Middle East Technical University
Department of Computer Engineering

Initial Design
for
“CL@SS++”

profIT

1250125 - İsmail ÇETİN
1250349 - Yavuz GÜRCAN
1250356 - Recep GÜRLEK
1250661 - Hakan ÖZTÜRK

Fall 2004-2005

Table of Contents

1 Introduction 4
1.1 Purpose 4
1.2 Scope 4
1.3 Major Design Constraints 4
2 Architectural Design and Decomposition Description 5
2.1 Overall System architecture 5
2.2 Presenter Client 6
2.2.1 Main Module 7
2.2.2 Connection Module 7
2.2.3 Presentation Module 7
2.2.4 Application Sharing Module 8
2.2.5 Whiteboard Module 9
2.2.6 Chat Module 11
2.2.7 File Transfer Module 12
2.2.8 Video Handling Module 12
2.2.9 Audio Handling Module 14
2.3 Server 15
2.3.1 User Connection Handler Module 15
2.3.2 Virtual Class Data Module 17
2.3.3 Synchronization and Broadcast Module 17
2.3.4 Database Connection and Storage Unit of Virtual Class 19
2.4 Participant 20
2.4.1 Main Module 20
2.4.2 Connection Module 21
2.4.3 Presentation Module 21
2.4.4 Application Sharing Module 22
2.4.5 Whiteboard Module 23
2.4.6 Chat Module 24
2.4.7 File Transfer Module 25
2.4.8 Audio/Video Handling Module 26
2.5 Data Decomposition 27
2.5.1 Data Dictionary 27
2.5.2 Database 29
2.5.2.1 ER Diagram 29
2.5.2.2 Table Descriptions 30
2.6 Graphical User Interface Design 34
3 Versioning 36
4 Project Schedule 38
5 Conclusion 40

Index of Figures

Figure 1 Overall View of the System 5
Figure 2 Presenter Module Decomposition 6
Figure 3 CL@SS++ Main Module 7
Figure 4 Connection Module 7
Figure 5 Presentation Module 8
Figure 6 Application Sharing Module 8
Figure 7 Collaboration Diagram for Presenter Application Sharing 9
Figure 8 Whiteboard Module 10
Figure 9 Collaboration Diagram for Presenter Whiteboard 11
Figure 10 Chat Module 11
Figure 11 File Transfer Module 12
Figure 12 Collaboration Diagram for presenter File Transfer 12
Figure 13 Video Module 13
Figure 14 Collaboration of Broadcast Video Presenter Side 14
Figure 15 Audio Handling Module 14
Figure 16 Collaboration of Broadcast Audio Presenter Side 15
Figure 17 Class Diagram of Connection Handler Module 16
Figure 18 Collaboration Diagram for Connection Handler Module 16
Figure 19 Class Diagram of Virtual Class Module 17
Figure 20 Class Diagram of Synchronization and Broadcast Module 18
Figure 21 Collaboration Diagram for Synchronization and Broadcast 19
Figure 22 Class Diagram for Database Connection and File Storage Unit 20
Figure 23 Collaboration Diagram for Database Connection and File Storage Unit 20
Figure 24 CL@SS++ Presenter Main Module 21
Figure 25 Connection Module 21
Figure 26 Presentation Module 21
Figure 27 Application Sharing Module 22
Figure 28 Collaboration Diagram for Participant Application Sharing 23
Figure 29 Whiteboard Module 23
Figure 30 Collaboration Diagram for Participant Whiteboard 24
Figure 31 Chat Module 24
Figure 32 Collaboration Diagram for Participant Chat Module 25
Figure 33 File Transfer Module 25
Figure 34 Collaboration Diagram for Participant File Transfer 26
Figure 35 Audio/Video Handling Module 26
Figure 36 Audio/Video Handling Collaboration Diagram 26
Figure 37 Entity Relationship Diagram 29
Figure 38 CL@SS++ Graphical User Interface 35

1 Introduction

This document structure is created taking into consideration the "IEEE Recommended Practice for Software Design Descriptions" document, but is restructured in relation to similar design templates and project content delivery needs.

1.1 Purpose

This document describes and gives an overall description of the CL@SS++ project after the initial design phase. During this phase the project team analyzed several software development platforms and their libraries which would be helpful for the progress of the project.

Although it is an initial design report the architecture is developed as much as possible since this expansion will be helpful for the progress of the design. In general, the purpose of the design is to satisfy student and teacher needs and supply them with a user-friendly environment.

1.2 Scope

The CL@SS++ project will mainly focus on creating a synchronous communication environment for students and teachers over the LAN. The participants will be supplied with features to facilitate learning and teaching on the internet. Since some general features are already provided by almost all of the existing systems, it is important to add new functionalities. In this respect the project will first deploy a base application to fulfill the requirements and then enhance it by supporting application sharing and facilitating the presentation of a specific application.

The project has two target audiences. One is the presenter (instructors) and the other is the participant (students). The presenter will be facilitated with giving presentations, sharing his/her applications, giving pop quizzes and also using whiteboard technologies. On the other hand, the system lets the students to watch the lecture (offline or online), upload and download his/her files, and chat with his/her friends.

1.3 Major Design Constraints

In the design stage of the project we have decided on some constraints that will enable us to achieve some of our tasks. These constraints will both enable us both with good quality software design and also limit some our requirements since they have limiting effects on other more important tasks.

The first of the constraints is the network structure that we use. We decided to use LAN rather than WWW since using WWW will bring many other tasks which are difficult to achieve. Also they will cause the consumption of time as we will be working on things out of our main scope. By using LAN the things will be easier that there will be limited number of clients. Also the communication, data transfer speed and synchronization of multimedia stream will be easier.

Although using LAN will ease our job in data transfer again we have to keep the size of the data to be transferred as small as possible, because we have to transfer the data simultaneously and the rate of the transfer fully depends on the size of the data. In order to achieve this goal, we will compress our video frames, desktop application images and whiteboard pictures before sending them. Also we will try to minimize the transfer rate of them (e.g. frames/second) in order to transfer data correctly and simultaneously. We will not send the images of the shared applications and the whiteboard as long as there is no change on them.

Another design constraint is that we will use our own application and UI instead of using a Web Browser. Our task will need the transfer of the real time data which will make our applications dynamic. As the dynamic applications have many problems in Browser and the Web languages such as HTML are inefficient in dynamic applications we will use application based programs and interfaces. Although the application based programs need installation, we will use SmartClient instead. The SmartClient applications do not need installation and they can be run over the internet. Moreover, the SmartClient technology enables and facilitates versioning within the Project.

The next design constraint is related with the Application Sharing and Whiteboard tasks. In these tasks at most one client will be in control of the application. The main control will be in hand of the Presenter and s/he can give or take back the control to the other users when s/he wants. This will be useful for operations not to conflict on the application. Similarly in the Whiteboard facility again the Presenter has the control to draw or write on the board. But s/he can also let someone else to edit the board.

2 Architectural Design and Decomposition Description

2.1 Overall System architecture

In this section the overall system and its decomposition is described. The system is mainly divided into three: server, presenter client(s) and participant client(s). Each of these systems is further decomposed into several modules to maintain the level of complexity while also taking into consideration low coupling and high cohesion. The following subsection describes the modular decomposition of each part in more detail.

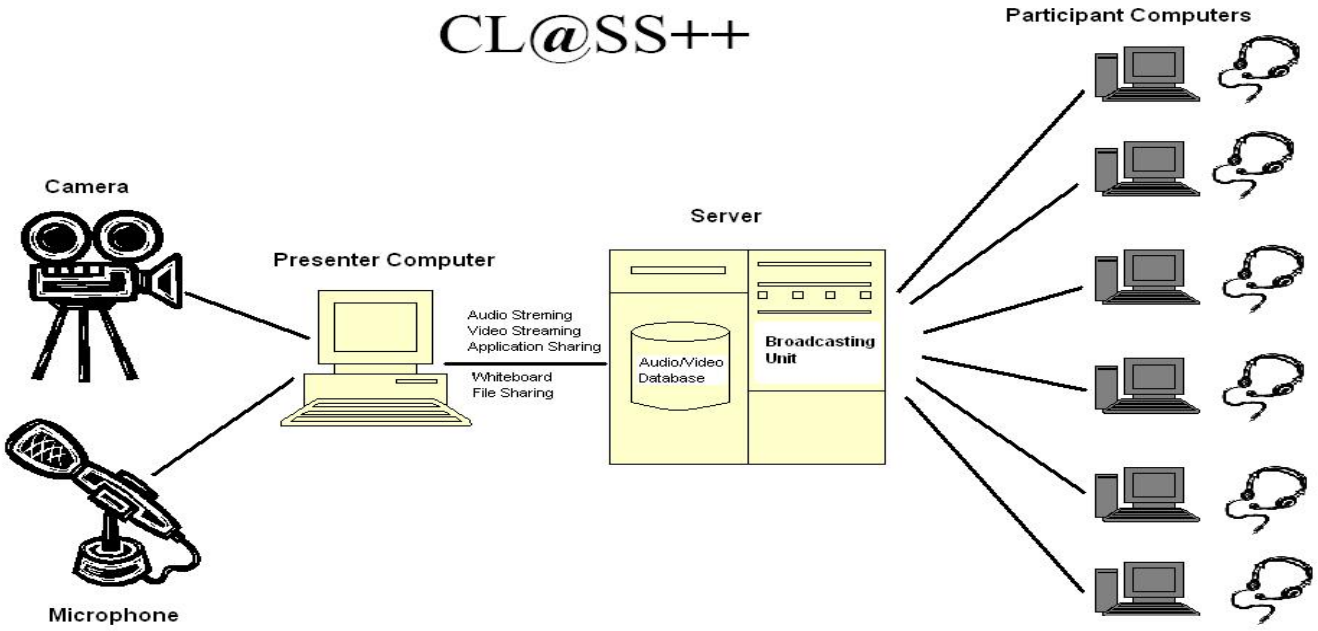


Figure 1 Overall View of the System

2.2 Presenter Client

A Presenter client has the control of one or more "virtual classes" and the ability to create a new one. However, the key functionality of a presenter client is being a source of information and information flow to be broadcasted to the participant clients in a synchronous manner. However, note that, this synchronous flow is maintained mostly on the server side.

The decomposition of the presenter client is based on the functionalities it can perform. These functionalities are:

- Streaming Video
- Streaming Audio
- Presenting Slides
- Using Whiteboard
- Sharing an application
- Transferring a File

Another important issue is that some or more of these functionalities should be carried out simultaneously. To enable these concurrent tasks to function in parallel without causing problems the system is designed to run modules largely independent from each other in different threads. Each of these functionalities is composed of two parts. While one of these parts deals with handling the activity on the client side, the other part deals with delivering the results of this activity to other participants.

Besides the functionalities listed above, the application will also perform other functionalities to satisfy the user requirements. These functionalities are:

- Monitoring participants
- Managing participants
- Administrating the virtual class environment

According to these criteria the presenter module is decomposed as in Figure 2

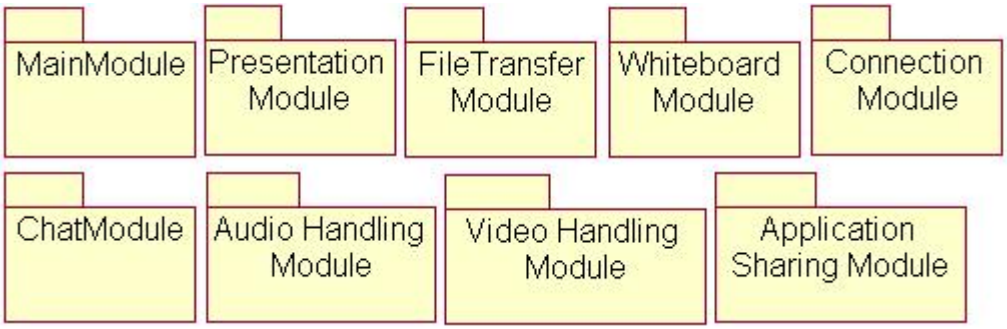


Figure 2 Presenter Module Decomposition

2.2.1 Main Module

This module is the entry point and backbone of the CL@SS++ application. The frame is a WindowsForm which is specified as an MDIContainer and will have many ChildWindows within.

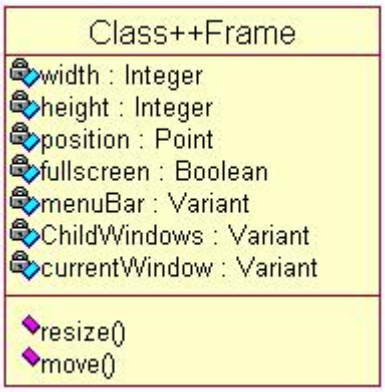


Figure 3 CL@SS++ Main Module

2.2.2 Connection Module

This module handles the connection with the server. Its functionality is to establish the connection and login.

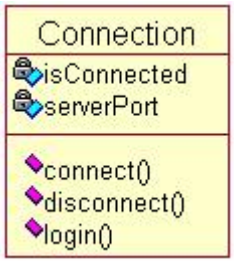


Figure 4 Connection Module

2.2.3 Presentation Module

This module allows the presenter to use PowerPoint slides during the Virtual Class. The Module consists of 4 classes (see Figure 5). The PresentationFrame class is a WindowsForm that displays the presentation within a Window. There can be one or more such frames. However, all are controlled by one PresentationToolbar. This class enables the user to perform desired actions on the presentation with the aid of the DrawingToolbar. The PresentationApplication is the actual class that will handle the presentations in the background. This will use a Visual Basic for Applications (VBA) library for Microsoft PowerPoint to use the PowerPoint Application installed at the presenter. This library can perform any action that can be performed with the PowerPoint Application. Some very useful actions are saveAs (in JPEG, GIF, HTML or any other format PowerPoint supports), openSlide, and goToSlide. Moreover, it is also possible to edit the presentation through this library.

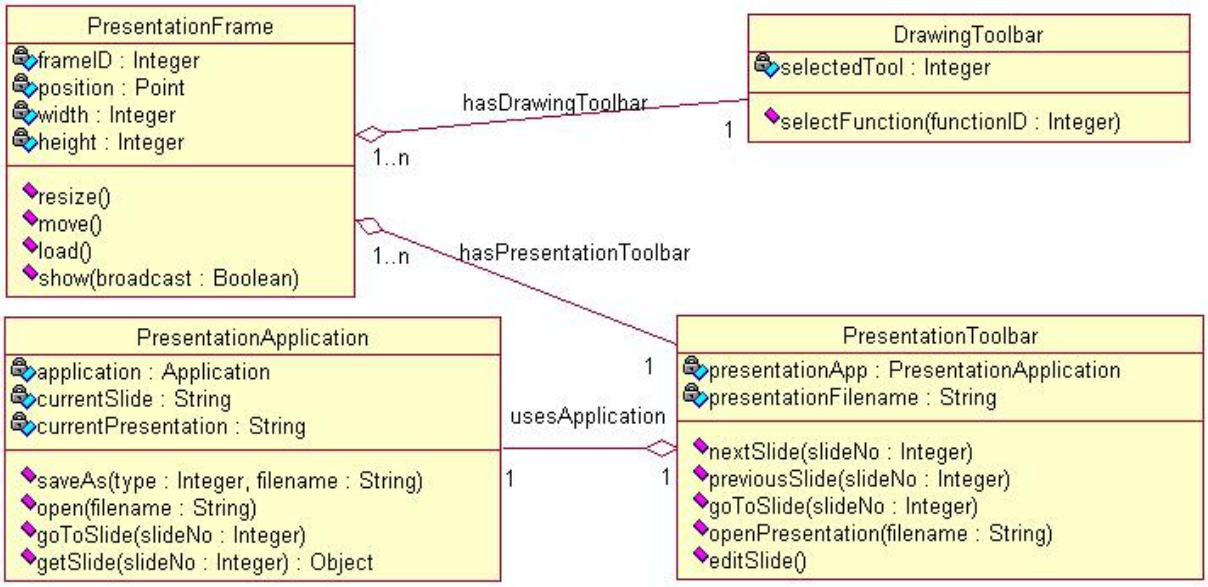


Figure 5 Presentation Module

2.2.4 Application Sharing Module

In the application sharing module we have the classes to implement our application sharing facilities. We have 3 classes to achieve this goal (see Figure 6) namely Application, ApplicationPresentationLayer, PresenterApplicationSharing. We will explain the details of the classes in the next paragraphs.

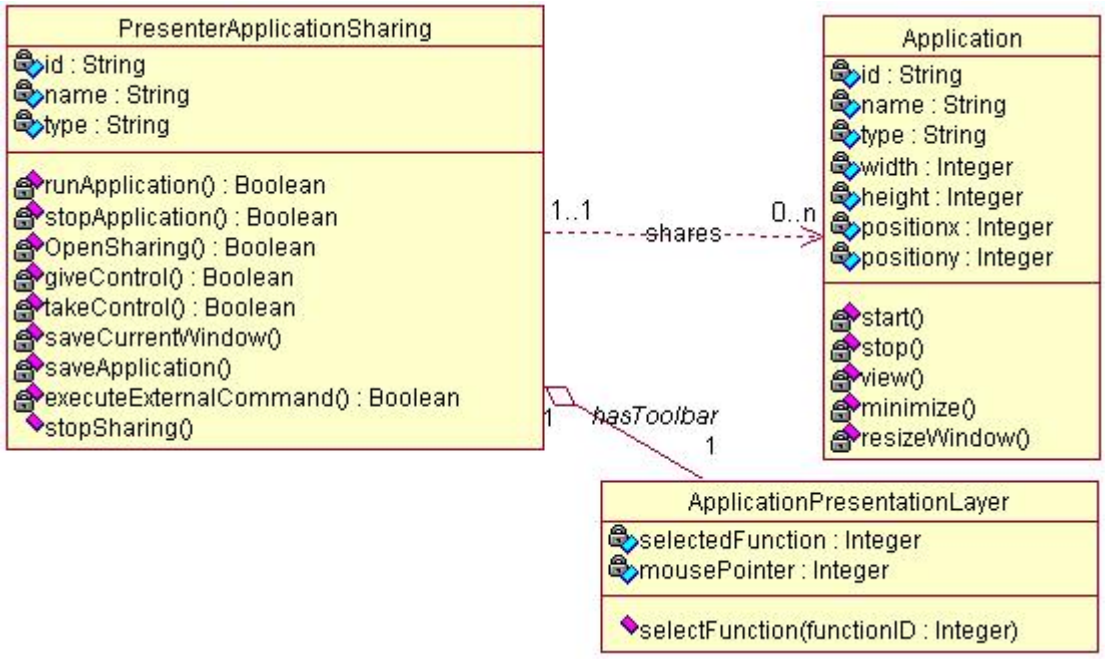


Figure 6 Application Sharing Module

To start with the Application Class it has application id, name, type and window properties such as width, height and position as the attributes. These attributes define the features of the application in the Presenter side to be shared with the others. The class has some methods to be applied on the application as starting and stopping the applications. Also it has some methods to apply window operations such as minimize or resize. This is the base class for the Presenter and the Participant

Classes. In the methods of the classes we will mainly use the MSDN library functions for window operations.

In the PresenterApplicationSharing class again we have the name, type and id as the attributes. This class is the Application class that resides on the Presenter client side. Some methods in the class are for Presenter to open, close, save whole or current image of application, and execute the application. Other methods in the class are opening the application to sharing, giving the control to some participant and taking back. After opening the sharing option the images are sent through sendImage method which is again in this class. Another method in the class is defined for executing the inputs or command coming from the Participants in the case of Control given to them. The taken commands and inputs are applied to the application and the results are sent back to all Participants. The images and permissions are sent through the network to the clients and inputs are taken back to the presenter client through network again. In the methods we will call the methods in Application Class to run the application as a normal application and we will send inputs to the methods in ShareApplication Class to transmit the images and the permissions.

The last class in this module is the ApplicationPresentationLayer Class which holds the application function type and mouse pointer information as attributes. It operates as a toolbar for selecting the function type by its method.

The collaboration diagram for this module is as in the following figure (see Figure 7). In this diagram the main GUI classes are started after login to the system. In this stage if the Presenter starts an application the Application class object is created and the methods in this class is called. The PresenterApplicationSharing class module starts after the Presenter selects to start an application. After this module has been started the Presenter can open the application for sharing and also may give control to the others and take back.

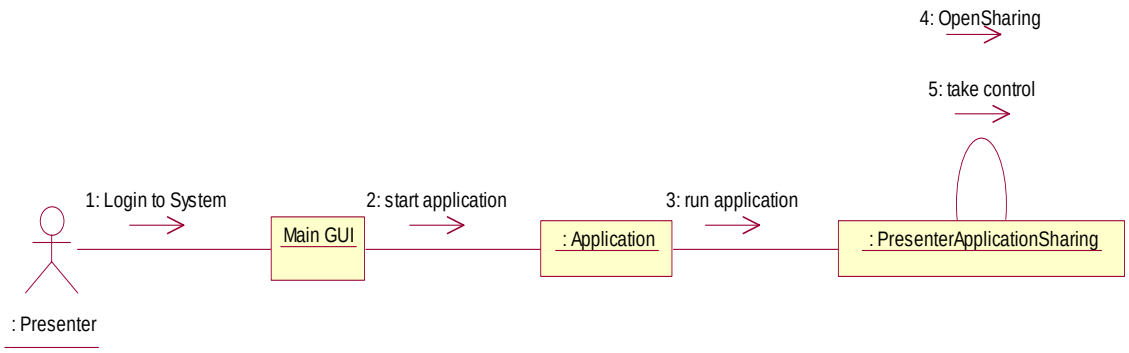


Figure 7 Collaboration Diagram for Presenter Application Sharing

2.2.5 Whiteboard Module

In the Whiteboard Module we have classes to implement Whiteboard facility. The classes are Board, PresenterBoard, ToolBox, and Palet (see Figure 8).

The Board Class is the general whiteboard window class with attributes width, height and position(x and y coordinates). It has methods to initialize window with given attributes and make some resizing and moving operations. Also it has editing methods such as write, erase used by the PresenterBoard Class.

The PresenterBoard Class has the same attributes as the original Board Class. It has methods to initialize a window calling the original Board Class and it has methods for editing the Board. As in

the Application Sharing Module, the board is sent as images to the Participants and thus this class has methods to save and send the image to the Participants. The editing methods use the MSDN library functions for drawing, erasing, copying and so on. The operations have similarities with Ms. Paint like programs. Therefore we can use the Win API for some of the operations. Again as in Application Sharing the PresenterBoard class has methods to give the control to the Participants and take back from them. And also it has methods to apply Participant's inputs and commands on the board.

Other than those board classes we have two other classes which are mainly for Presenter to write on the Board and edit it. First of them is the Toolbox Class which simulates a simple toolbox to write on the Board. The Toolbox class has the type attribute which holds the information about the type of the editing operation. These types can be as ‘writetext’ or ‘draw’ which are like the toolboxes in Paint like programs. The methods in the class are for showing and hiding the toolbox and also most important one is the method for changing the type attribute.

The second class is the Palet which is defined for the color of the drawing. By this class the current user of the Board can choose the color of the drawings or the text. The class has the color attribute. The methods are similar to Toolbox Class. They are for showing or hiding the Palet and changing the color.

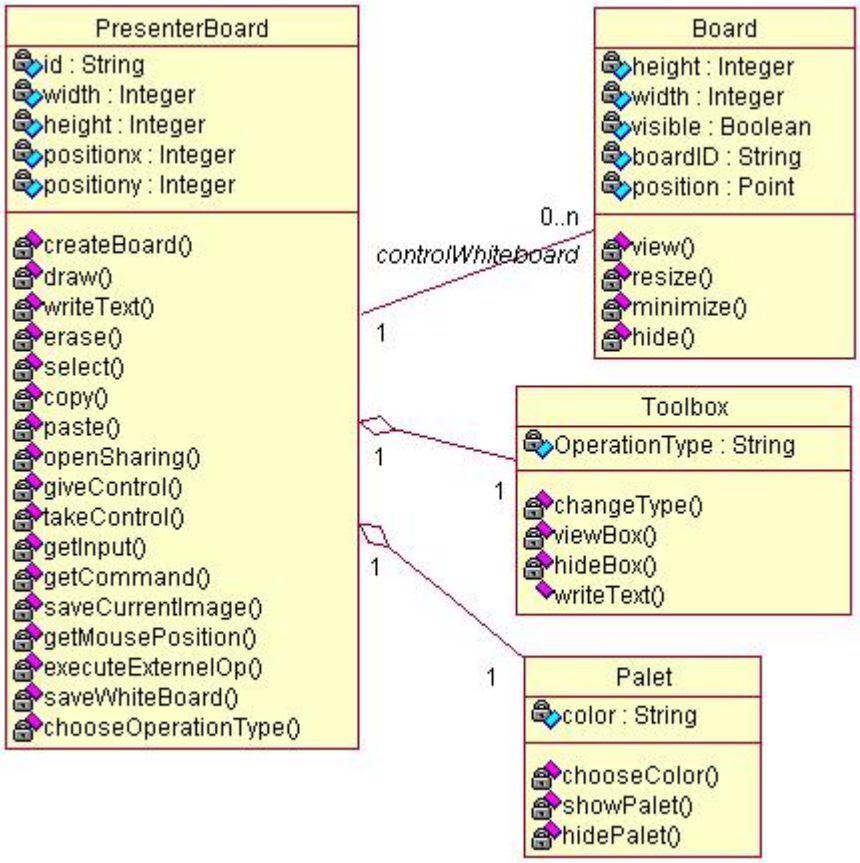


Figure 8 Whiteboard Module

The collaboration of the whiteboard classes in the Presenter side has the following figure (see Figure 9). In this figure the main GUI starts after system login and all the operations are followed after this. If the presenter chooses to open the board the Board Class object is created and a board window is initialized. After that if the Presenter chooses to use it in the lecture, it chooses to createBoard option which initializes the whiteboard by the method in PresentreBoard Class and shares it with all the users. If the Presenter gives the control of the whiteboard to the Participants, then this class can receive inputs from them to apply on the Board which is again a method in

PresenterBoard Class. Also From this state, the Presenter can choose to view the Toolbox or the Palet from the corresponding Classes to be able to use different edit options or different colors.

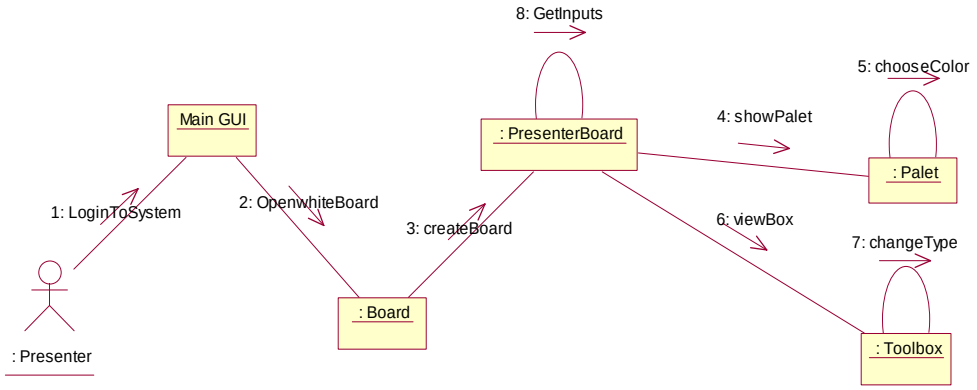


Figure 9 Collaboration Diagram for Presenter Whiteboard

2.2.6 Chat Module

This module (see Figure 10) handles the actions to chat with other users, either participant or presenter. The MainWindow represents the main chat window.

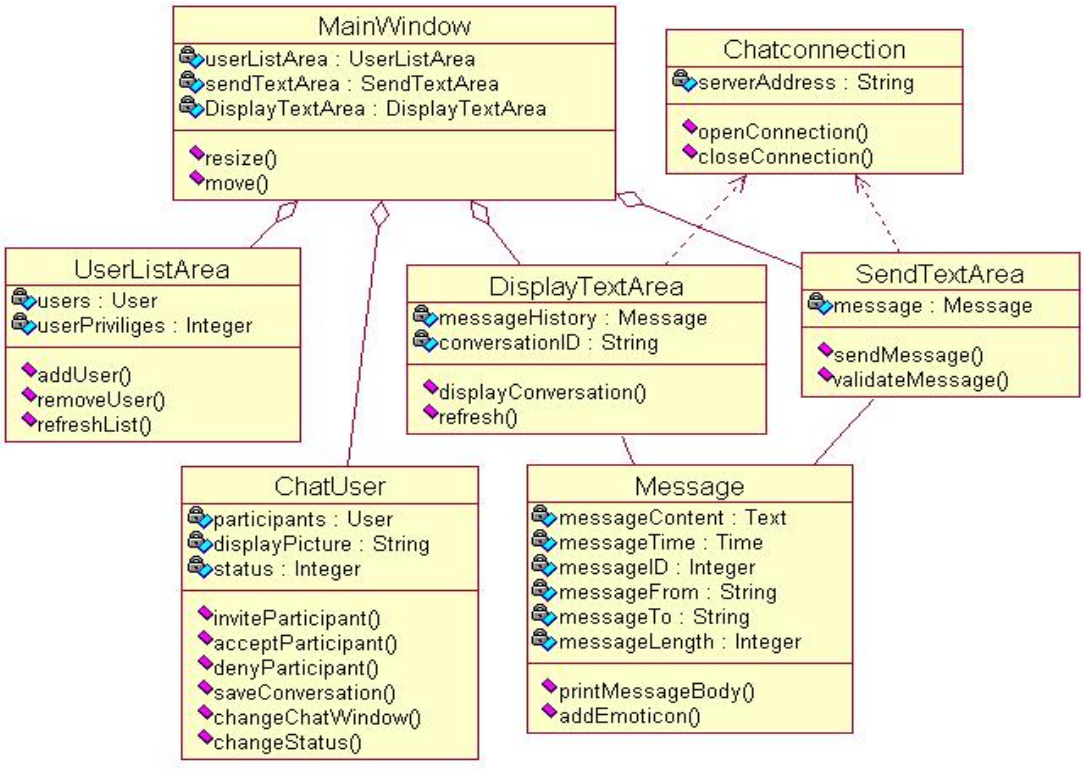


Figure 10 Chat Module

2.2.7 File Transfer Module

The class diagram for this module is in Figure 11. This module is responsible from sending files to the server for future use or sharing.

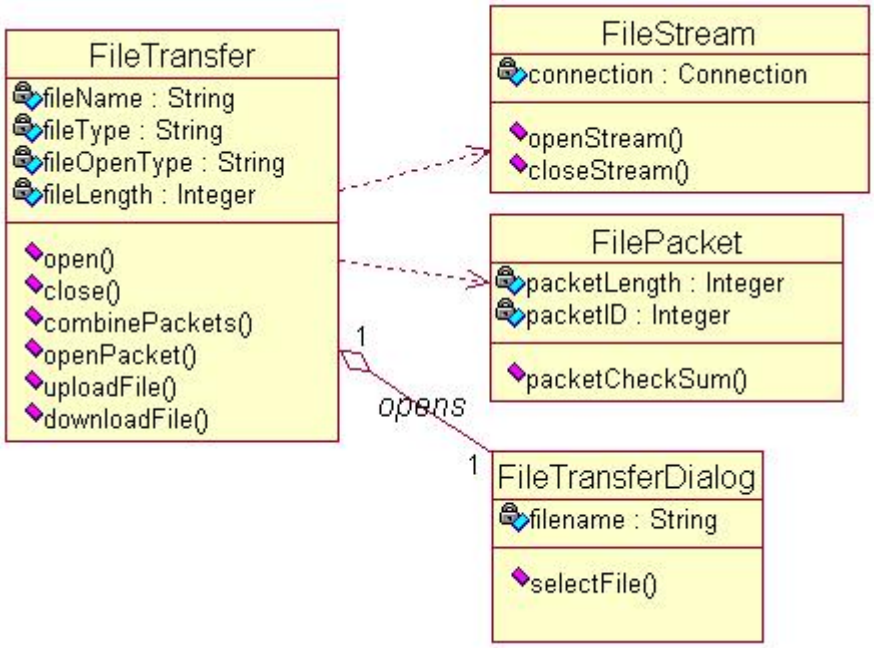


Figure 11 File Transfer Module

The presenter who wants to download or upload a document is confronted with a file transfer dialog window. FileTransferDialog class takes the name and location of the file by its selectFile(String) function. Then FileStream class is used to open a connection, after that file is separated into packets by the help of FilePacket class. Finally, the file is opened and combined by FileTransfer class. The Figure 12 below is representing that event visually.

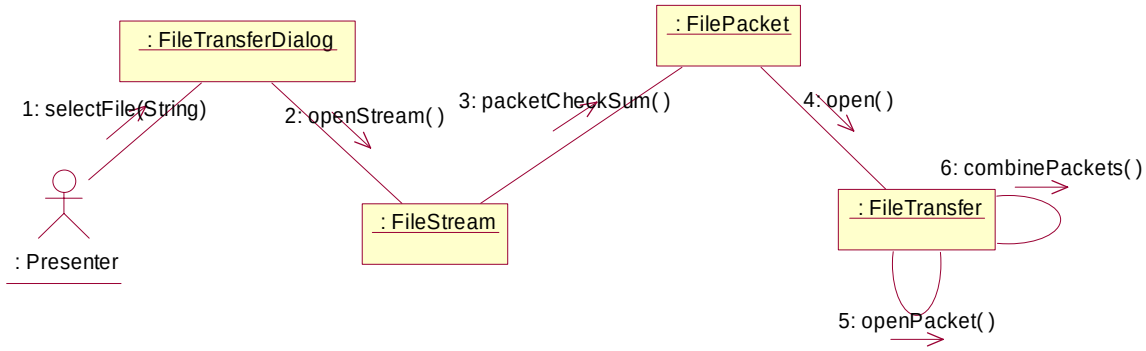


Figure 12 Collaboration Diagram for presenter File Transfer

2.2.8 Video Handling Module

In that module we have three classes to handle the video streaming issue on the presenter client side (see Figure 13). These classes are VideoCapturer, VideoPackager, VideoCompress.

The VideoCapturer class is responsible for starting/stopping video streaming and getting video streams from the video capturing device of the presenter. It has some attributes to define the height, width and the quality of the capturing. It also keeps the counter of number of frames per second. It also keeps information of about its source.

Our second class which is VideoPackager class deals with packaging of the video streams. In order to achieve its goal this class gets the video stream which is continuously flowing from the VideoCapturer class. It is responsible from packaging of these flowing video streams.

The third class which is responsible of compression of video packages is VideoCompresser class. It gets the video packages and compresses them in order to minimize packet sizes. It is necessary not to exceed the bandwidth of the LAN.

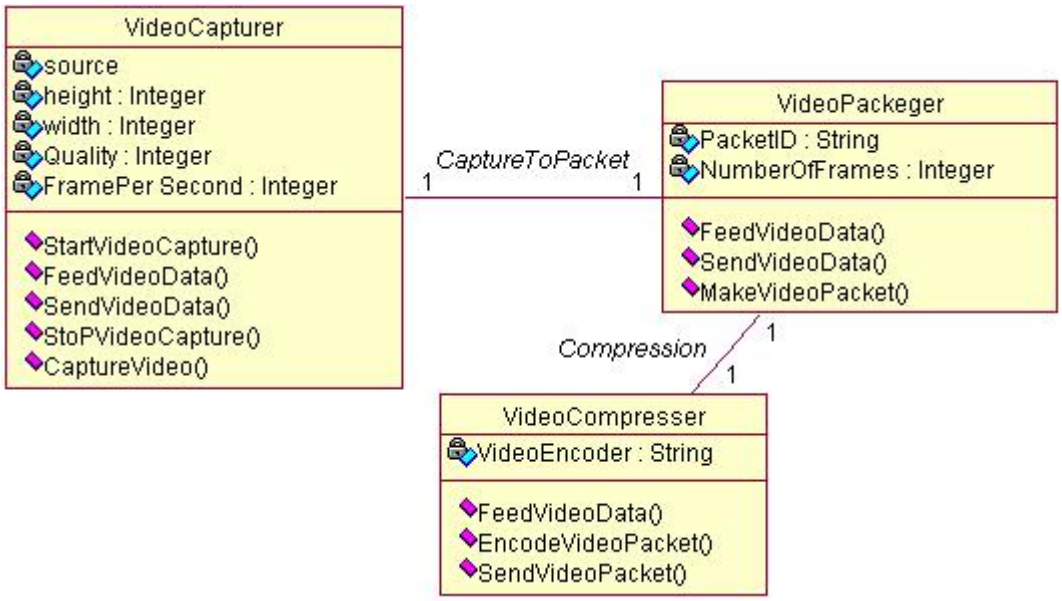


Figure 13 Video Module

The collaboration diagram (see Figure 14) below explains the presenter side use cases. The presenter starts the video capturing by using StartVideoCapture function then VideoCapturer class starters the capturing event then the captured data is fed in to VideoPackager. In that class the video stream are packaged and then fed in to the VideoCompressor class. This class uses an encoder to compress the video data and send these compressed packages to server.

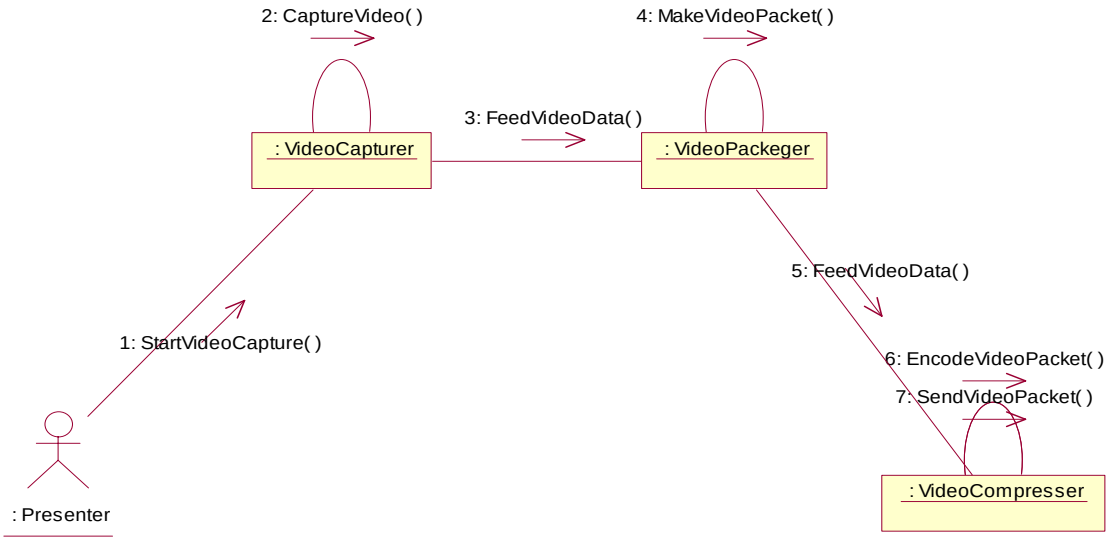


Figure 14 Collaboration of Broadcast Video Presenter Side

2.2.9 Audio Handling Module

In that module we have three classes to handle the Audio streaming issue, too. These classes are AudioCapturer, AudioPackager, AudioCompresser.

The AudioCapturer class is responsible for starting/stopping audio streaming and getting audio streams from the microphone. It has some attributes to define the quality of the capturing. It also keeps the counter of number of bits per second. It also keeps information of about its source.

Our second class which is AudioPackager class deals with packaging of the audio streams. In order to achieve its goal this class gets the audio stream which is continuously flowing from the CaptureAudio class. It is responsible from packaging of these flowing audio streams.

The third class which is responsible of compression of audio packages is CompressAudio class. It gets the audio packages and compresses them in order to minimize packet sizes.

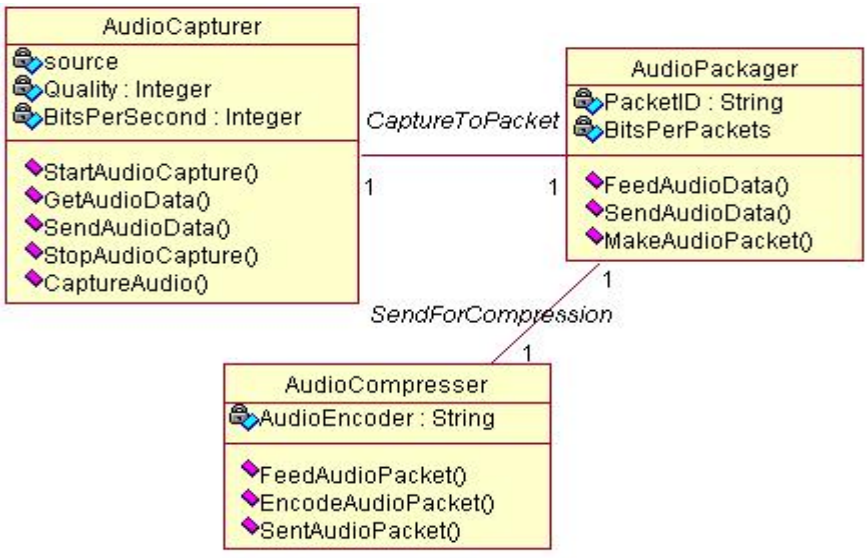


Figure 15 Audio Handling Module

The collaboration diagram (see **Figure 16**) below explains the presenter side use cases. The presenter starts the audio capturing by using StartAudioCapture function then AudioCapturer class starters the capturing event then the captured data is fed in to AudioPackager. In that class the audio stream are packaged and then fed in to the AudioCompressor class. This class uses an encoder to compress the audio data and send these compressed packages to server.

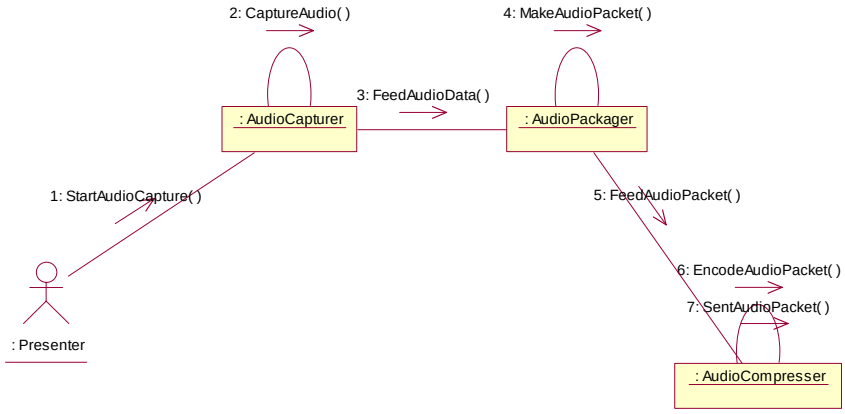


Figure 16 Collaboration of Broadcast Audio Presenter Side

2.3 Server

The server is a centralized control mechanism for the whole system. Its main functionalities are:

- Managing users and user connections
- Storing virtual class data
- Synchronizing and broadcasting live virtual class content
- Storing and providing virtual class content

2.3.1 User Connection Handler Module

This module provides the classes to handle some tasks such as user login and rub applications. The classes in this module are mainly the user classes and the ConnectionHandler Class. Also the ConnectionListener Class is present to check whether a connection is requested from any of the users. (See Figure 17)
To start with the user classes they all have attributes describing the user. All the classes of this category are derived from the User Class. They hold information about the users related with the type of the user. The basic methods in these classes are joinClass and leaveClass methods which are common to all user classes. The User class also has the methods named loadDefaultSettings and saveSettings methods.

The ConnectionListener class has serverPort and serverSocket as attributes which identifies the Connection. The methods in this class are listenPort, which waits for the connection, and the accepUser method, which accepts the user request for connection.

The ConnectionHandler Class is the last class in this module and has the user and the socket as the attributes. The methods in this class are for logging on and running the connection after login. These methods are named as login and run.

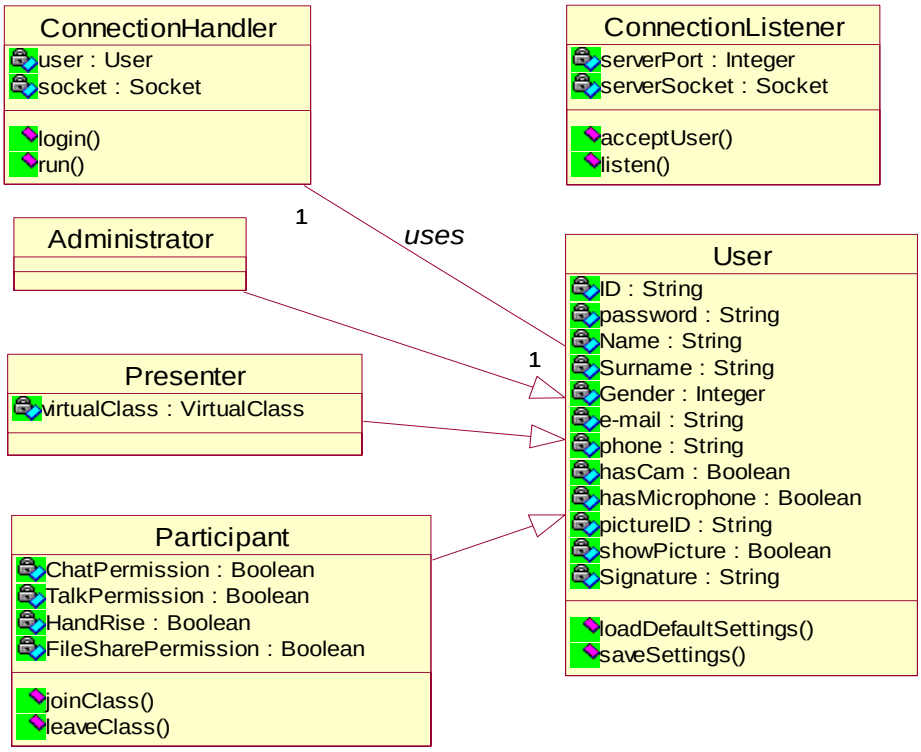


Figure 17 Class Diagram of Connection Handler Module

The Figure 18 shows the collaboration diagram for connection handler in system server. The three different user types which are Administrator, presenter, participant connect the system by using connection listener. This module helps the user to log in to server bye using ConnectionHandle class function login(). Then connection to server is completed.

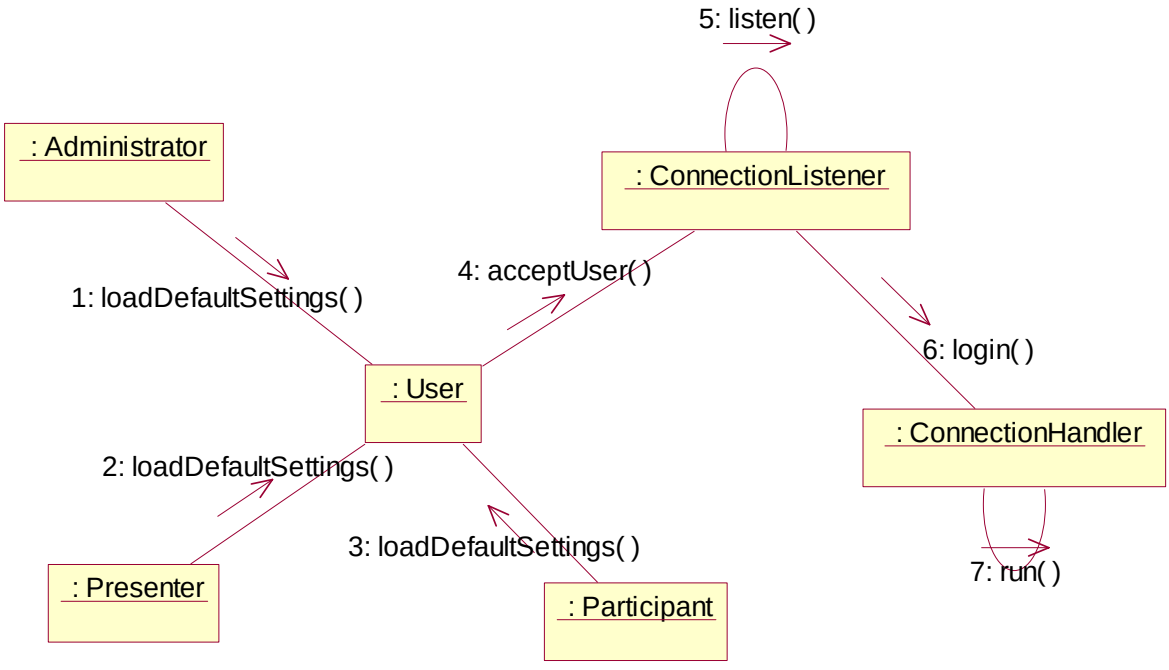


Figure 18 Collaboration Diagram for Connection Handler Module

2.3.2 Virtual Class Data Module

This module is the base module for the whole system in the server side. It has the main class for all of the tasks we provide the users for a particular session of virtual class (see Figure 19). The first attribute of this class is the classID, which is the distinct virtual class environment. The second one is presenters, which is the information about the current presenters of the virtual class. The third one is the participants which is the list of the participants. The other attributes are the maxParticipants, broadcaster, recurrencePattern and startTime.

The methods in this class are mainly the accomplishing the tasks of the classes from all of the classes of the clients. The methods works consistently with the other class methods since all of them are called form those classes. The first method in this class is addPresentation to add presentation to the virtual class. The addVideo, addAudio and addWhiteboard methods works similarly to add the corresponding items to the current session. The shareApplication method starts the application sharing. The start method starts the virtual class. The schedule method makes the changes on the schedule of the classes. AddParticipant method adds new participants to the class. The uploadFile method uploads the files from the disk and the inviteParticipant method sends invitation to the participants for the class session.

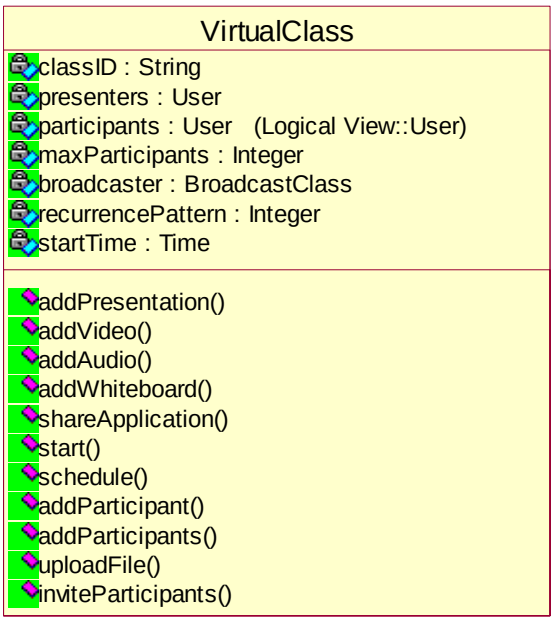


Figure 19 Class Diagram of Virtual Class Module

2.3.3 Synchronization and Broadcast Module

In this module there are classes to achieve the transferring the data, streaming audio-video, synchronizing them and broadcasting to the clients. There are eight classes for these tasks (see Figure 20). Their names are UserStatus, VideoBroadcast, AudioBroadcast, WhiteboardBroadcast, PresentationBroadcast, ApplicationBroadcast, and Synchronizer.

Starting from the UserStatus Class it has port and socket as attributes. These keep the port and socket numbers of the user connection. The send method in this class is responsible for sending the status of the user. The status info of the users includes type of the user and the port and socket numbers.

The VideoBroadcast Class is the class responsible for the video stream. The attributes of the class are port, socket and source. The methods are send and getVideoStream which are responsible for getting and sending the video stream.

The AudioBroadcast Class is the same as the VideoStream Class having the same kind of attributes and methods. These are again responsible for sending the audio to the clients.

The WhiteboardBroadcast, PresentationBroadcast and ApplicationBroadcast classes have the same attributes as the above classes. These attributes keeps the same kind of information about the corresponding classes. All of the classes have the methods for sending the whiteboard image to the clients.

The last class of this module is the Synchronization class. This class keeps the state of the data if it is ready (synchronized) or not. The methods of this class are wait, which waits for the video and audio to be synchronized, synchronize, which synchronizes the audio and video, isReady, which checks the state of the data.

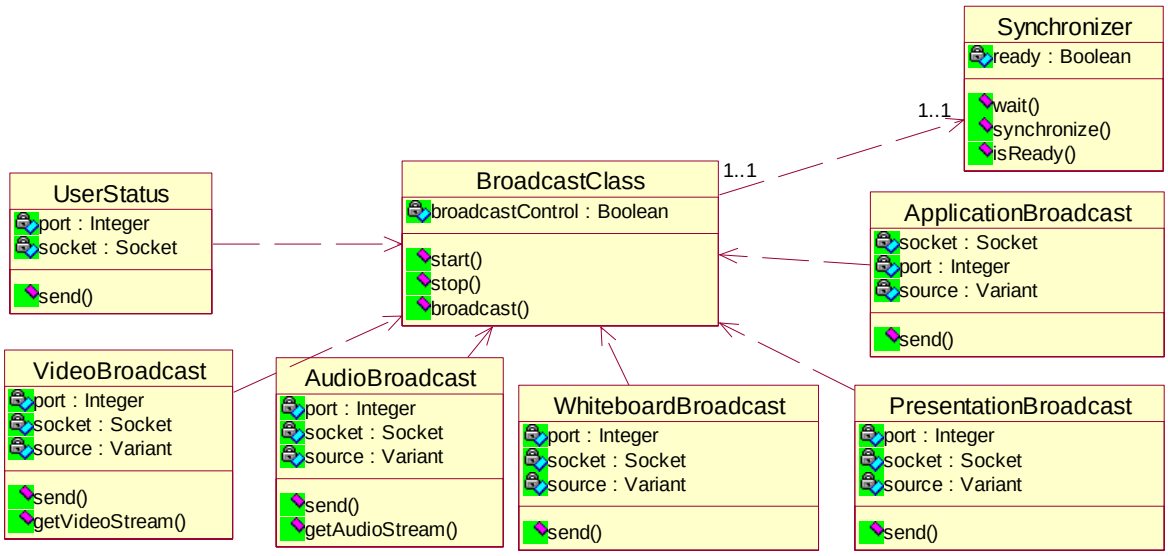


Figure 20 Class Diagram of Synchronization and Broadcast Module

The collaboration diagram (see Figure 21) describes the broadcast event. We are actually broadcasting not only the audio and video but also whiteboard presentations and the applications. To achieve our goal we are using our BroadcastClass. The Broadcasting class checks whether the Synchronizer is ready or not. If it realizes that the Synchronizer is ready it sends its data. Then our Synchronizer class harmonizes these data.

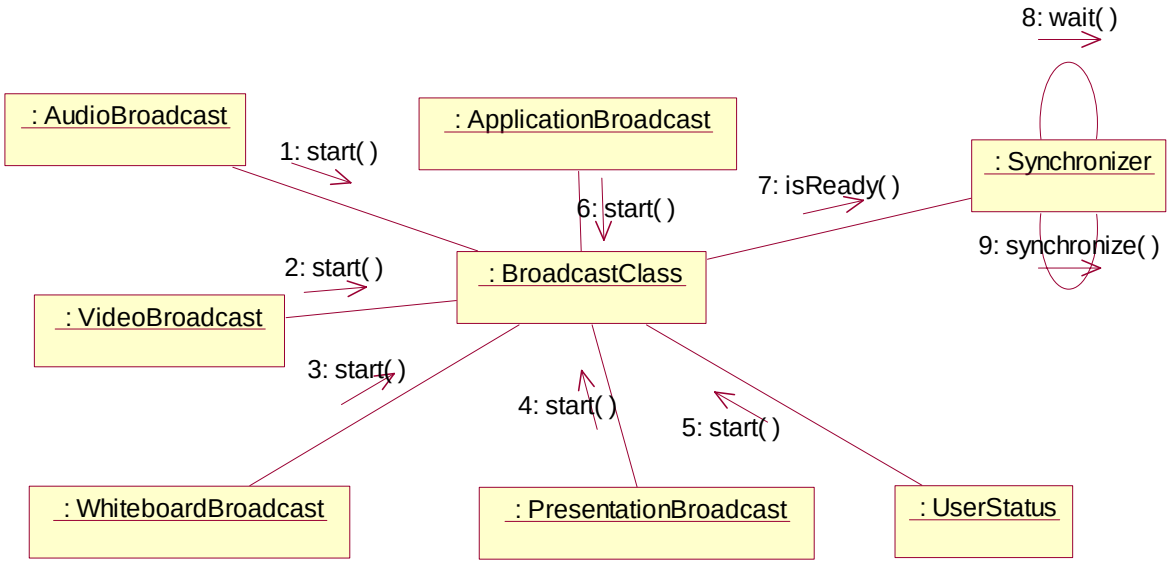


Figure 21 Collaboration Diagram for Synchronization and Broadcast

2.3.4 Database Connection and Storage Unit of Virtual Class

The last module of the server side is the Database Connection and Storage Unit. This module is responsible for Database operations and the storage of the intermediate files during any class session. The classes in this module are the DBConnection class and FileStorageUnit. (See Figure 22

The DBConnection class is the class for the methods of database operations. The attributes of this class are hostname, portNumber, name, password, DBname, DBUsername, DBUserpassword. This information is used when a database connection is to be started. The methods in this class are are basic database functions such as query, insert, update and delete, and the connect method for establishing connection. Also there is a method for checking the connection if it exists or not.

The FileStorageUnit class has filename, type and the storageDirectory as attributes. These attributes are used when storing or retrieving files from the database. The methods are namely saveFile and loadFile methods and they do the operations as their names. They keep all the file information of the files to be saved in the server side

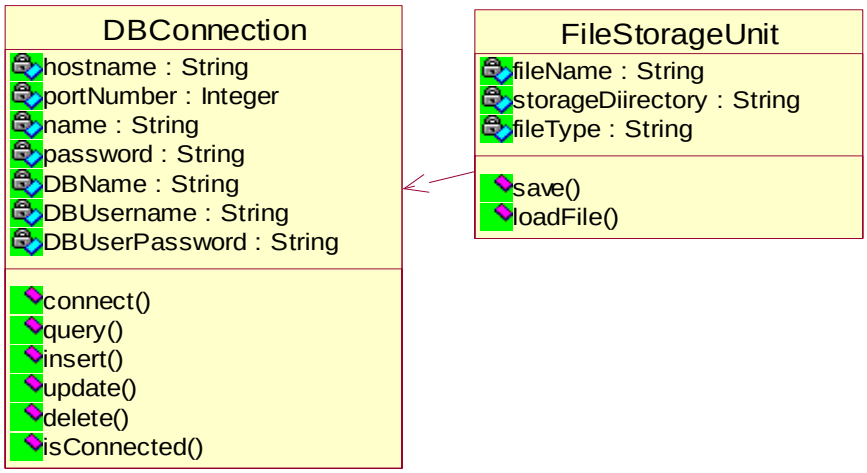


Figure 22 Class Diagram for Database Connection and File Storage Unit

The file storage unit in the server side deals with the database management of the lecture data such as presentations, audio video streams etc. We have a system database which is connected from FileStorageUnit. We have some operations that we can do on our database such as insert, delete and update operations.

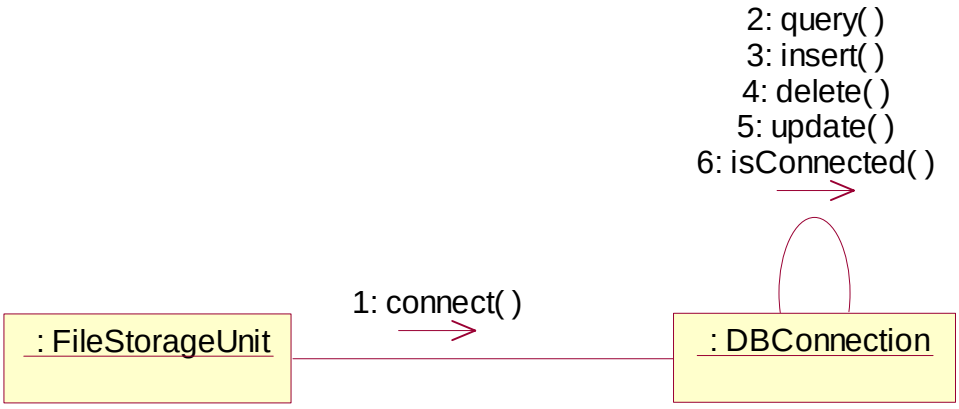


Figure 23 Collaboration Diagram for Database Connection and File Storage Unit

2.4 Participant

The participant client is similar to the presenter client, but with rather limited functionalities. However, there is also an important difference. The presenter is in general responsible for delivering content, whereas the participant client is primarily interested in receiving and formatting this content.

2.4.1 Main Module

This module is the entry point and backbone of the CL@SS++ application. The frame is a WindowsForm which is specified as a MDIContainer and will have many ChildWindows within.

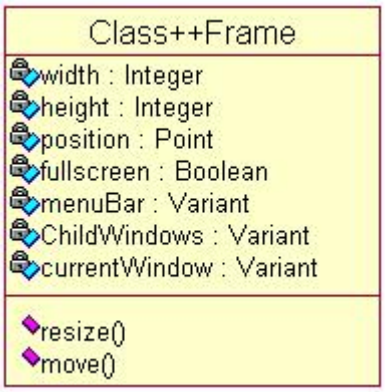


Figure 24 CL@SS++ Presenter Main Module

2.4.2 Connection Module

This module handles the connection with the server. Its functionality is to establish the connection and login.

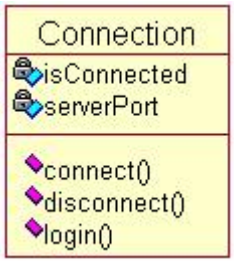


Figure 25 Connection Module

2.4.3 Presentation Module

This module enables the participant to view the presentation broadcasted by the presenter or a previous presentation from the server. The participant has not as much control of the presentation as the presenter. The isLive Boolean is to further restrict the user. For example when the presentation is live the user can not control the presentation flow. However, s/he may do it by opening another Window and viewing a local copy.

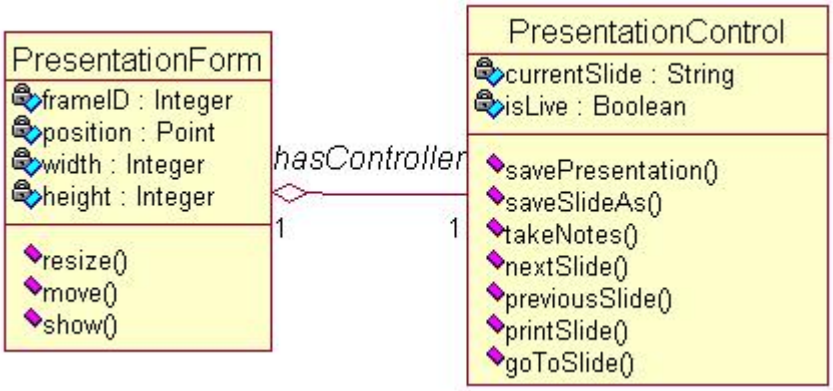


Figure 26 Presentation Module

2.4.4 Application Sharing Module

This module lets the participant share the application of the presenter (see Figure 27) and consists of 3 classes. These classes are detailed in the following paragraphs.

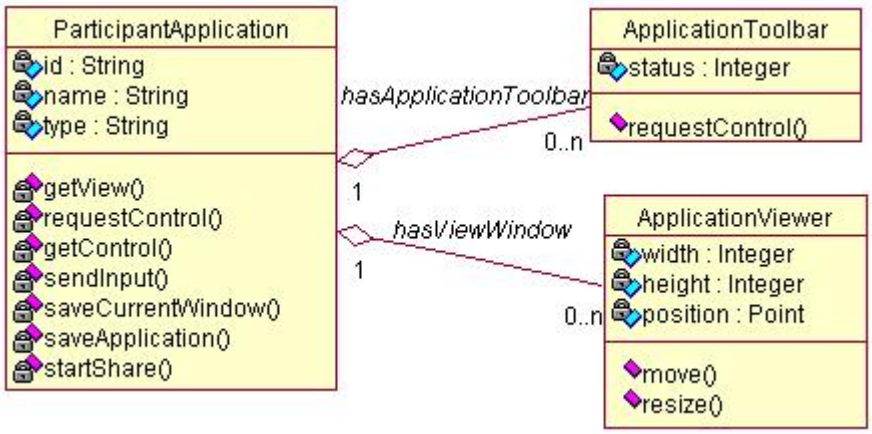


Figure 27 Application Sharing Module

The ParticipantApplication Class, used by the Participant Clients, has the same attributes as the main Application Class but the methods are different from both of the above classes. Firstly it has a method to open an application window which initializes the window to take the images from the real application. Note that we will not transfer the real application inputs to the participants but the current image of the application window. After initialization the Participant can receive the images whenever they are sent by the ShareClass class and displays them inside the window by the methods inside it. Also the ParticipantApplication Class has methods to save the current image or the whole application images in order which uses basic MSDN Win API for saving. These methods use the MSDN Windows Media library functions to display the images and make operations on them. Another method in this class is request for Control to use the applications. After request if the Presenter gives the control the participant, s/he can send inputs, commands and mouse operations to the application using the methods for sending those. All these are sent as inputs to the ShareApplication Class.

The ApplicationViewer and ApplicationToolbar classes work on the Participant Client side. The ApplicationViewer Class has the window position and size info as attributes and has methods for moving or resizing the application frame window in the client side. The methods in this class will mainly use Win API functions to achieve the required task. ApplicationToolbar Class has the status info of the shared application frame as it has control or not. From this toolbar we can make control request by requestControl method. The requests will be sent through the methods in ShareApplication Class.

The classes in this module are correlated as in the following figure (see Figure 28). The participant can access the facilities by logging on the system which starts the main GUI for Participant client side. From the menu in this GUI, the Participant may choose to initialize a window to get views from an application if the Presenter shares one, which is initialized by the Application Class. Then if the Participant wants the frames s/he creates ParticipantApplication Class by shareApplication method in this class. In the shareApplication Class the user call the methods as requesting and getting the control or only receive the image frames from the real Application.

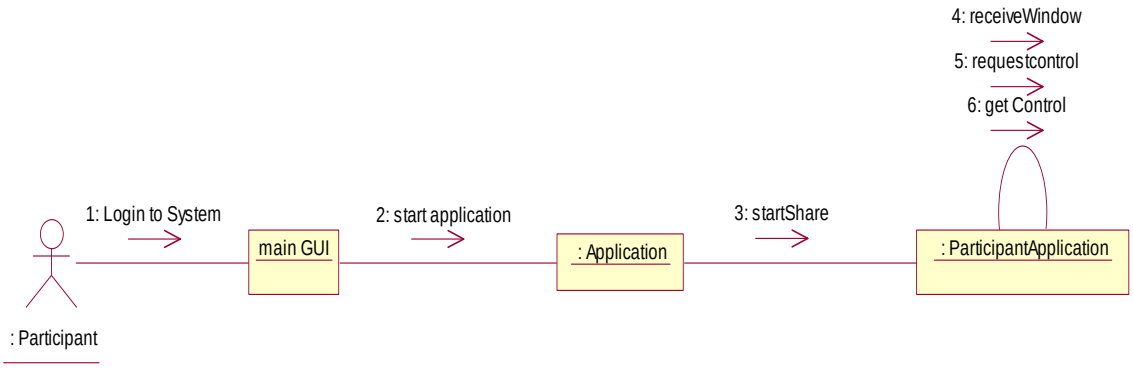


Figure 28 Collaboration Diagram for Participant Application Sharing

2.4.5 Whiteboard Module

This module lets the participant to view the Whiteboard that the presenter uses. This module has 3 classes (see Figure 29), namely ParticipantBoard, ParticipantToolbar, and BoardViewFrame.

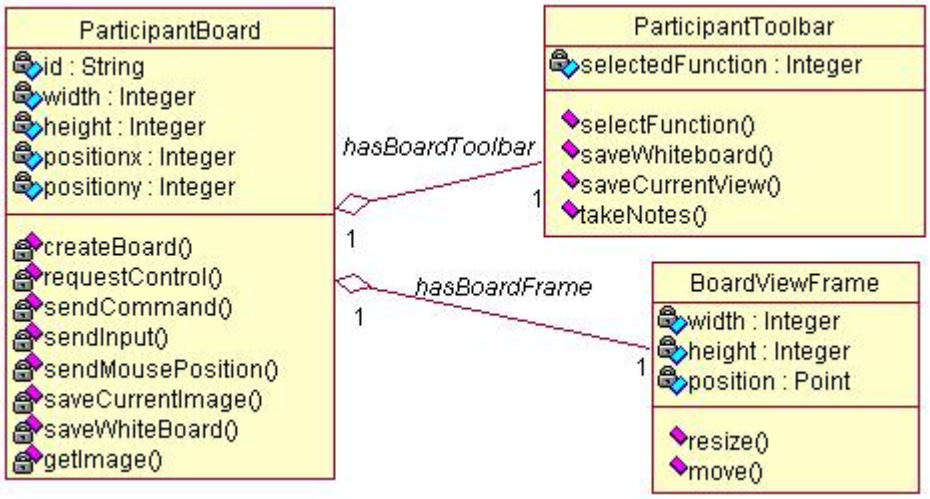


Figure 29 Whiteboard Module

The ParticipantBoard Class has again the same attributes. But the methods are different than others. In this class there is a method to initialize the window. It has method to get the image of the current board from the PresenterBoard class when there is a change on the Board. Also there are methods to save the current board and save all of them in order. The Participant can request control from the Presenter and if s/he takes the permission s/he can send input, mouse information or command to be executed to the PresenterBoard class.

The next one is BoardViewFrame which holds the viewing frame of the whiteboard in the Participant side. It has attributes as width, height and position. The methods inside it are resizing and moving which are implemented as the other similar methods.

The final Class is the ParticipantToolBar. This toolbar is used by the Participant if s/he has the control of the whiteboard. The attribute of the Toolbar is the function type and the methods on this class are selectfunction, savecurrentview, saveWhiteboard and takenotes. These methods are used by the Participants and have similar functions as in the Presenter side.

The collaboration Diagram for these classes is as in the next figure (see Figure 30). The Participant again enters the system from the login page. After login the Participant can initialize a board window to take the images from the whiteboard using Board Class method. After creating the board, to take the images from the real whiteboard the Participants initializes a ParticipantBoard class object. They can get the image from the real whiteboard by using the getImage method from this Class

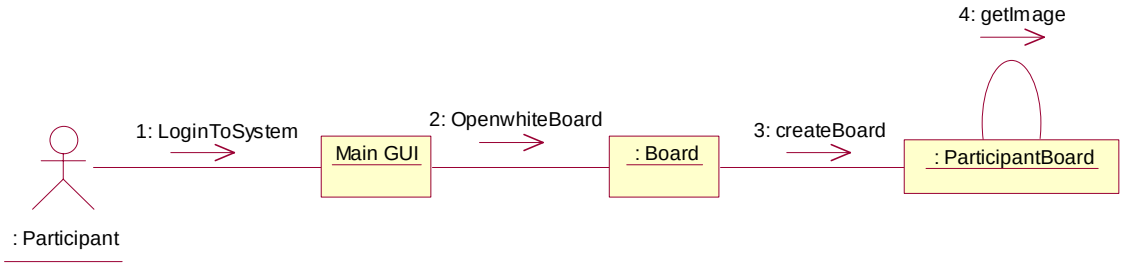


Figure 30 Collaboration Diagram for Participant Whiteboard

2.4.6 Chat Module

This module handles the actions to chat with other users, either participant or presenter. The class diagram of this module is as in Figure 31. This module is similar to the module on the presenter client side. However, the presenter may have the privilege to control the chat operation of the participants, allow chat during classes or not.

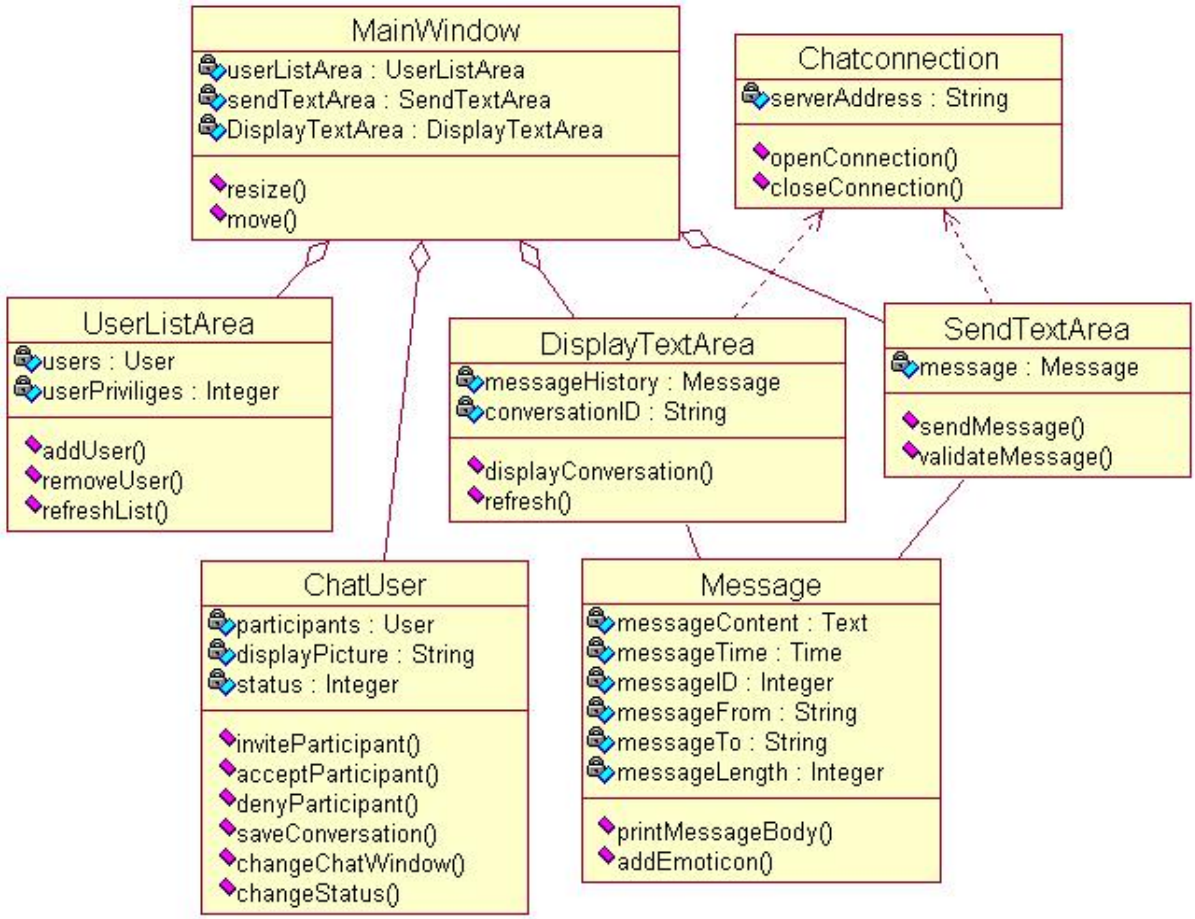


Figure 31 Chat Module

The Participant is confronted with the main window and the MainWindow class of the module makes all of the initializations. Participant can see the entire user by the UserListArea class. The Participant can add or remove user with addUser and removeUser functions of that class. The chatUser class enables the Participant to chat with other Participants. The message is saved into the Message class. The saved message is transferred to the server side by making connection with the Chatconnection class. After the connection is established the message is transferred to the server side. Finally the transferred messages are received from other Participants and displayed with the DisplayTextArea class. You can see the overall process from Figure 32.

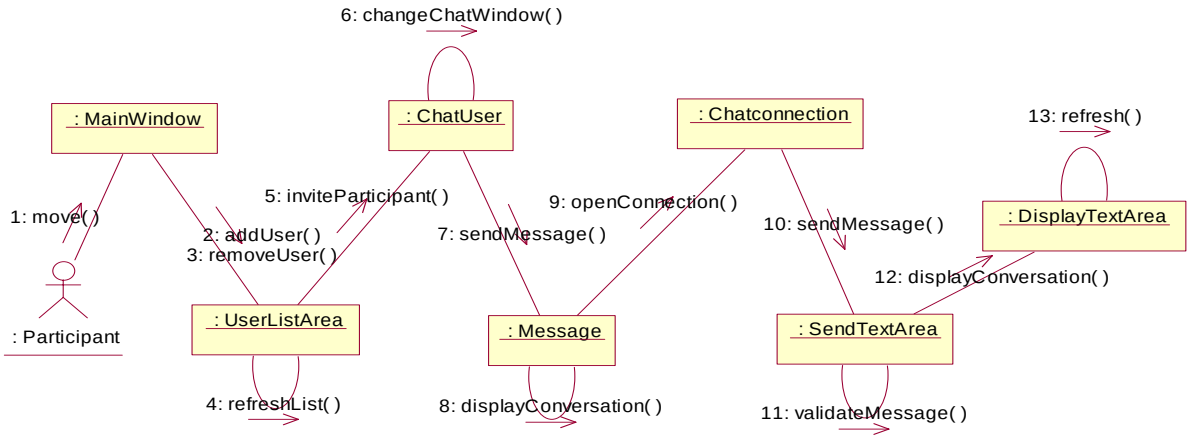


Figure 32 Collaboration Diagram for Participant Chat Module

2.4.7 File Transfer Module

This module enables the participant to upload/download files to/from the server. The module consists of mainly 4 classes. The FileTransferDialog provides user interaction and takes the location of the file to be uploaded or the download location.

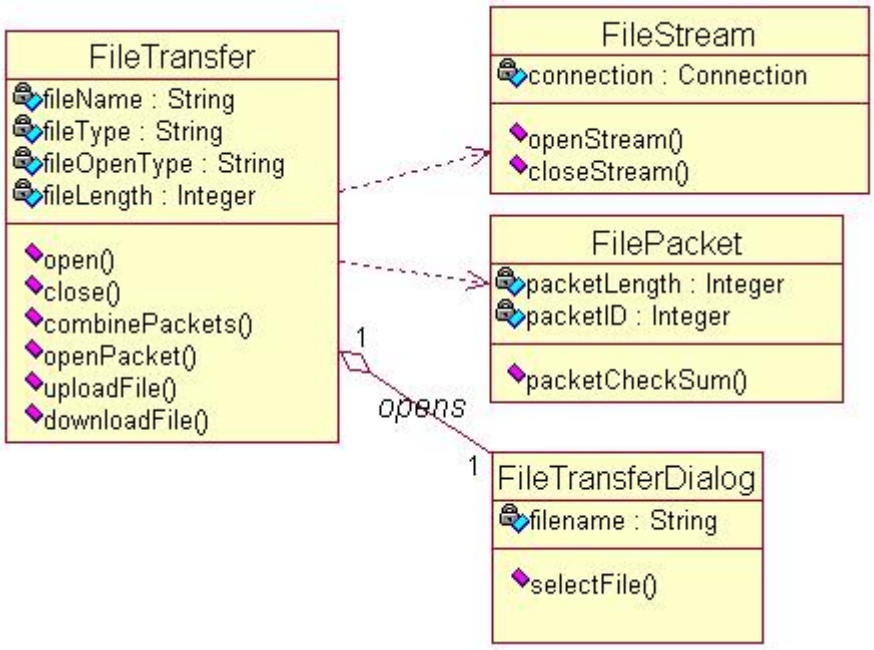


Figure 33 File Transfer Module

The Participant FileTransfer module is similar to Presenter File Transfer collaboration diagram. The processes occur in the same sequences as in the Presenter side. The participant who wants to download or upload a document is confronted with a file transfer dialog window.

FileTransferDialog class takes the name and location of the file by its selectFile(String) function. Then FileStream class is used to open a connection, after that file is separated into packets by the help of FilePacket class. Finally, the file is opened and combined by FileTransfer class.

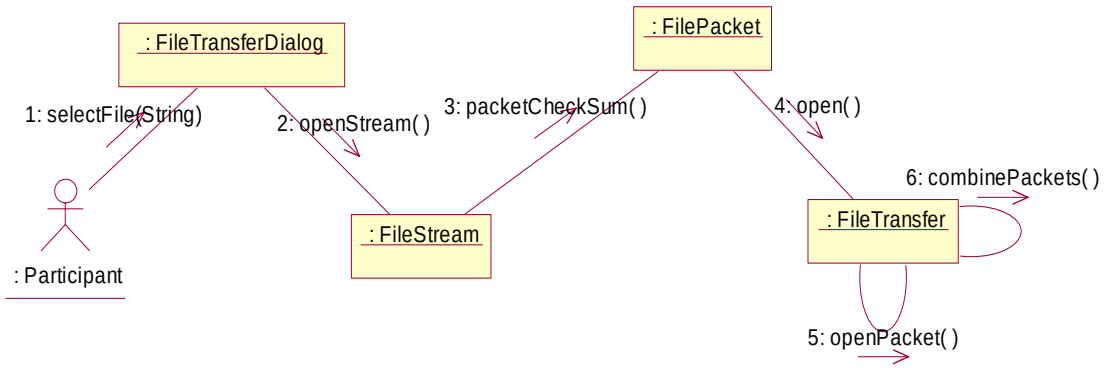


Figure 34 Collaboration Diagram for Participant File Transfer

2.4.8 Audio/Video Handling Module

This module lets the participants to play the video and audio packages that are broadcasted by the server. This module

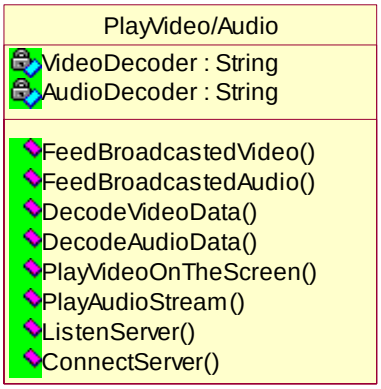


Figure 35 Audio/Video Handling Module

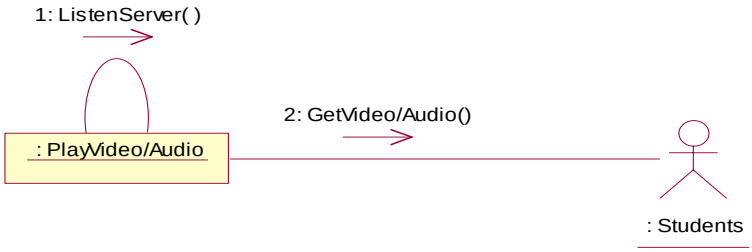


Figure 36 Audio/Video Handling Collaboration Diagram

2.5 Data Decomposition

2.5.1 Data Dictionary

Name :	Visual Data
Aliases :	Picture, Frame, Video
How and where used :	Is captured from video device and broadcasted to participants
Format :	Byte stream

Name :	Sound
Aliases :	Voice, Audio
How and where used :	Is captured from microphone and broadcasted to participants
Format :	Byte stream

Name :	Screen Display
Aliases :	Screenshot, Print screen
How and where used :	Is used to share desktop with other participants
Format :	ASCII character file, Binary file

Name :	Video Stream
Aliases :	Lecture Movie
How and where used :	Is used to transfer the video content
Format :	Byte stream

Name :	Digital Sound
Aliases :	Voice Data, Audio
How and where used :	Is used to transfer the audio content
Format :	Byte stream

Name :	Message
Aliases :	Chat, Private Chat, Public Chat, Ask Question
How and where used :	Is used to communicate with text
Format :	Stream of ASCII characters

Name :	Whiteboard
Aliases :	Blackboard, Paint Program
How and where used :	Corresponds to the blackboard in the classroom
Format :	Image file, ASCII character file, Binary file

Name :	Compressed Video
Aliases :	Encoded Video, Zipped Video
How and where used :	Is used to optimize the transmission of data over some network
Format :	In raw binary packets

Name :	Compressed Audio
Aliases :	Encoded Audio, Zipped Audio
How and where used :	Is used to optimize the transmission of data over some network
Format :	In raw binary packets

Name :	Video Packet
Aliases :	Video Bundle
How and where used :	Is used to pack the captured video stream from camera
Format :	In raw binary packets

Name :	Audio Packet
Aliases :	Audio Bundle
How and where used :	Is used to pack the captured audios from microphone
Format :	In raw binary packets

Name :	Application Sharing
Aliases :	Remote Desktop Sharing
How and where used :	Is used to share an application with the whole class without the installation of the application by the users
Format :	Image file, ASCII character file, Binary File

Name :	E-mail
Aliases :	Electronic Mail
How and where used :	Is used when a user send something to teacher or to his/her friends
Format :	ASCII File, Image File

Name :	Students
Aliases :	Participant, User, Learner
How and where used :	Attend and watch the class activities
Format :	Data of Participant

Name :	Teacher
Aliases :	Presenter
How and where used :	Handle the classroom environment and gives the lecture
Format :	Data of Presenter

Name :	Raise Hand
Aliases :	Notify Teacher
How and where used :	When a students wants to answer a question or notify teacher about sth
Format :	Boolean value

Name :	Administrator
Aliases :	Manager
How and where used :	Deals with the applications and registrations of the students
Format :	Data of Admin

2.5.2 Database

2.5.2.1 ER Diagram

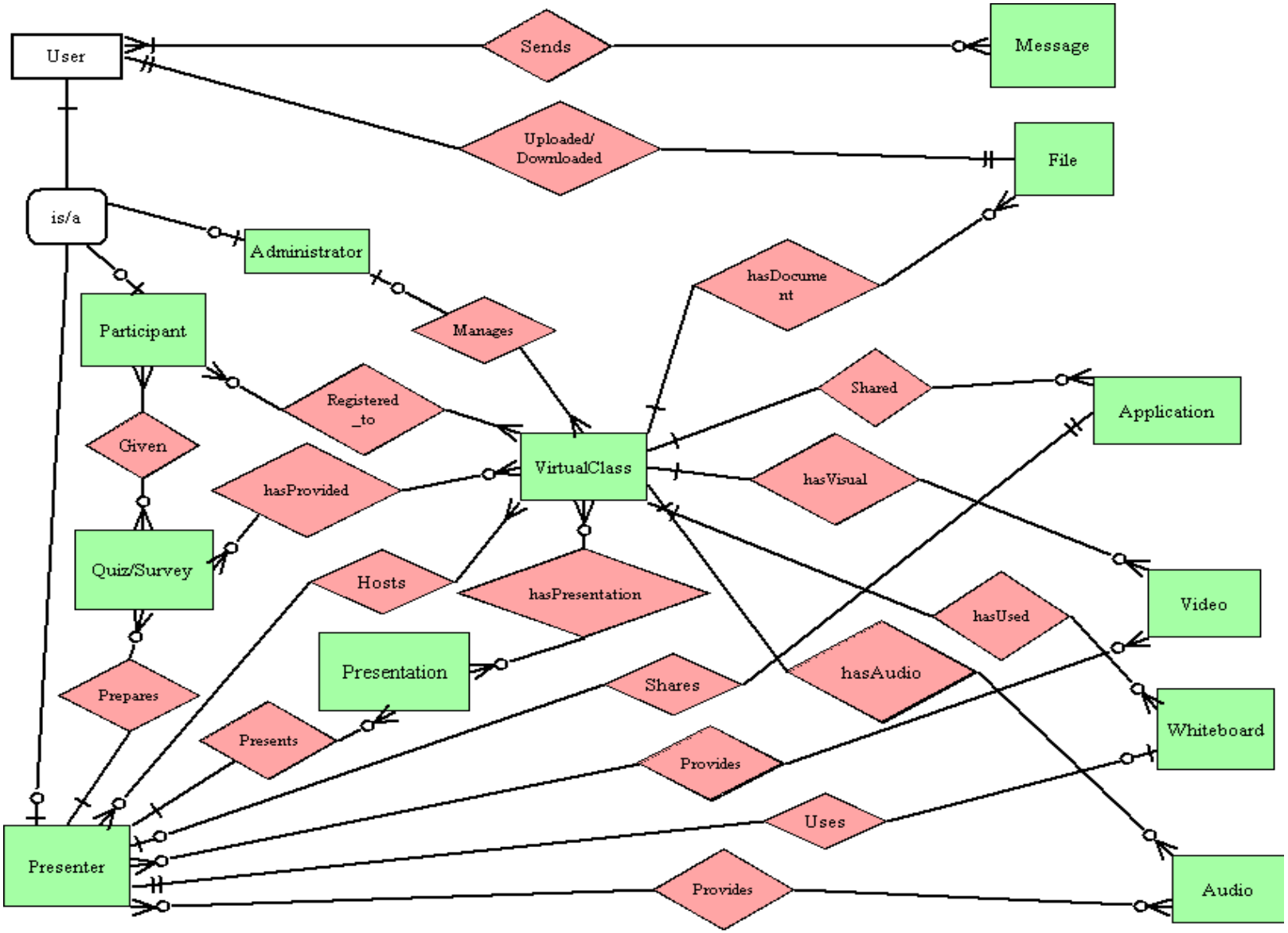


Figure 37 Entity Relationship Diagram

The ER Diagram of the database on the server is given in Figure 37. The relations between entities and their multiplicities are given taking into consideration their use and functionalities.

The following section gives detailed information about each entity together with its attributes and its initial table description.

2.5.2.2 Table Descriptions

2.5.2.2.1 User

This table holds personal data of the users. The attributes are quite clear. hasCam and ghasMicrophone is used to detect whether the user has a Camera or Microphone respectively. Moreover, it would be better to hold the location (a system URI) of the picture instead of holding the picture itself. The username (sometimes referred to as userID as well) is unique for each user. Therefore it is the key of the table.

Attribute Name	***Type***
username	String
password	String
name	String
surname	String
e-mail	String
phone	String
type	Integer
gender	Integer
hasCam	Boolean
hasMicrophone	Boolean
pictureLocation	String

Table 1 User Table

2.5.2.2.2 Message

A message is every packet send for chatting. A unique identifier is the messageID. Moreover, the logID is kept to identify messages in the same session (private or public). This can facilitate to get a whole conversation. Finally, the formattedMessage represents the message send in a formatted manner. Such a format should enable using Fonts, Emoticons and other formats.

Attribute Name	***Type***
messageID	String
logID	String
senderID	String
receiverID	String
time	Date
date	Date
formattedMessage	String

Table 2 Message Table

2.5.2.2.3 File

Each uploaded file is recorded in this table. The permission is an integer identifying the permissions granted to different user groups. This permission format will be elaborated later on in more detail.

Attribute Name	***Type***
FileID	String
Filename	String
fileLocation	String
SourceID	String
Size	Integer
Extension	String
Permission	Integer

Table 3 File Table

2.5.2.2.4 Application

Every shared Application is saved with a unique identifier. An animation format will be stored to enable future access to and use of the application sharing session.

Attribute Name	***Type***
applicationID	String
Name	String
ProviderID	String
applicationAnimationLocation	String

Table 4 Application Table

2.5.2.2.5 Video

Videos will be stored in the file system as a whole. The table is used for referencing the files and identifying them.

Attribute Name	***Type***
VideoID	String
Name	String
Length	Integer
Fps	Integer
SourceID	String
fileLocation	String

Table 5 Video Table

2.5.2.2.6 Whiteboard

Like all of the other media types, the whiteboards should also be stored within the database. As a whiteboard can have multiple pages, it is stored under a directory as a whole, with an identifying sequence number naming convention (to be described in detail later).

Attribute Name	*Type*
whiteboardID	String
OwnerID	String
Width	Integer
Height	Integer
numberOfPages	Integer
directoryLocation	String

Table 6 Whiteboard Table

2.5.2.2.7 Audio

The audio stream is similar to the Video. A unique identifier is assigned again.

Attribute Name	*Type*
AudioID	String
Name	String
Length	Integer
fileLocation	String
Bps	Integer
source	String

Table 7 Audio Table

2.5.2.2.8 Presentation

Although a presentation can be considered as a raw file, this table provides information about its class usage as well. An animation (in pps or gif format) is also referenced to replay the presentation later.

Attribute Name	*Type*
presentationID	String
numberOfSlides	Integer
fileLocation	String
SourceID	String
animatedFilename	String

Table 8 Presentation Table

2.5.2.2.9 Quiz/Survey

As quizzes or surveys can be given by the presenters, this should also be stored in a database. A presenter will be able to prepare a quiz beforehand or instantaneously. It can have different formats like n-multiple-choice or classical questions. Statistics are collected for each answer and quiz or survey to be displayed for informative purposes. As can be seen in the overall database ER diagram, this table has three relations: with the VirtualClass, the presenter, and the participants.

Attribute Name	*Type*
ID	String
Name	String
timeGiven	Date
expiryTime	Date
numberOfQuestions	Integer
Type	Integer
QuestionNumber	Integer
Choices	String
Answer	String
percentageStatistic	Double

Table 9 Quiz/Survey Table

2.5.2.2.10 Participant

As a participant is different from a presenter some types of information just related to the participant are stored in this table. The hand-rise status is one of them.

Attribute Name	*Type*
currentClass	String
ID	String
Status	Integer
displayPicture	Boolean
displayInformation	Boolean
Permission	Integer
Handrise	Boolean

Table 10 Participant Table

2.5.2.2.11 Presenter

The presenter table stores information about another specific type of user. This information is very dynamic in itself. The status and currentClass attributes will change frequently. The status field will have several different values which will be defined in the detailed design phase.

Attribute Name	*Type*
currentClass	String
ID	String
status	Integer

Table 11 Presenter Table

2.5.2.2.12 VirtualClass

Each VirtualClass as well as each of its sessions is also stored in the database. The VirtualClass has relations with all of the other tables. Note that, the attributes provided in this report only represent the individual tables and will change in the detailed design phase to create the tables according to the relations defined.

Attribute Name	*Type*
ClassID	String
SessionID	String
Name	String
startDate	Date
endDate	Date
Duration	Integer
maxNumberOfParticipants	Integer
Type	Integer
recurrencePattern	Integer

Table 12 Virtual Class Table

2.5.2.2.13 Administrator

An administrator is another type of specific user. The status and type is stored in this table. Here the type is a level of administration of the system.

Attribute Name	*Type*
Status	Integer
Type	Integer
ID	String

Table 13 Administrator Table

2.6 Graphical User Interface Design

This section describes the design and the key points considered during the process of designing the Graphical User Interface. Whatever the functionality of a system be, it is crucial to have a good Graphical User Interface (GUI).

The GUI is an MDI Interface to enable multiple Window Forms (see Figure 38). This will facilitate the integration of the system. Each module is a separate Window Form. Thus it is very easy to add or remove modules. Together with the SmartClient technology, which helps versioning of the system, does not require setup for applications and can lets it to be loaded and updated automatically from the internet, this Graphical User Interface structure will change from version to version and will help in developing the system incrementally.

The nature of independence of each table will facilitate the upgrade of a module and the main Graphical User Interface will enable the communication and synchronization of these modules.

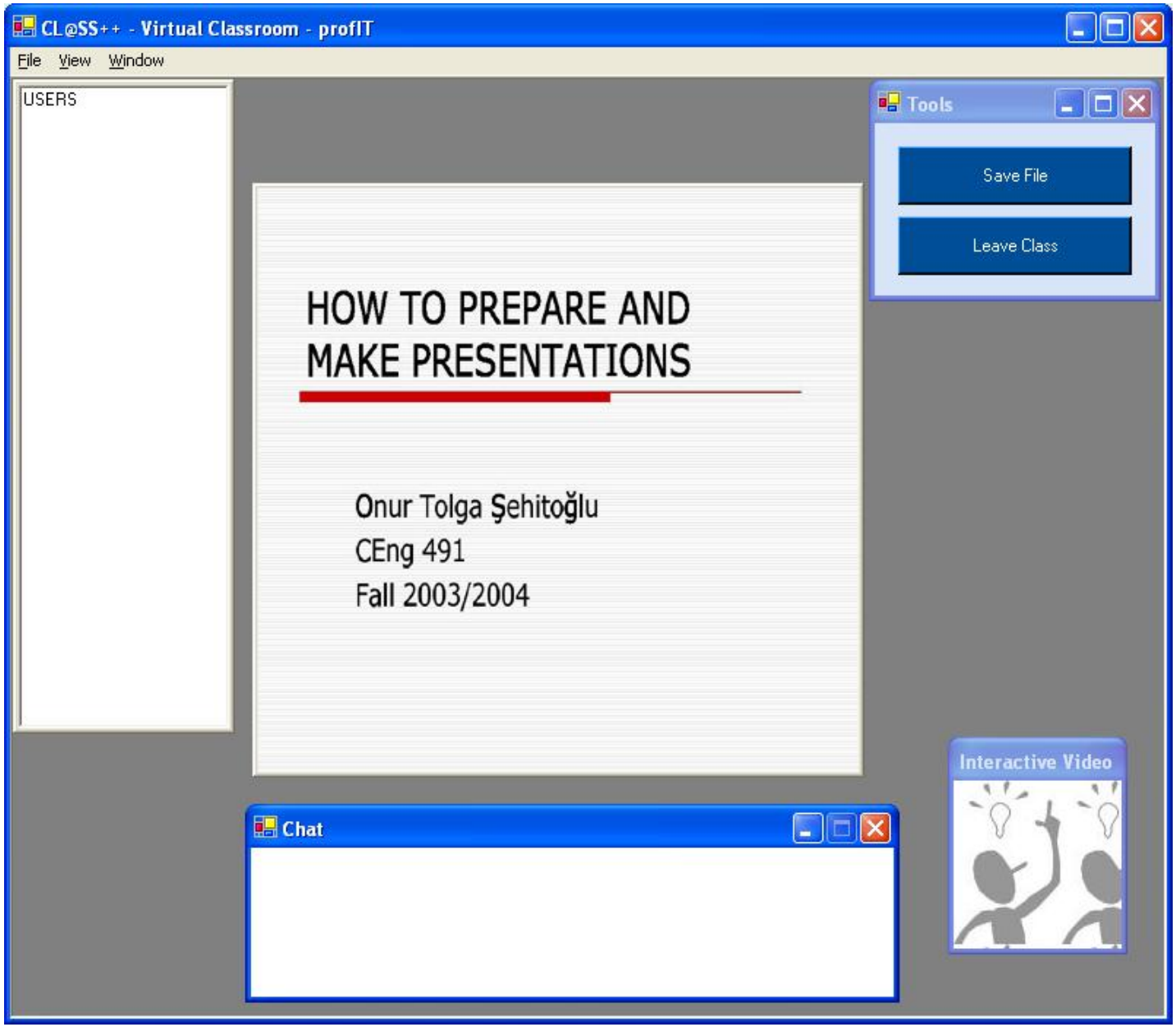


Figure 38 CL@SS++ Graphical User Interface

3 Versioning

Versioning is one of the key events in application development. During the progress of our project, the versions of the software represent the improvement and status of the project. We give version numbers of CL@SS++ according to modules in the project. These modules are roughly enumerated by observation of the development of the project. Modules are defined by looking at the project features that can be separated into packages. In other words modules are separable features of CL@SS++. We also make the versioning by increasing order of development.

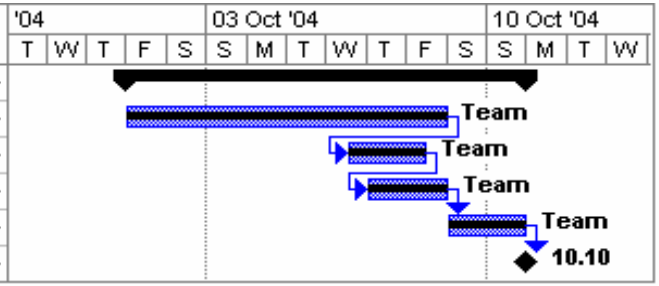
The main reason of developing such a versioning is to have a complete project at the end of each version. Complete means that this project can satisfy at least partly the requirements of the project. That is, the project team will conform to this versioning and in case the team cannot deliver all of the functionalities it will be able to deliver a version with the functionalities described below. If the project team accomplishes version 1.0.0 before the end of the term, new features or improvements can be considered for 2.0.0.

- Version 0.1.0
  - Socket / Ports
  - Connection
  - User Login
- Version 0.1.1
  - Initial GUI
- Version 0.2.0
  - Chat
  - Public Chat
  - Private Chat
- Version 0.2.1
  - DB connection / creation
- Version 0.3.0
  - File Transfer
  - File streams
  - File packets
- Version 0.3.1
  - File upload /download
- Version 0.4.0
  - Presentation
  - Archive Presentations
- Version 0.4.1
  - Synchronization
- Version 0.5.0
  - Audio Broadcast
  - Audio streaming
  - Alternative Audio Streaming Qualities
  - Audio Archiving
- Version 0.5.5
  - Speech to text Translation (only in English)
- Version 0.6.0
  - Quiz

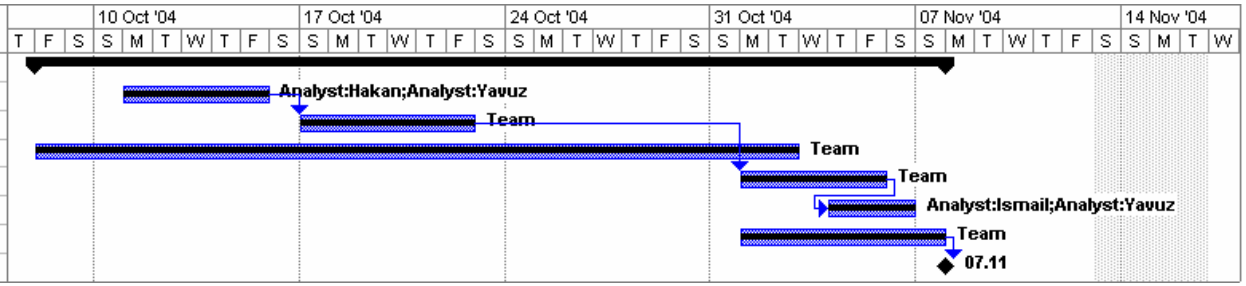
    - Survey
    - Virtual Class Outline
- Version 0.7.0
    - Application sharing
- Version 0.8.0
    - Interactive Whiteboard
- Version 0.9.0
    - Broadcast
    - Video streaming
    - Alternative Video Streaming qualities
    - Video archiving
- Version 1.0.0
    - Conferencing
        - Single presenter conferencing
        - Multi presenter conferencing
    - Multi-way conferencing
    - Alternative connection qualities support
- Version 2.0.0
    - Integration of new features
    - Improvement of the system

4 Project Schedule

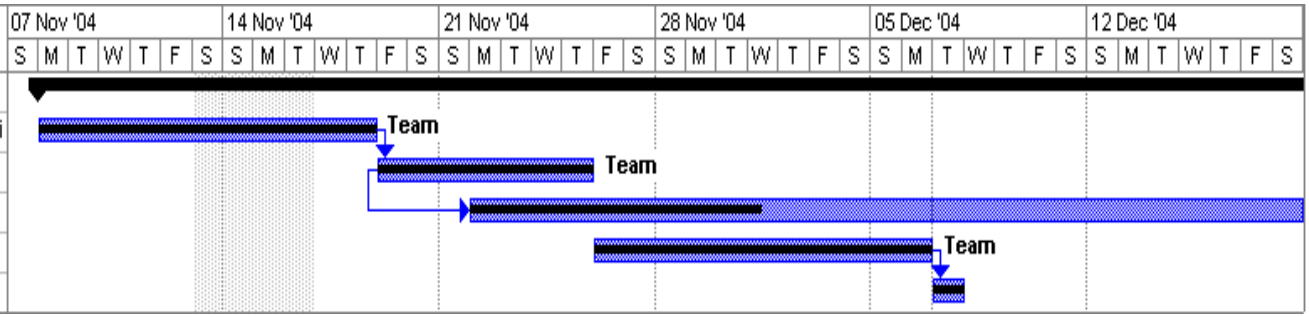
ID	Task Name	Duration	Start	Finish
1	**Proposal**	**10 days**	**Fri 01.10.04**	**Sun 10.10.04**
2	Determine project scope	6 days	Fri 01.10.04	Fri 08.10.04
3	Define preliminary resources	2 days	Wed 06.10.04	Fri 08.10.04
4	Obtain core resources	2 days	Thu 07.10.04	Fri 08.10.04
5	Finalize Project Proposal Report	1 day	Sat 09.10.04	Sun 10.10.04
6	Proposal Complete	0 days	Sun 10.10.04	Sun 10.10.04

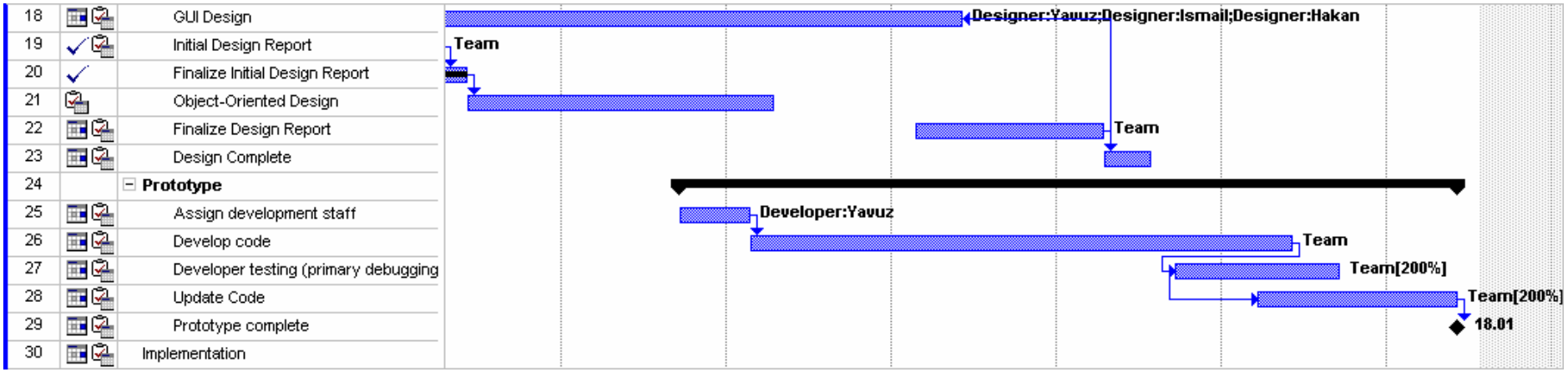


ID	Task Name
7	**Requirements Analysis**
8	Draft preliminary software requirements
9	Review software requirements
10	Existing Systems Analysis
11	Conduct Interviews
12	Incorporate feedback on software specifications
13	Finalize Requirements Analysis Report
14	Requirements Analysis complete



ID	Task Name
15	**Design**
16	Gather information about Environment and Appli
17	Develop functional specifications
18	GUI Design
19	Initial Design Report
20	Finalize Initial Design Report




18 GUI Design
19 Initial Design Report
20 Finalize Initial Design Report
21 Object-Oriented Design
22 Finalize Design Report
23 Design Complete
24 Prototype
25 Assign development staff
26 Develop code
27 Developer testing (primary debugging
28 Update Code
29 Prototype complete
30 Implementation
Designer:Yavuz;Designer:Ismail;Designer:Hakan
Team
Team
Developer:Yavuz
Team
Team[200%]
Team[200%]
18.01

5 Conclusion

In this report we tried to give the system modules in detail with the system progress. For that purpose we used data collaboration and class diagrams. At this phase of the project we analyzed the virtual class and partitioned the system into modules. We have also defined our database design and class structures with their collaborations between each other. Besides these, we have stated the versions of the project which will be helpful for constructing a well defined prototype and also the progress of the project. Finally with the help of our Gantt chart we tried to make a plan for the next phase of the project. We will move on to prototyping of the project for the next phase.