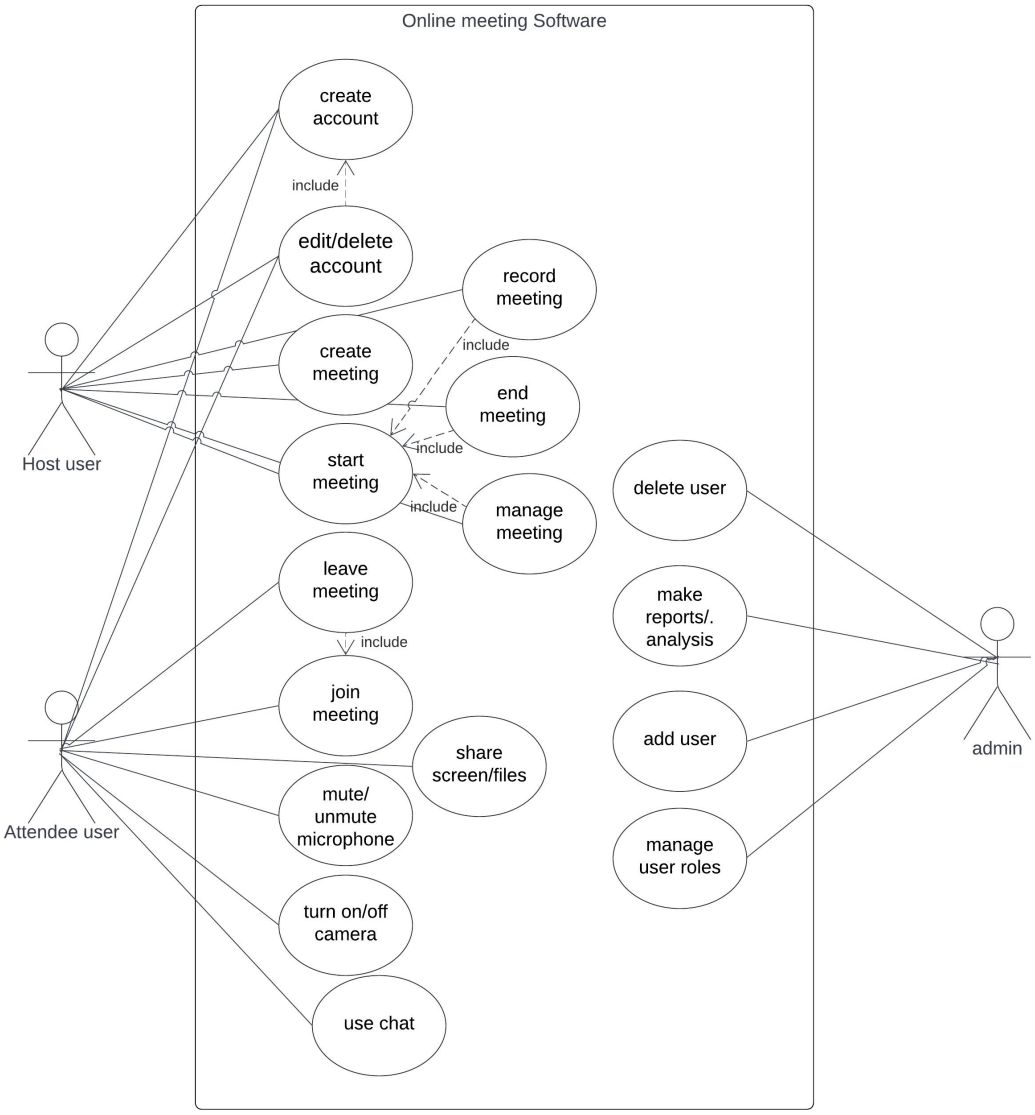
⑥ Communication further have child classes Audiochat, Textchat Videochat class and have functionality like starting/stopping chat, muting/unmuting participant, managing chat history.

⑦ Content also have child classes has screenshare, Document having function as sharing screen with other attendee, Document enables document collaboration.

Conclusion: Hence we studied what are UML diagram and made Use Case and Class Diagram for Online meeting software. Also studied why these Diagrams are useful.

Implementation :

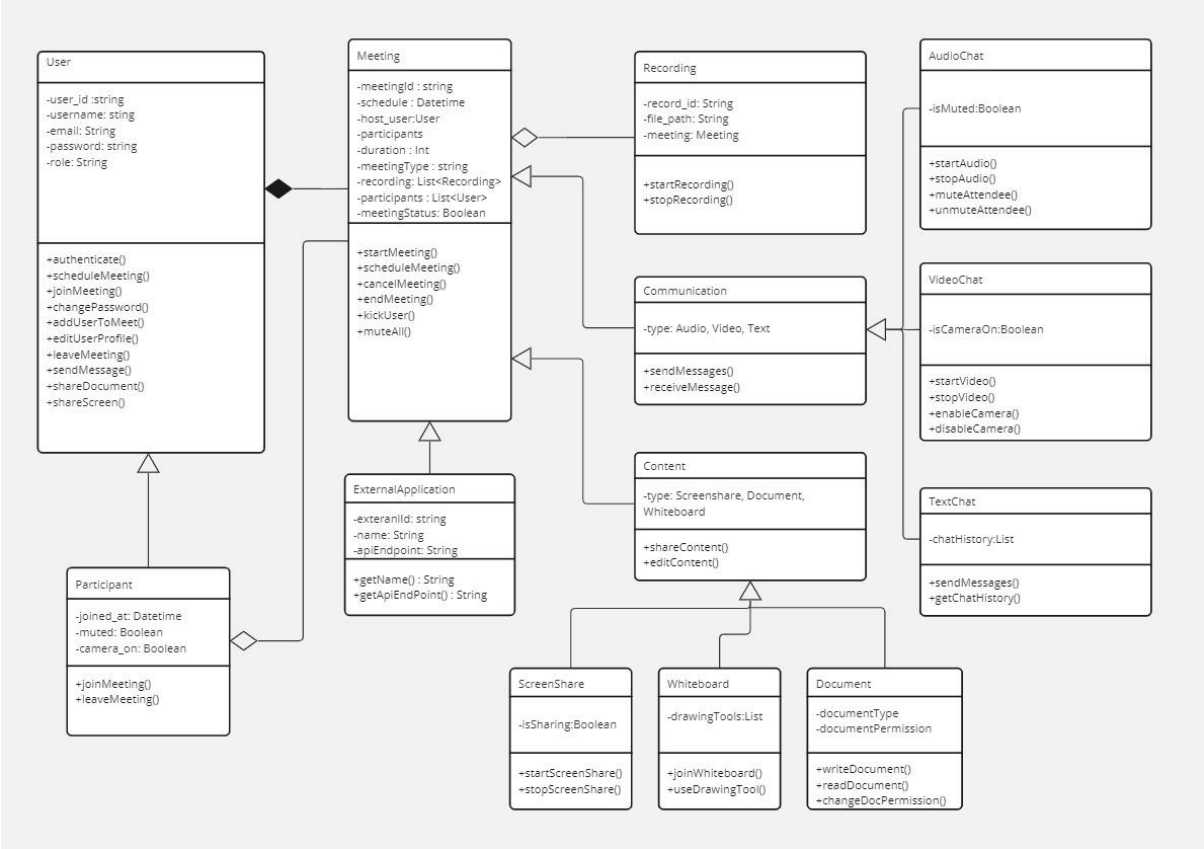
1. <u>For Use case diagram :</u>



The use case diagram for an online meeting software application depicts the interactions between various actors and the system itself. The actors include the Host User, responsible for scheduling meetings, inviting attendees, managing the meeting flow, and potentially recording and analyzing sessions. Attendees, on the other hand, focus on joining meetings, participating through chat and audio/video functionalities, and staying updated on meeting information. An optional Admin User might exist to manage user accounts, roles, and generate reports on meeting usage. The system facilitates these interactions by providing functionalities for creating and scheduling meetings, managing invitations and attendees, enabling communication features like

chat and audio/video, and offering optional recording and analysis tools. Actors interact with the system to acquire information like meeting details, shared content, and communication from others. They might also need to inform the system about changes like scheduling conflicts, connectivity issues, or user account modifications. The system keeps actors informed about meeting status, unexpected changes, and potentially generates reports for further analysis. This interplay between actors and the online meeting software application allows for effective communication and collaboration in a virtual setting.

2. <u>For Class diagram :</u>



Analyzing the class diagram of the online meeting software application reveals several key entities involved in its operation.

External entities include Users (participants) and potentially External Applications that integrate with the meeting software. Internal to the system, various data elements exist as things. These encompass Meetings themselves, Recordings of meetings, Communication (chat messages), and Content that can be shared during meetings (screen shares, documents, whiteboards). Actors within the system interact with these

elements based on their roles. The primary role identified here is the Participant (meeting attendee).

The core functionality revolves around the Meeting class. It has attributes like meeting ID, schedule, host user, list of participants, duration, type, and status. Recordings are linked to meetings and store the file path. Communication during meetings is captured by the Communication class, which has subclasses for AudioChat, VideoChat, and TextChat. Each subclass might have attributes specific to its medium. The Participant class represents a user within a meeting and tracks their join time, mute status, and camera status.

Structure classes define functionalities used during meetings. These include functionalities like AudioChat, VideoChat, TextChat, ScreenShare, Whiteboard, and Document. Some structure class attributes might be redundant with Participant attributes (e.g., isMuted in AudioChat vs. muted in Participant). The TextChat class manages the chat history. The Content class acts as a base class for shared content during meetings, with subclasses like ScreenShare, Whiteboard, and Document. Each subclass might have its own attributes like sharing status for ScreenShare or document details for Document.

Operations on these classes define the system's behavior. Users can authenticate, change passwords, and edit profiles. Meetings can be scheduled, started, ended, canceled, and managed through functionalities like kicking users, muting all, and adding users. Recordings can be started and stopped. Participants can join and leave meetings, potentially with methods to retrieve their names (which might be redundant depending on how usernames are accessed). Communication functionalities like sending and receiving messages are implemented within the Communication class and its subclasses. TextChat specifically offers the ability to retrieve chat history. Content can be shared and edited through its subclasses. Screen sharing can be started and stopped, while Whiteboards allow users to join and use drawing tools. Documents can be written to, read from, and have their permissions changed.