# Middle East Technical University
## Department of Computer Engineering

# Detailed Design
# for
# “CL@SS++”


profIT

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


**Fall 2004-2005**

# Table of Contents

# Index of Figures

# 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 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.

Since it is design report the architecture is developed as detailed as possible because this report will be the base for the progress of the implementation. In general, the purpose of the design is to satisfy student and teacher needs and supply them with a user-friendly environment with special properties that are not supplied by the similar virtual classroom tools.

## *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.

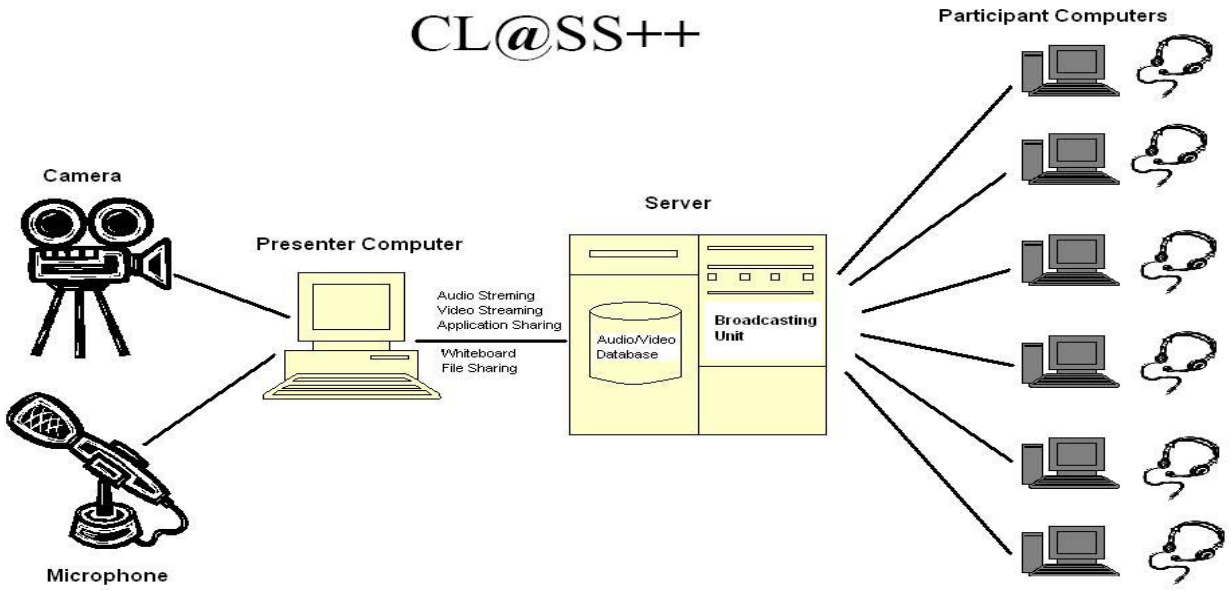


**Figure 1 Overall View of the System**

## *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

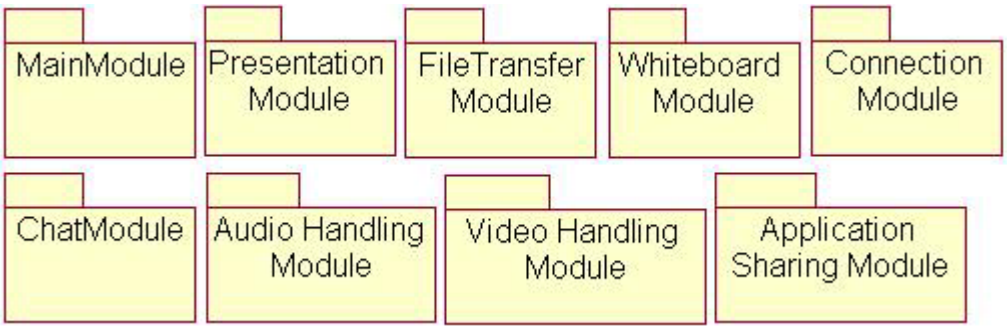


**Figure 2 Presenter Module Decomposition**

### 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.

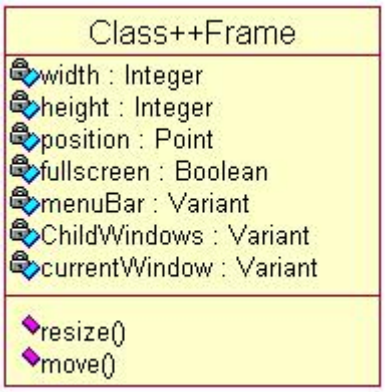


**Figure 3 CL@SS++ Main Module**

### 2.2.2 Connection Module

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

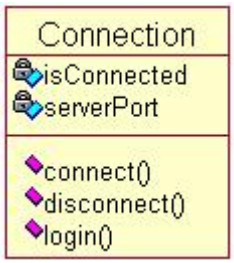


**Figure 4 Connection Module**

### 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.

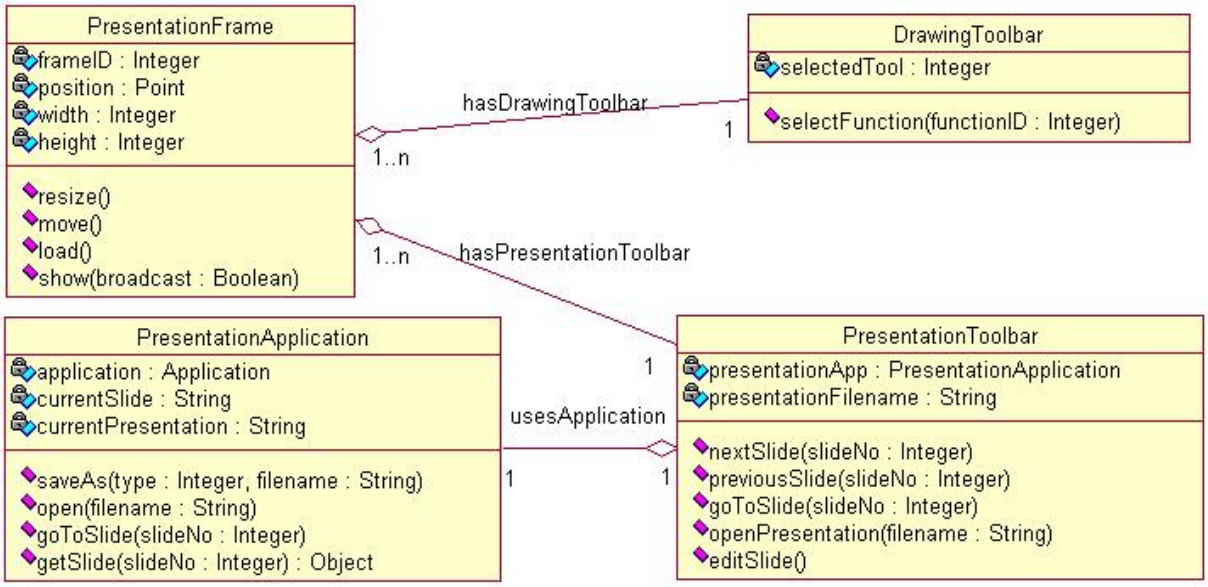


**Figure 5 Presentation Module**

## 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.

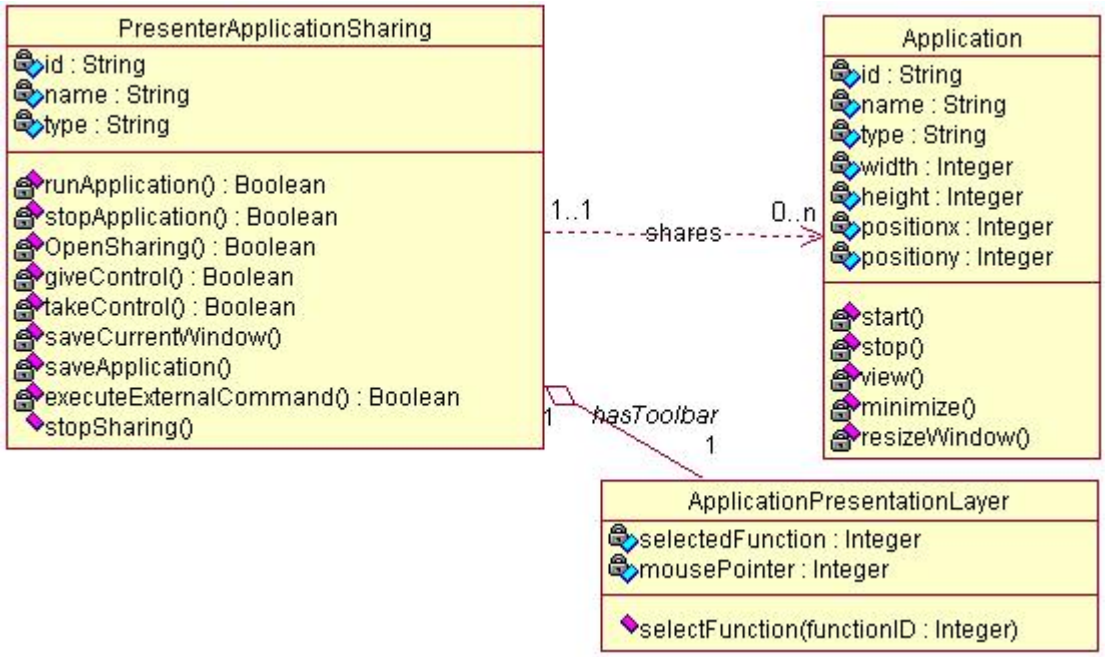


**Figure 6 Application Sharing Module**

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.

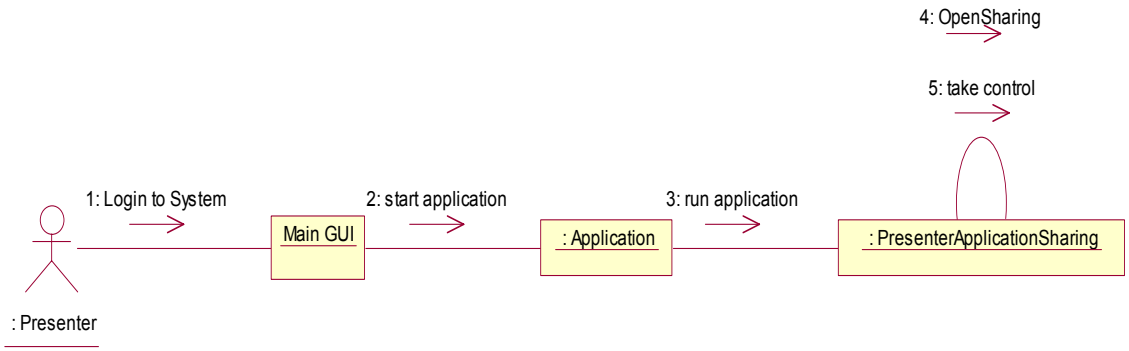


**Figure 7 Collaboration Diagram for Presenter Application Sharing**

### 2.2.5 Whiteboard Module

In the Whiteboard Module we have classes to implement Whiteboard facility. The classes for presenter usage are Board, PresenterBoard, ToolBox, and Palet (see Figure 8) and these will be included in the 'profit.vc.whiteboard' package

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. The editing methods use the GDI+ Libraries for drawing different shapes and writing text. It will also handle some utilities such as selecting, erasing, copying, undoing and so on. The operations have similarities with Ms. Paint like programs. Therefore we can use the Win API for some of the

operations. Also 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. There will be methods for changing the font type of the texts. The other methods in the class are for showing and hiding the toolbox and also most important one is the method for changing the type attribute. Also there is showPalet() method for displaying the Color dialog and starting the Palet Class.

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. We will use the Color Dialog of the Microsoft Windows for this task which will be opened through the Toolbox.

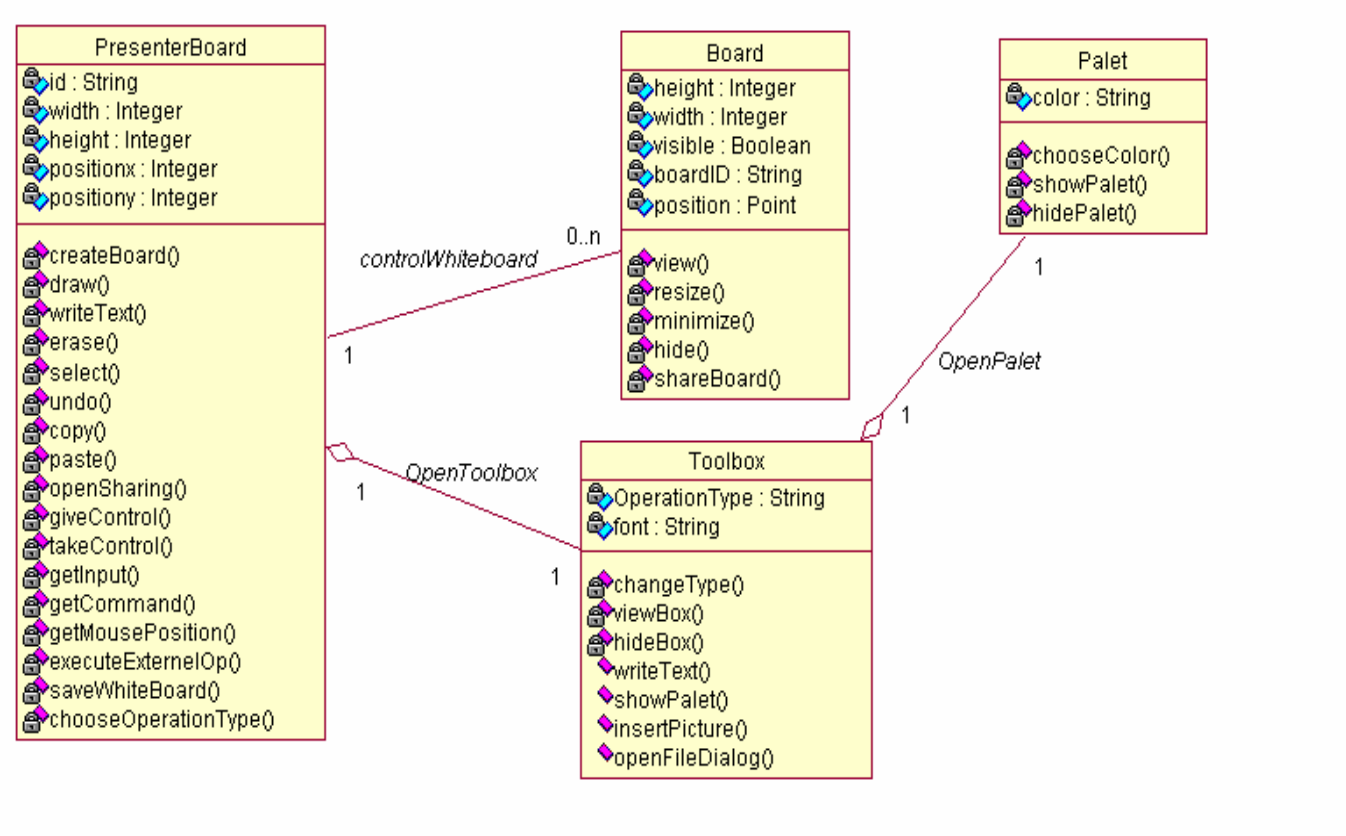


**Figure 8 Whiteboard Module**

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 PresenterBoard Class and shares it with all the users. The presenter can do all of the editing operations on this board such as drawing shapes and saving. 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 Class to

be able to use different edit options. And from this toolbox the presenter can open the Color Palet to change color options.

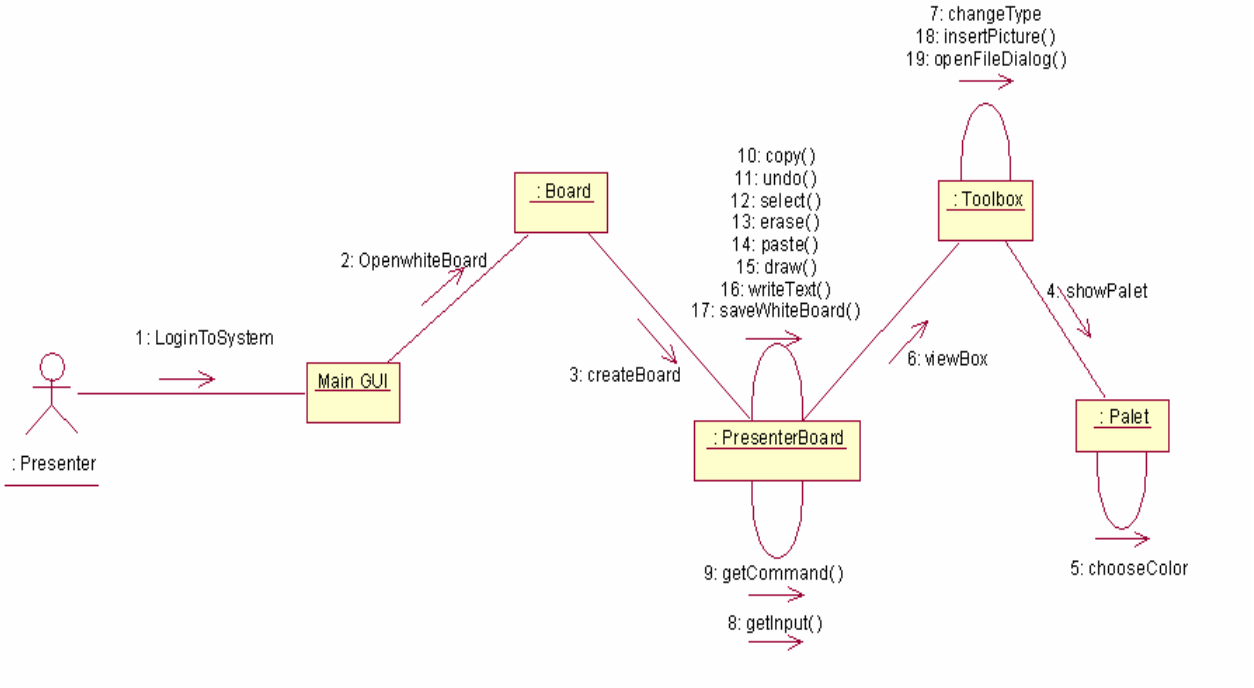


**Figure 9 Collaboration Diagram for Presenter Whiteboard**

### 2.2.6 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 32. 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.

The namespace of this module is “profit.vc.chat”.

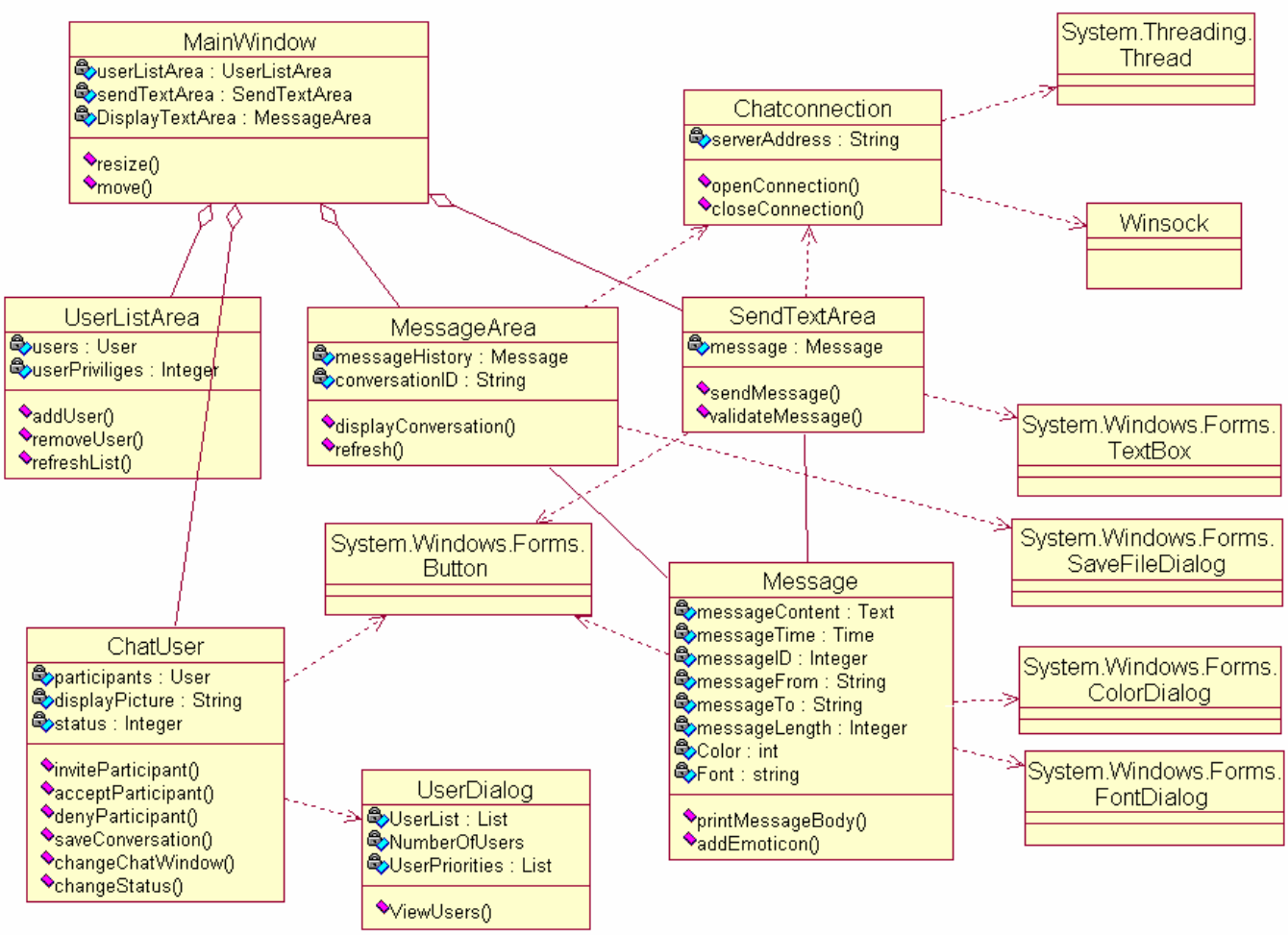


**Figure 10 Chat Module**

The presenter sees the main window and the MainWindow class of the module makes all of the initializations. Presenter can see the entire user by the UserListArea class. The Presenter can add or remove user with addUser and removeUser functions of that class. The chatUser class enables the Presenter chat with other Participants. The message is saved into with Message class. The saved message is transferred to the server side by making connection with the Chatconnection class. The connection is established with the System.Threading.Thread class and Winsock Library. 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. System.Window.Forms.Button class is used in many places in the user interface to enable the interaction with the user. The other dialog classes are used in relation with the System.Window.Forms.Button class. The message class is related with ColorDialog and FontDiolog classes because the message can have different color and font properties. Moreover the Textbox class is related with the SendTextArea class because the SendTextArea class retrieves the message from TextBox class.System.Windows.

Forms.SaveFileDialog is related with MessageArea class. When pressed on a save button the SaveFileDiolog class will be used and a file will be saved on to local hard drive.

The collaboration Diagram of this module is as follows;

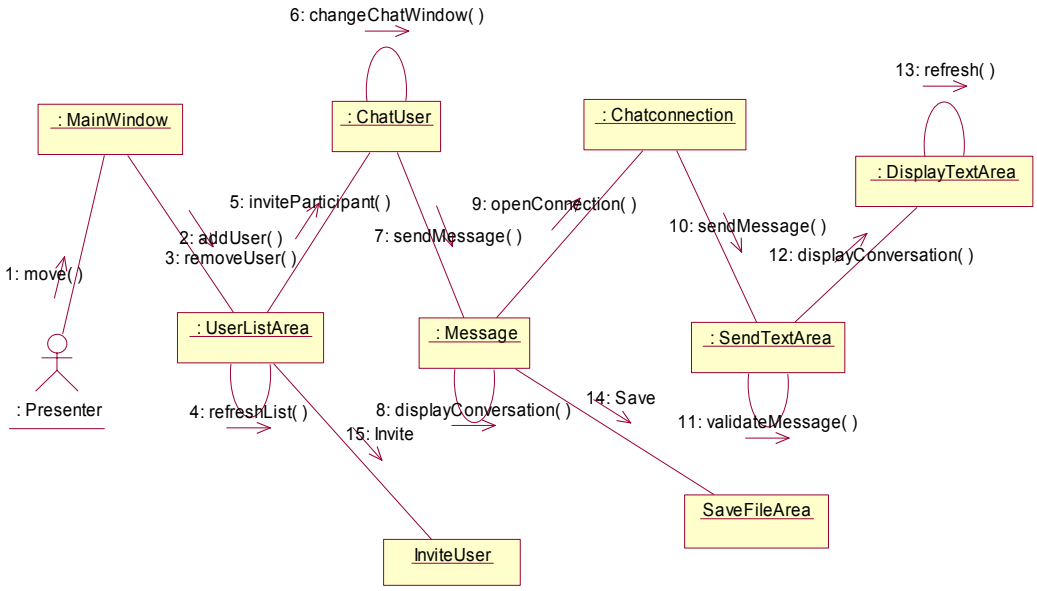


**Figure 11 Collaboration Diagram for Presenter Chat Module**

## 2.2.7 File Transfer Module

The class diagram for this module is in Figure 12. This module is responsible from sending files to the server for future use or sharing.

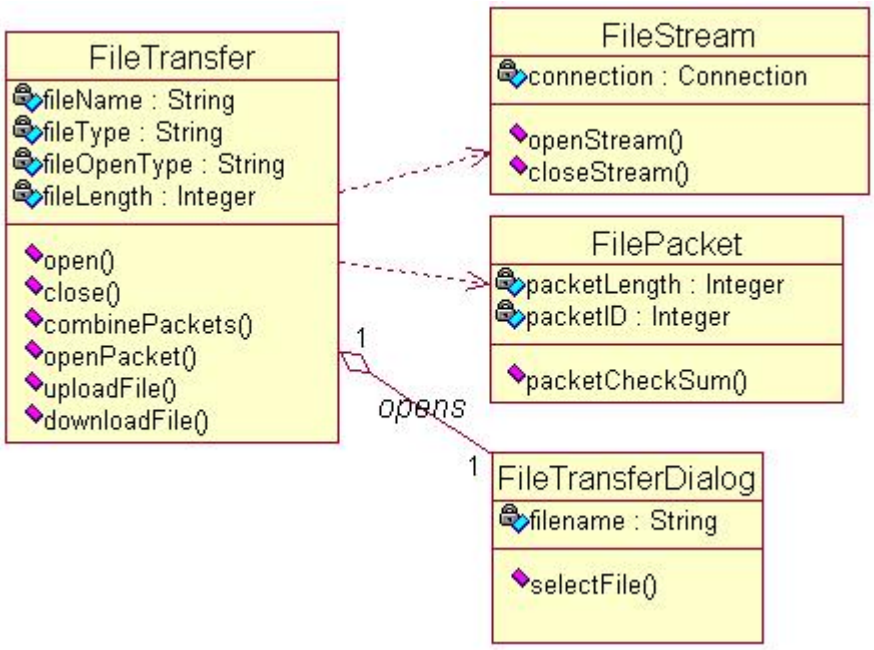


**Figure 12 File Transfer Module**

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 seperated into packets by the help of FilePacket class. Finally, the file is opened and combined by FileTransfer class. The Figure 13 below is representing that event visually.

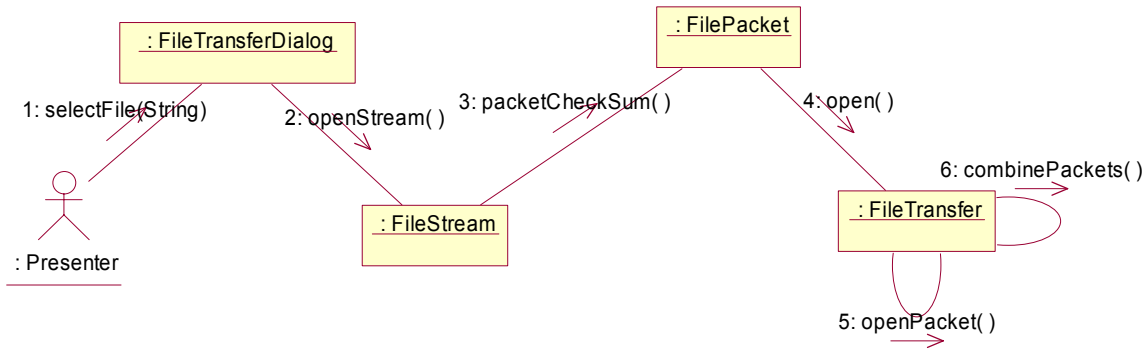


**Figure 13 Collaboration Diagram for presenter File Transfer**

### 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 14). These classes are VideoCapturer, VideoPackager, VideoCompress and VideoEncodeController.

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.

The last class is responsible from encoding control of the video since encoding video process involves complex computations and is very time consuming. The encoding process may not be completed within a pre-defined time. To avoid the buffer overflow or memory conflict, this unit is used to control the captured video data that stores in the buffer.

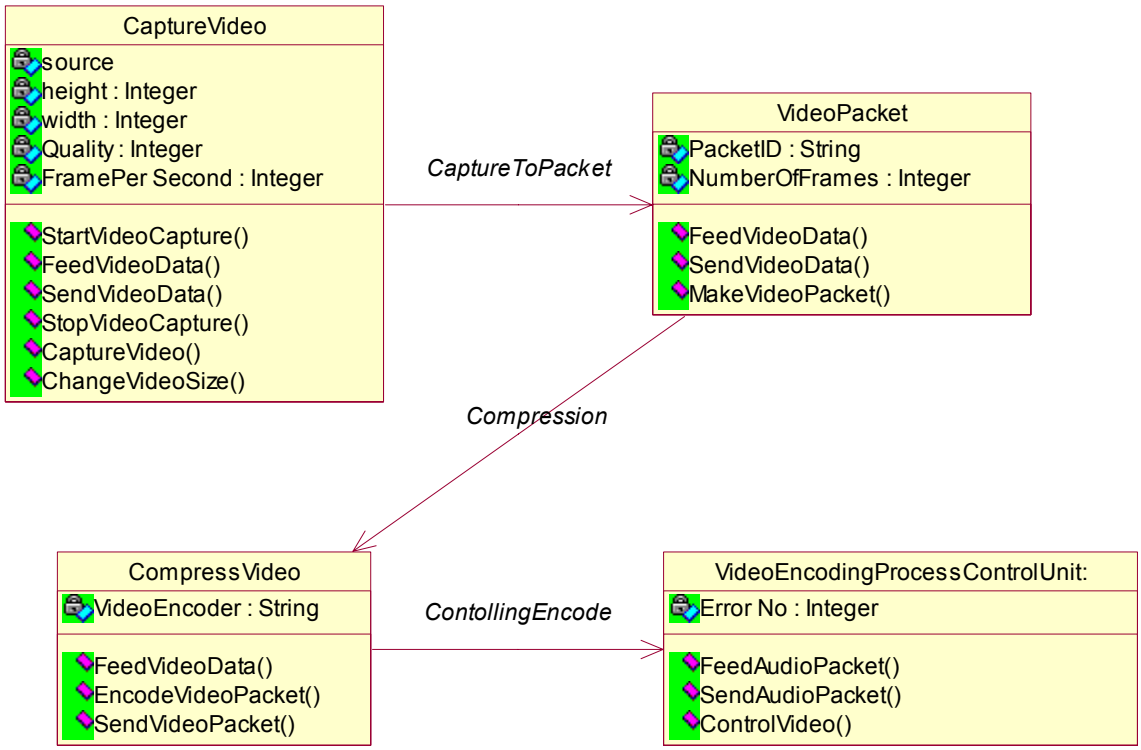


**Figure 14 Video Handling Module**

The collaboration diagram (see Figure 15) 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 EncodingProcessControlles in order to be controlled. After having controlled our video packet is sent to server.

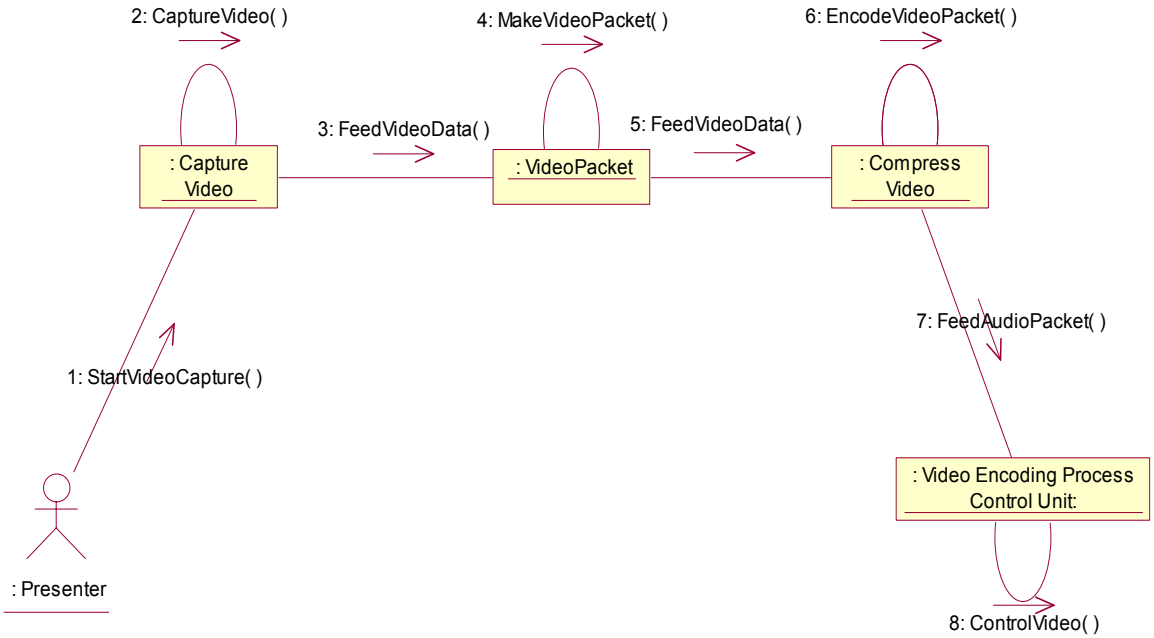


**Figure 15 Collaboration of Broadcast Video Presenter Side**

### 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.

The last class is responsible from encoding control of the audio since encoding audio process involves complex computations and is very time consuming. The encoding process may not be completed within a pre-defined time. To avoid the buffer overflow or memory conflict, this unit is used to control the captured audio data that stores in the buffer.

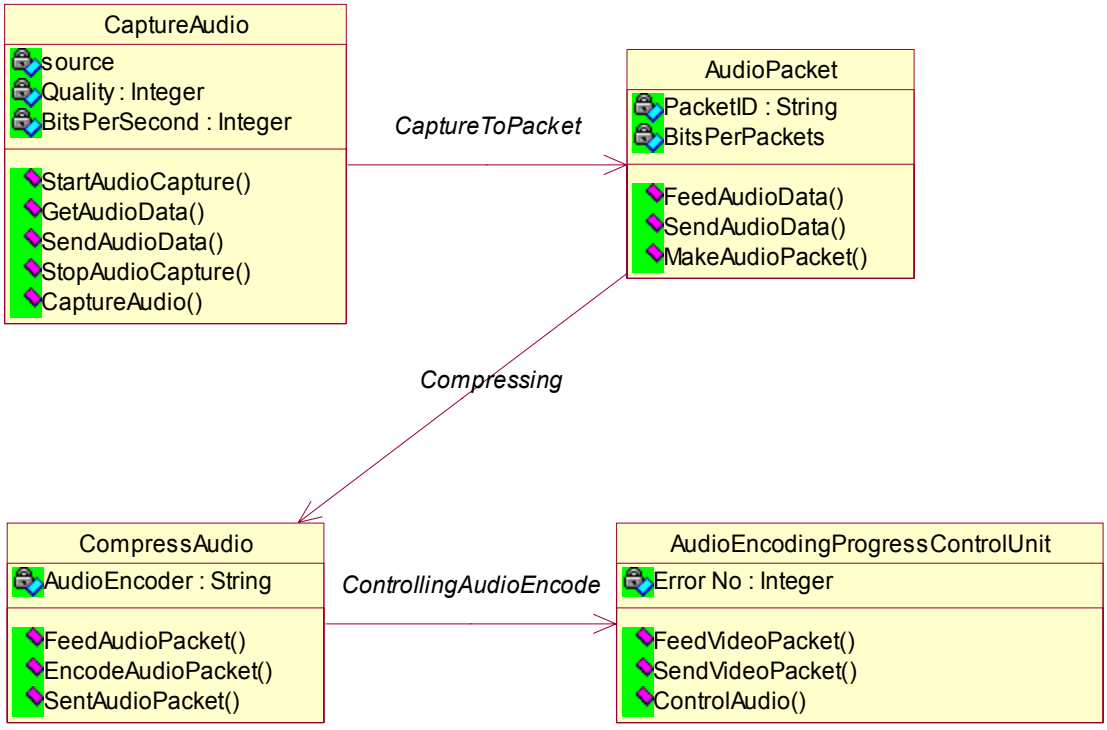


**Figure 16 Audio Handling Module**

The collaboration diagram (see **Figure 17**) 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 for controlling. The controller gets this data and sent it to the server after controlling it.

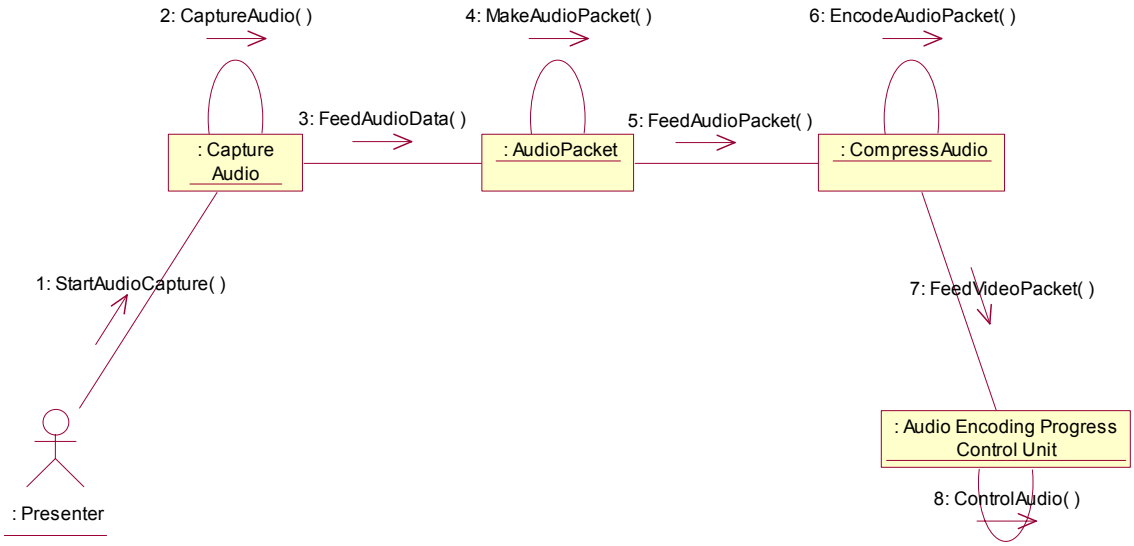


**Figure 17 Collaboration of Broadcast Audio Presenter Side**

## *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 18)
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.

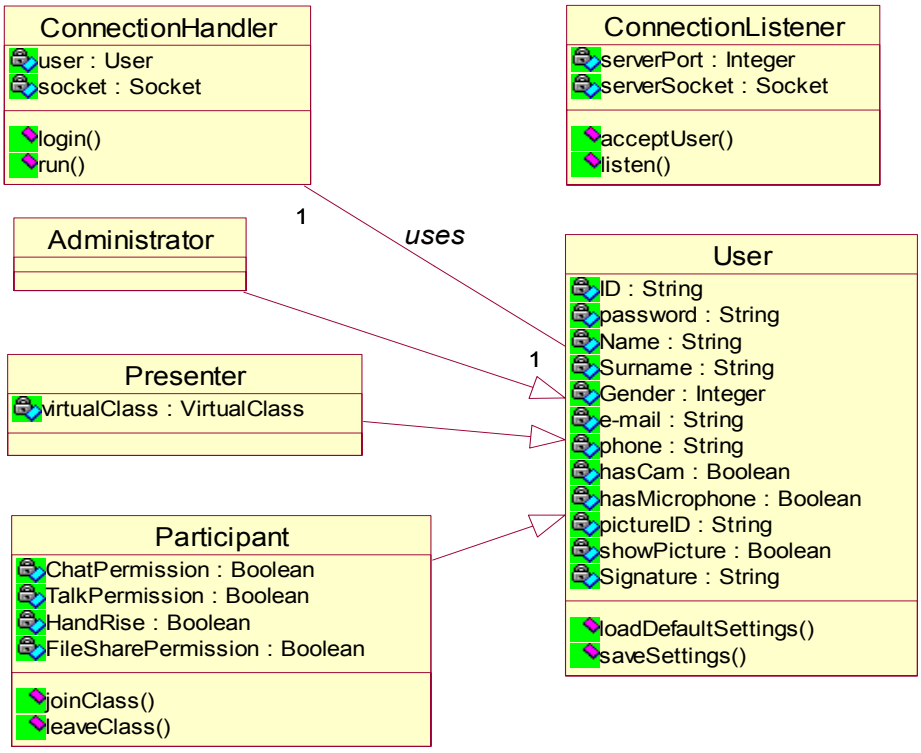


**Figure 18 Class Diagram of Connection Handler Module**

The Figure 19 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.

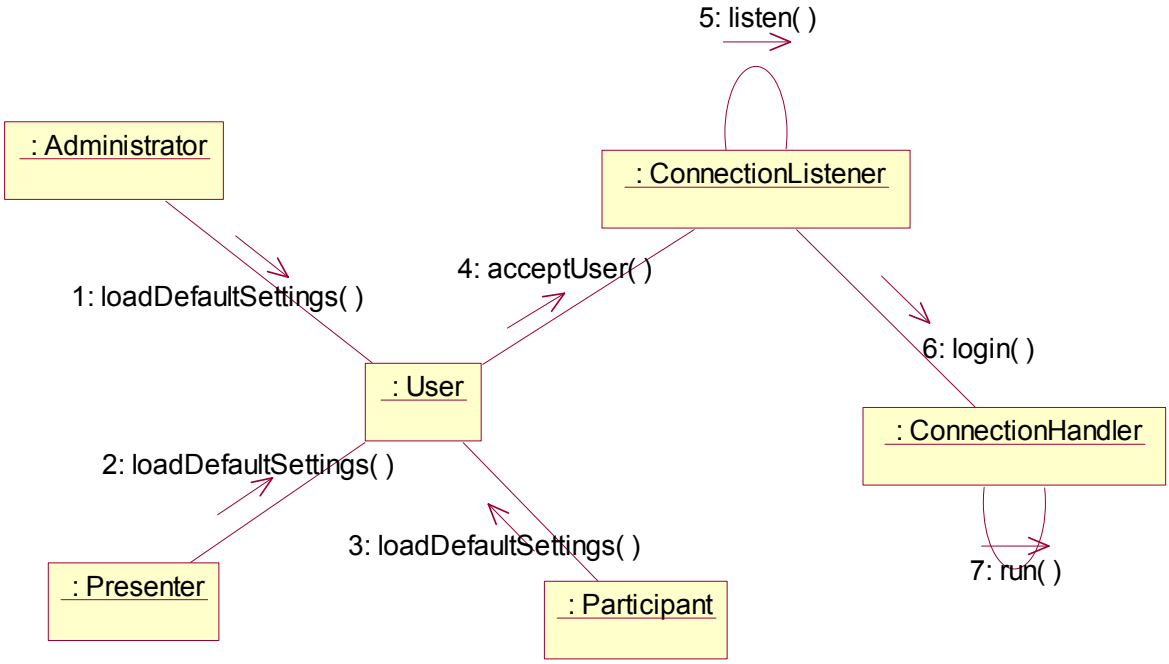


**Figure 19 Collaboration Diagram for Connection Handler Module**

### 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 20). 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 work 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.

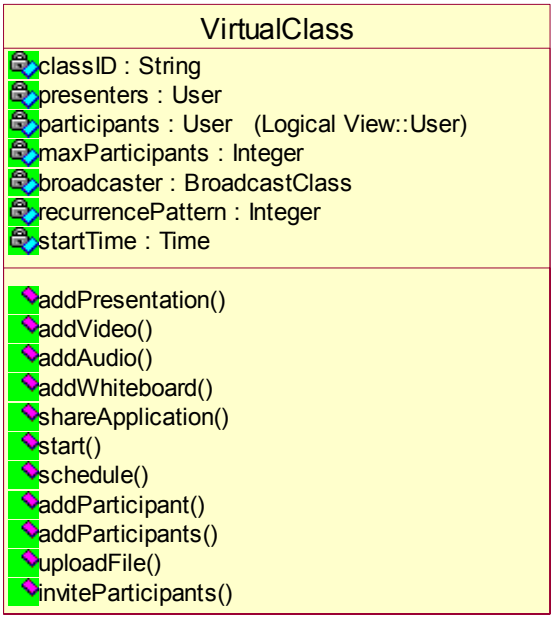


**Figure 20 Class Diagram of Virtual Class Module**

### 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 21). 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 checksthe state of the data.

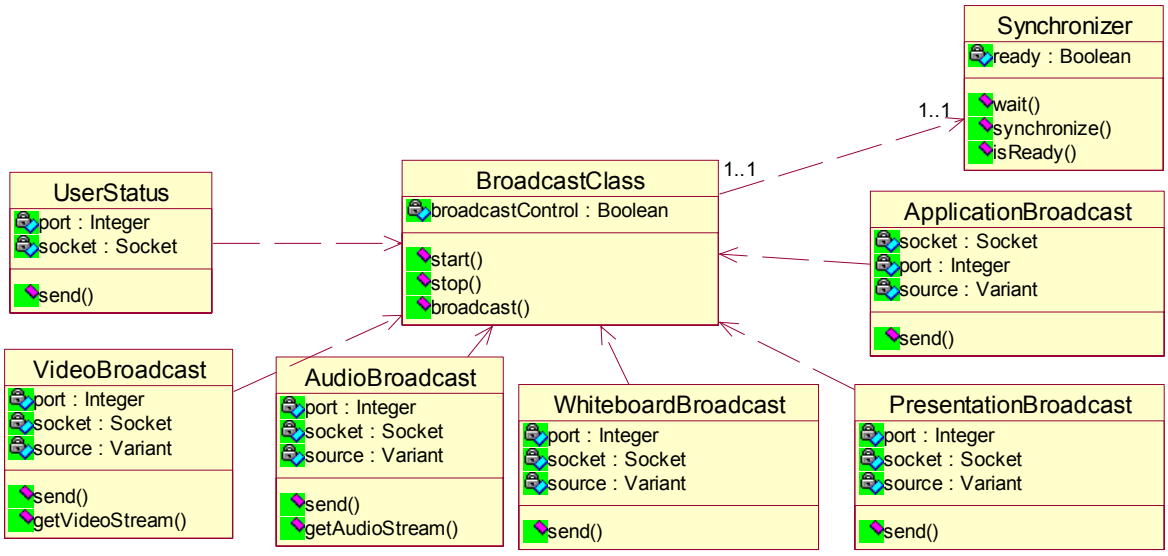


**Figure 21 Class Diagram of Synchronization and Broadcast Module**

The collaboration diagram (see Figure 22) 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.

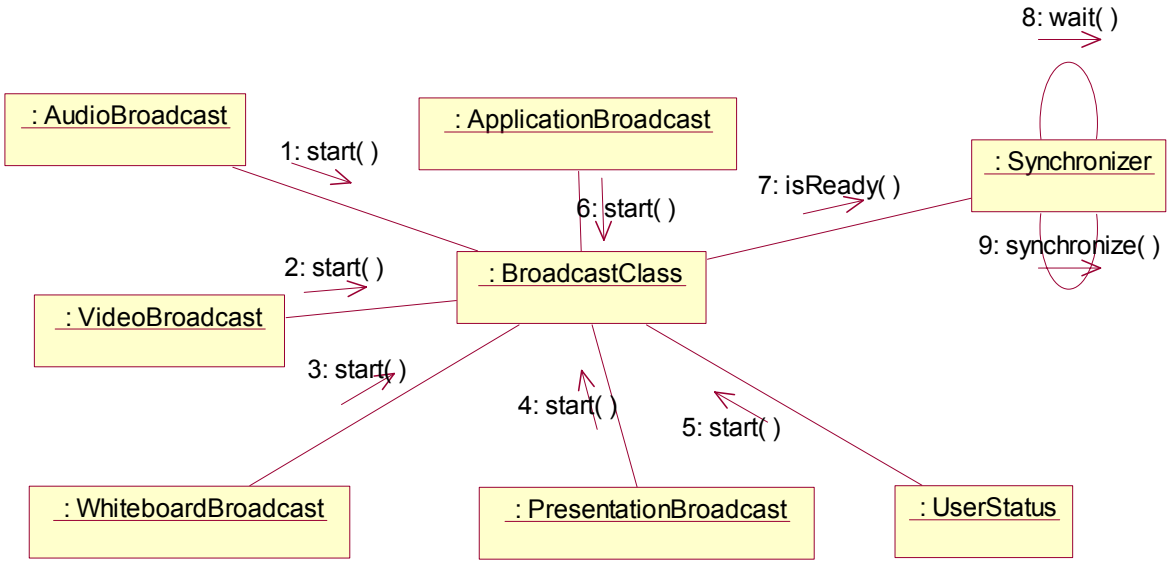


**Figure 22 Collaboration Diagram for Synchronization and Broadcast**

### 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 23

The DBConnection class is the class for the methods of database operations. The attributes of this class are hostname, portNumber, name, password, DBname, DBUsername, and DBUserpassword. This information is used when a database connection is to be started. The methods in this class 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

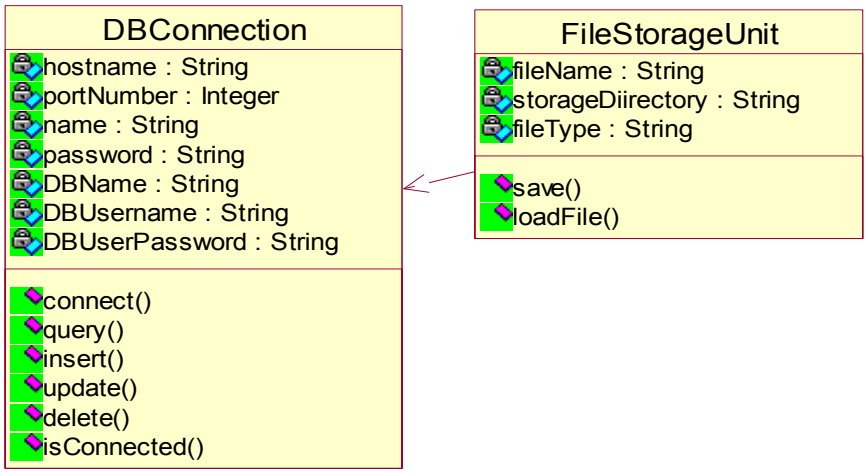


**Figure 23 Class Diagram for Database Connection and File Storage Unit**

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.

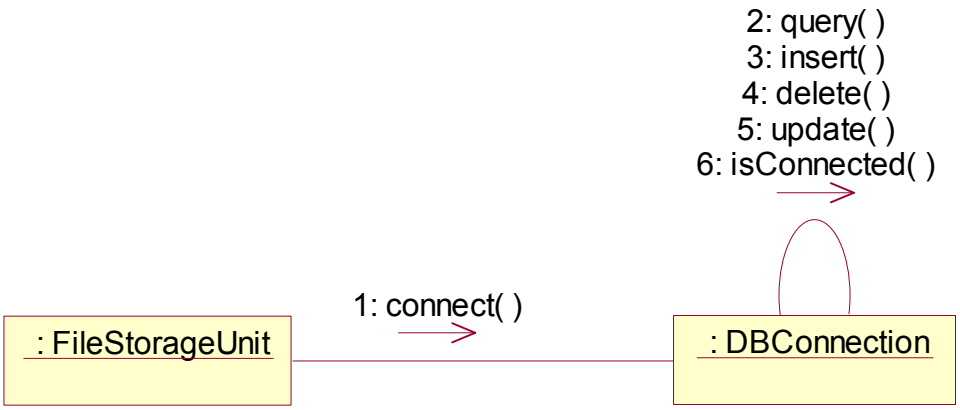


**Figure 24 Collaboration Diagram for Database Connection and File Storage Unit**

## *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.

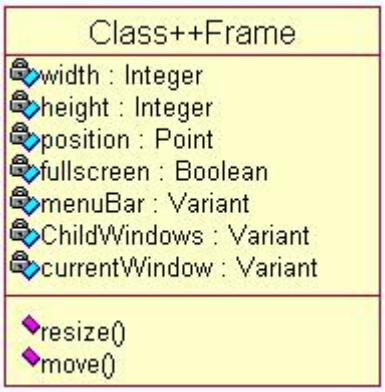


**Figure 25 CL@SS++ Presenter Main Module**

### 2.4.2 Connection Module

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

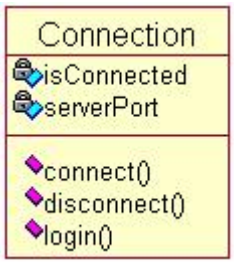


**Figure 26 Connection Module**

### 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.

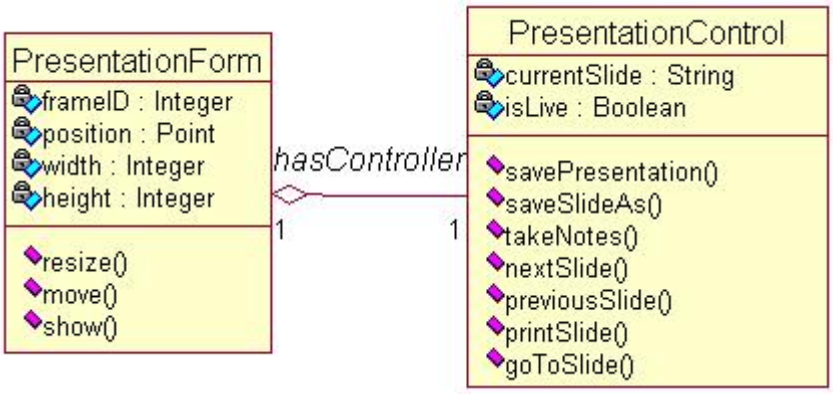


**Figure 27 Presentation Module**

### 2.4.4 Application Sharing Module

This module lets the participant share the application of the presenter (see Figure 28) and consists of 3 classes. These classes are detailed in the following paragraphs.

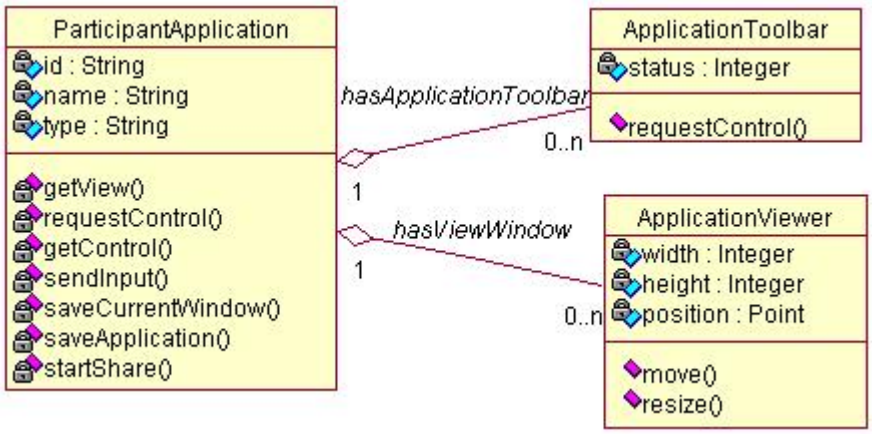


**Figure 28 Application Sharing Module**

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 29). 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.

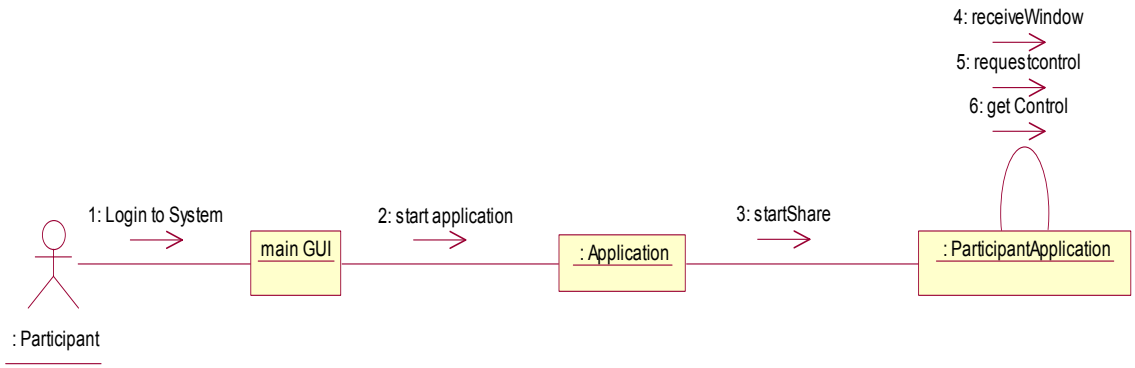


**Figure 29 Collaboration Diagram for Participant Application Sharing**

### 2.4.5 Whiteboard Module

This module lets the participant to view the Whiteboard that the presenter uses. This class is similar to the Presenter's side. This module has 3 classes (see Figure 30), namely ParticipantBoard, ParticipantToolbar, and BoardViewFrame. Again these classes will be in the package named 'profit.vc.whiteboard'.

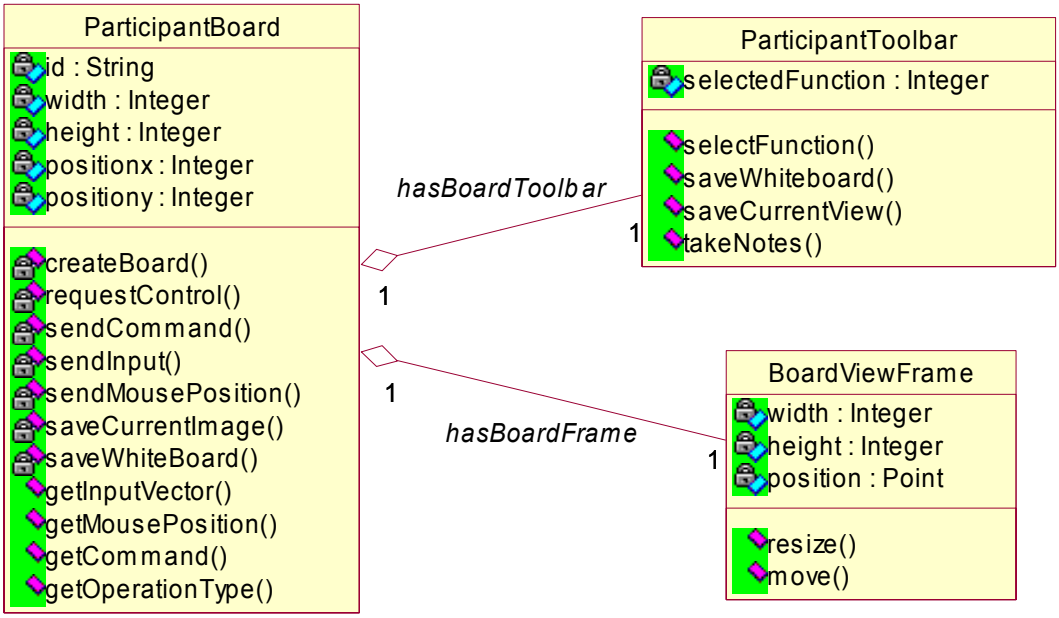


**Figure 30 Whiteboard Module**

The ParticipantBoard Class has the window attributes such as position and size as attributes. In this class there is a method to initialize the window. It has methods to get the inputs and operations done on the Presenter side and they are applied to the board of the Participants. 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. Also his changes and edits are sent to the other participants as they share the same board.

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 31). The Participant again enters the system from the login page. After login the Participant can initialize a board window to view the edits on the board. After opening the whiteboard the changes made by the Presenter are sent to the ParticipantBoard Class to be executed in the same way as in the Presenter's side. The view will be the same as the others. The inputs can be taken by the methods in the ParticipantBoard Class. Also the Participant can make request to take control from here. If s/he gets the permission, then he can make edits on the board. But these can be controlled by the Presenter.

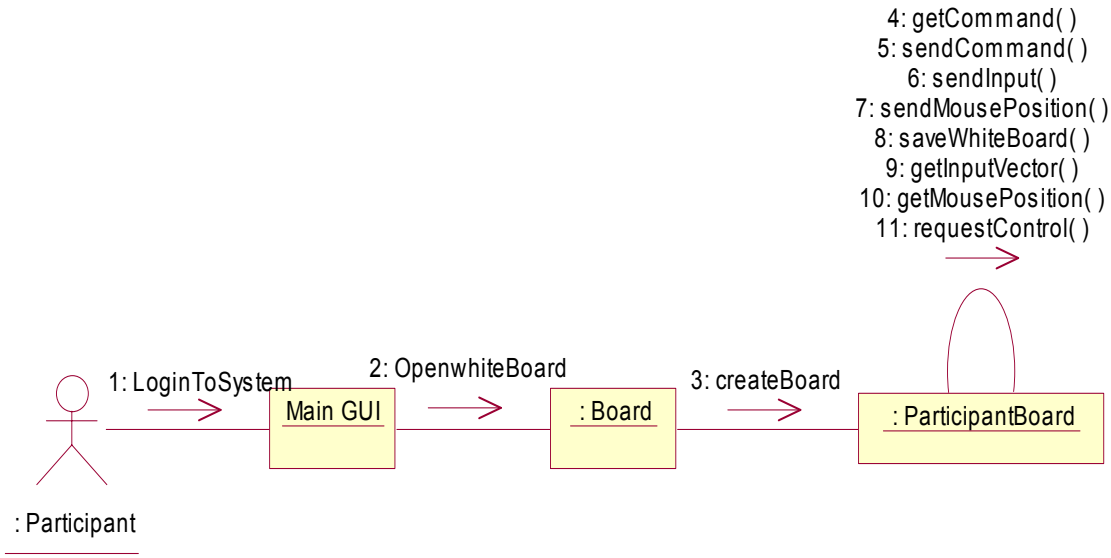


**Figure 31 Collaboration Diagram for Participant Whiteboard**

### 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 32. 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.

The namespace of this module is “profit.vc.chat”.

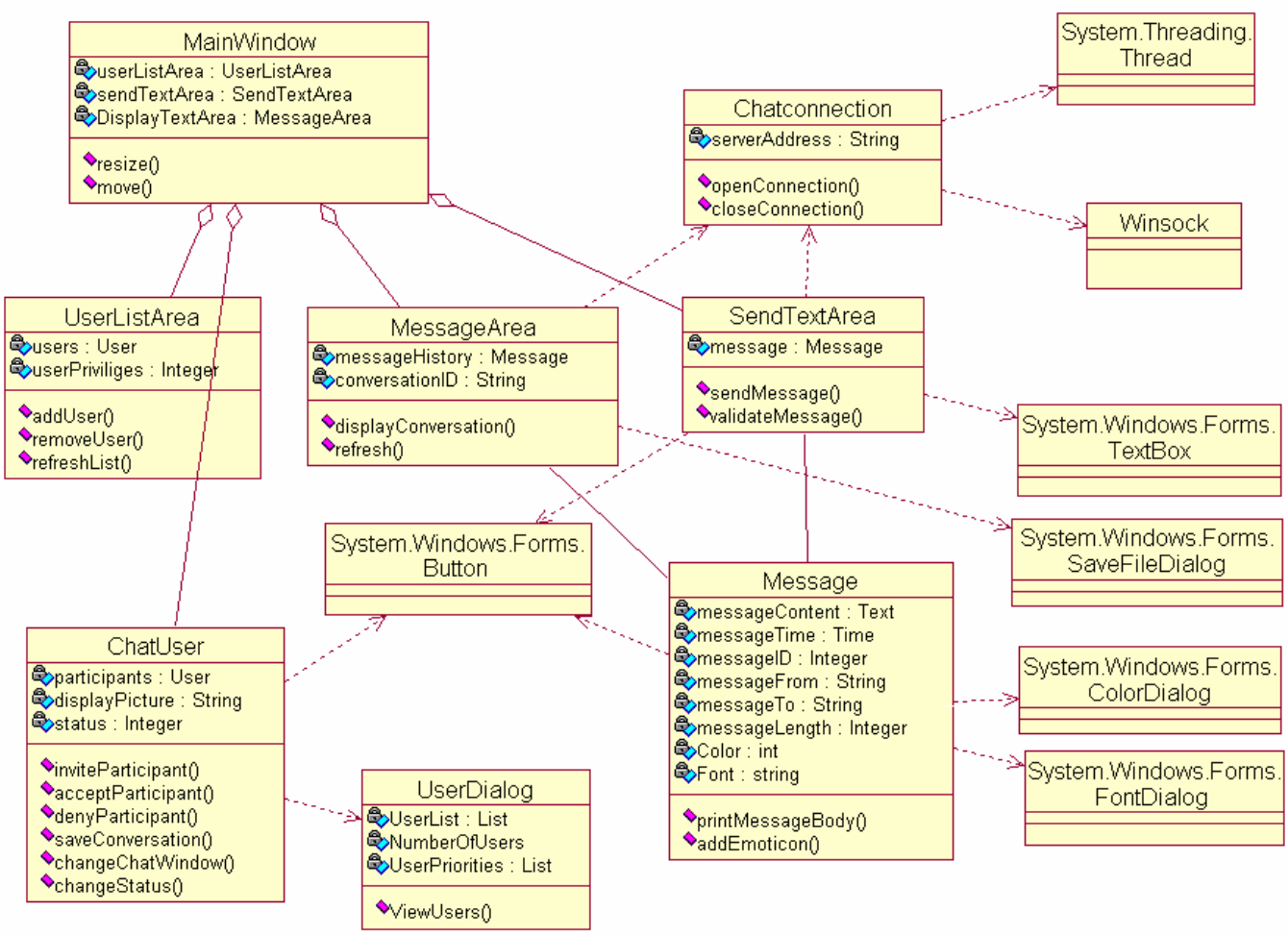


**Figure 32 Chat Module**

The Participant sees 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. The connection is established with the System.Threading.Thread class and Winsock Library. 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. System.Window.Forms.Button class is used in many places in the user interface to enable the interaction with the user. The other dialog classes are used in relation with the System.Window.Forms.Button class. The message class is related with ColorDialog and FontDiolog classes because the message can have different color and font properties. Moreover the Textbox class is related with the SendTextArea class because the SendTextArea class retrieves the message from TextBox class.System.Windows.

Forms.SaveFileDialog is related with MessageArea class. When pressed on a save button the SaveFileDiolog class will be used and a file will be saved on to local hard drive.

The collaboration Diagram of this module is as follows:

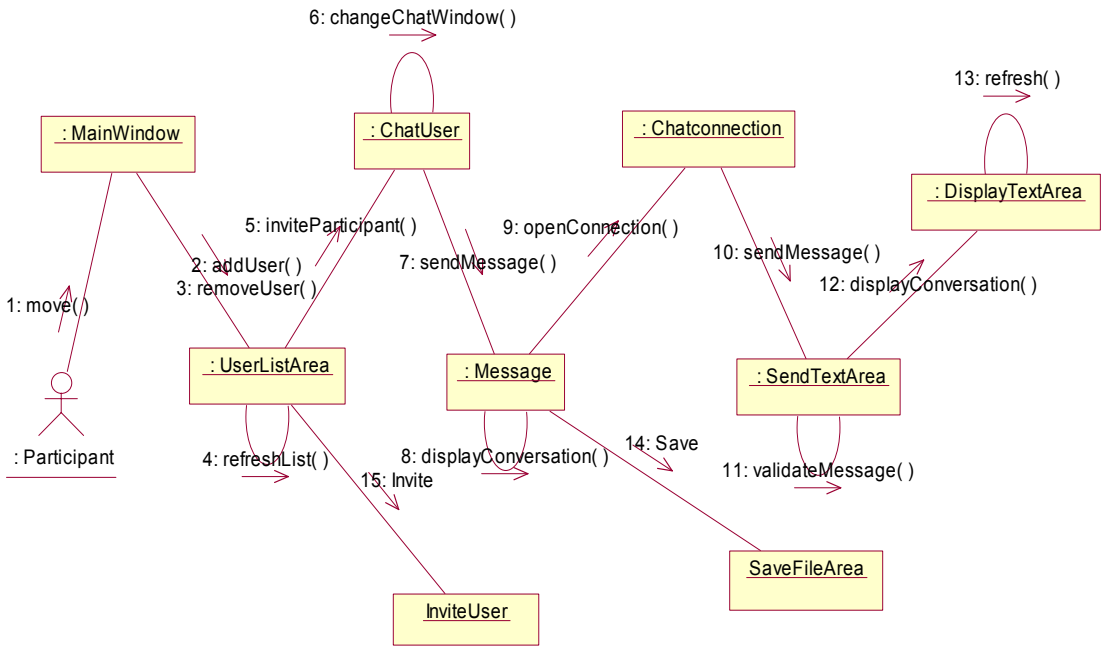


**Figure 33 Collaboration Diagram for Participant Chat Module**

## 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.

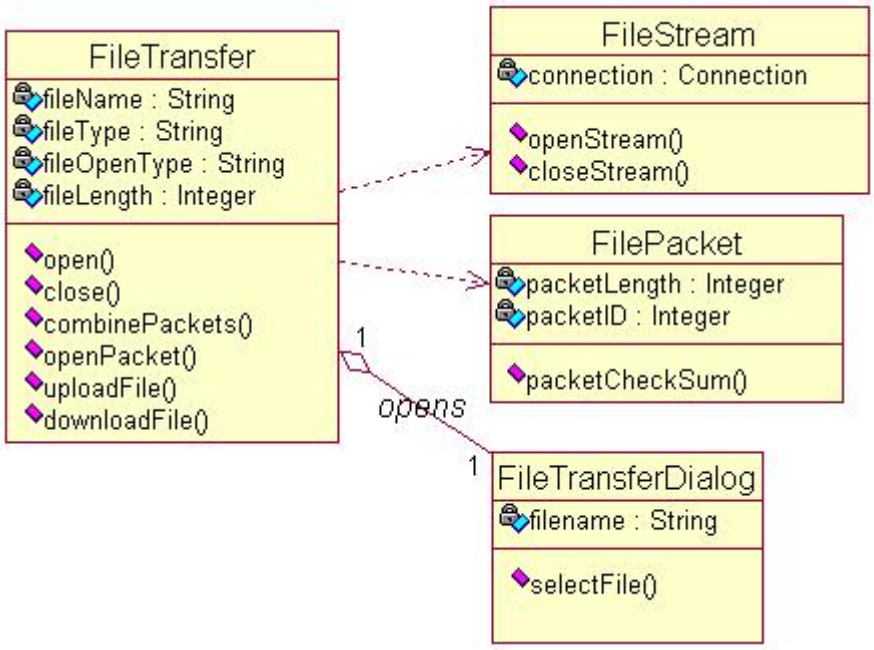


**Figure 34 File Transfer Module**

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.

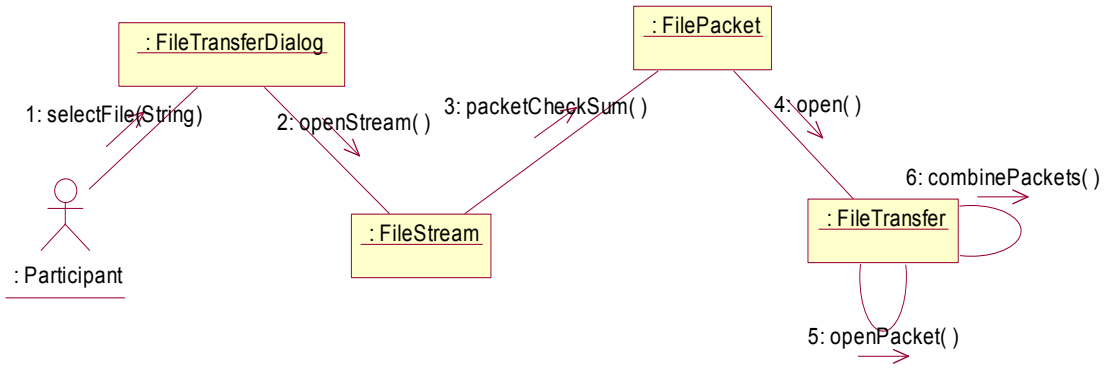


**Figure 35 Collaboration Diagram for Participant File Transfer**

## 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. The data transmitted from the server are decoded by video and audio decompression classes and the decoded data are played by the Play Video/Audio class. This class lets our participants to Play, Stop, Pause and Record the video and audio.

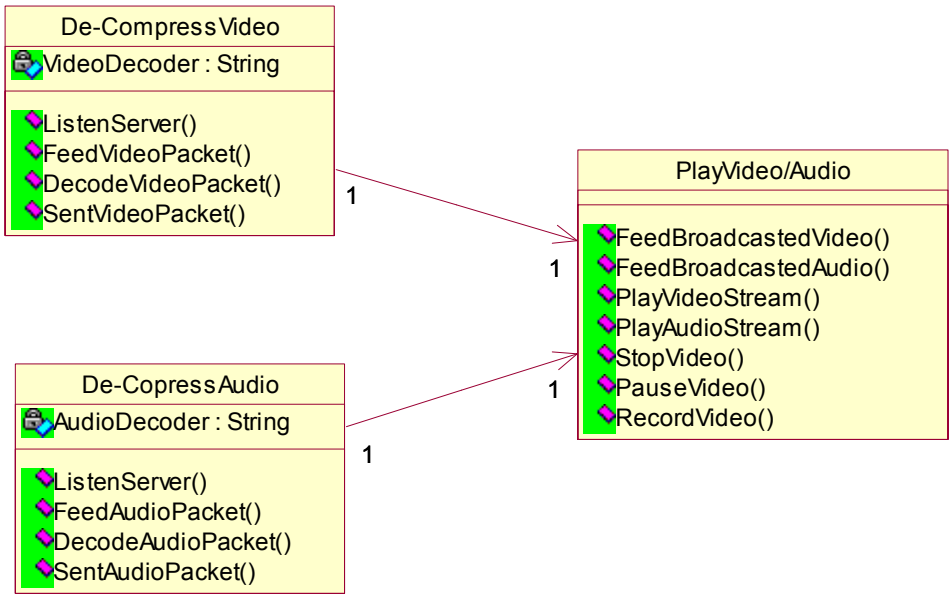


**Figure 36 Audio/Video Handling Module**

The Figure below shows our collaboration diagram for Audio/Video Handling in the participant side. Our decompression class listen the server for video and audio streams. Then after decomposition it feeds the decompressed audio and video streams to PlayVideo/Audio class. After getting the data from the de-compressor the PlayVideo/Audio class can do operations on the stream. For instance it can start playing video and stop it by using its PlayVideo and StopVideo functions respectively. It can also pause and record the video/audio streams.

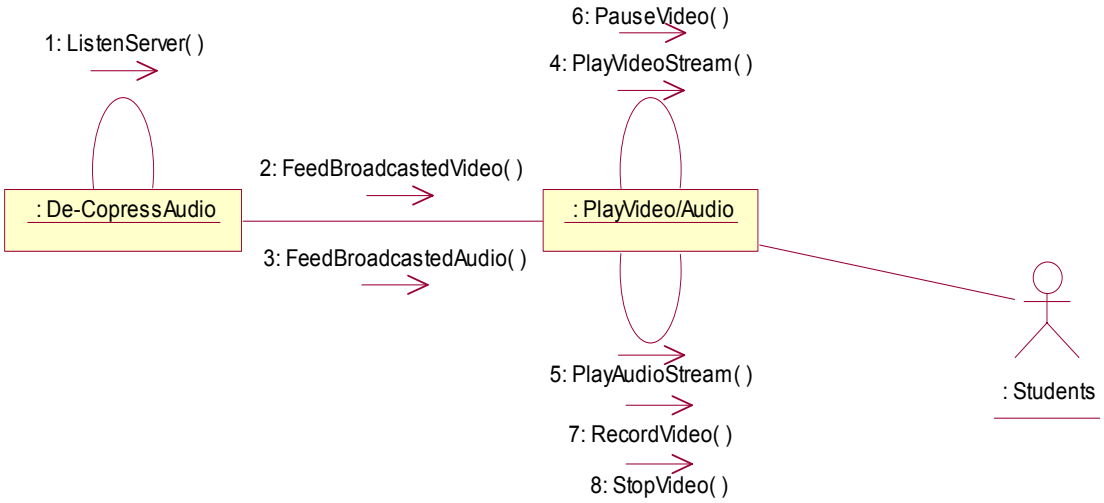


**Figure 37 Audio/Video Handling Collaboration Diagram**

## *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 something |
| 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

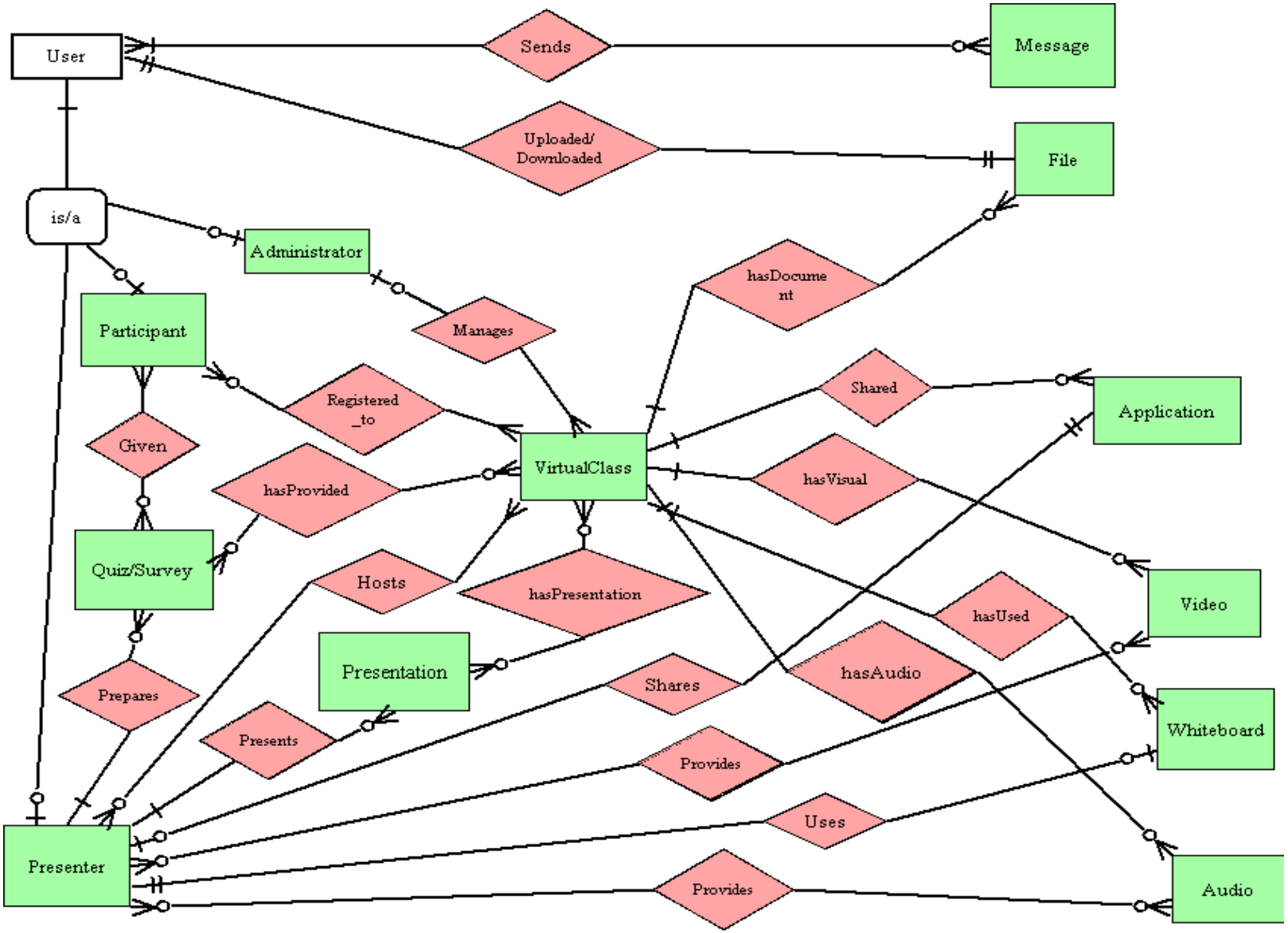


**Figure 38 Entity Relationship Diagram**

The ER Diagram of the database on the server is given in Figure 38. 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. 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 |
| e-mail | 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 **Error! Reference source not found.**). 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.

### 2.6.1 Main Window

The Main Window is a Multiple Document Interface (MDI) Container (see **Error! Reference source not found.**). The Main Window is the parent for all of the other Windows Forms.

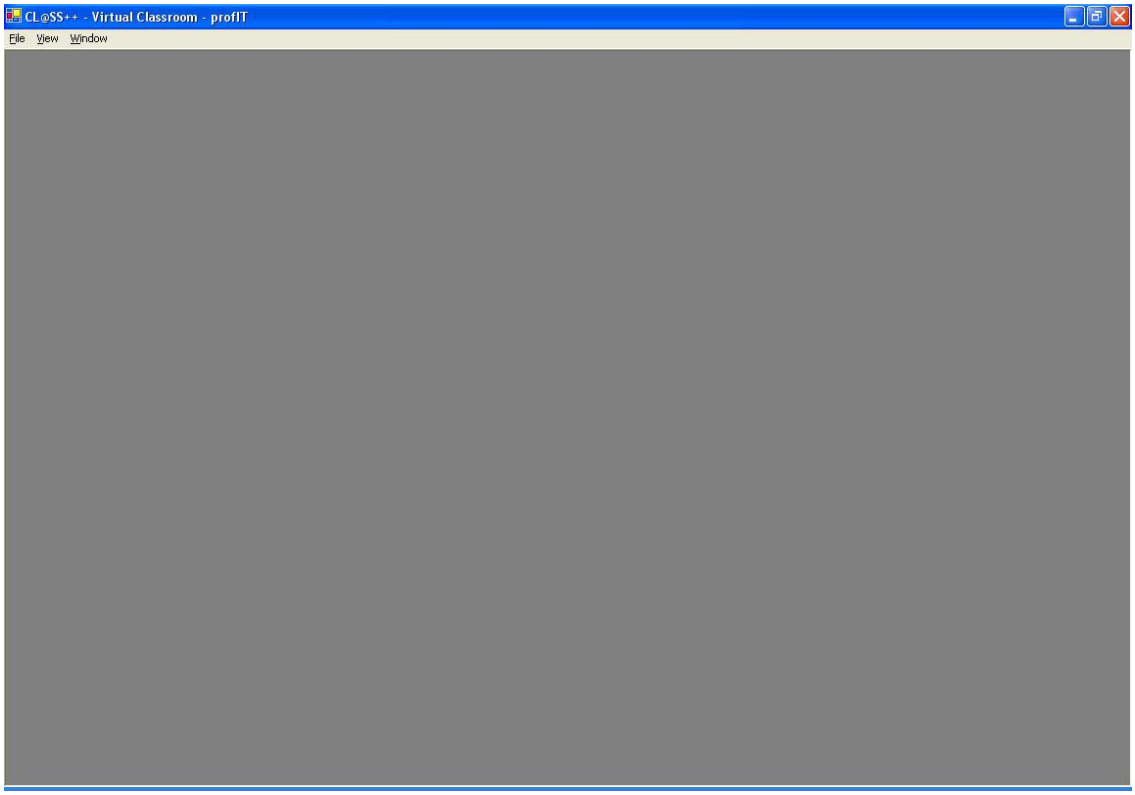


**Figure 39 CL@SS++ Main Graphical User Interface**

### 2.6.2 Presentation Module GUI

The Presentation Module provides three interfaces working together. The first is the Presentation Viewer shown in Figure 40. This interface is responsible from viewing the slides both on the Presenter side and the Participant side. Moreover, it is possible to draw or mark on the presentation on the Participant part.

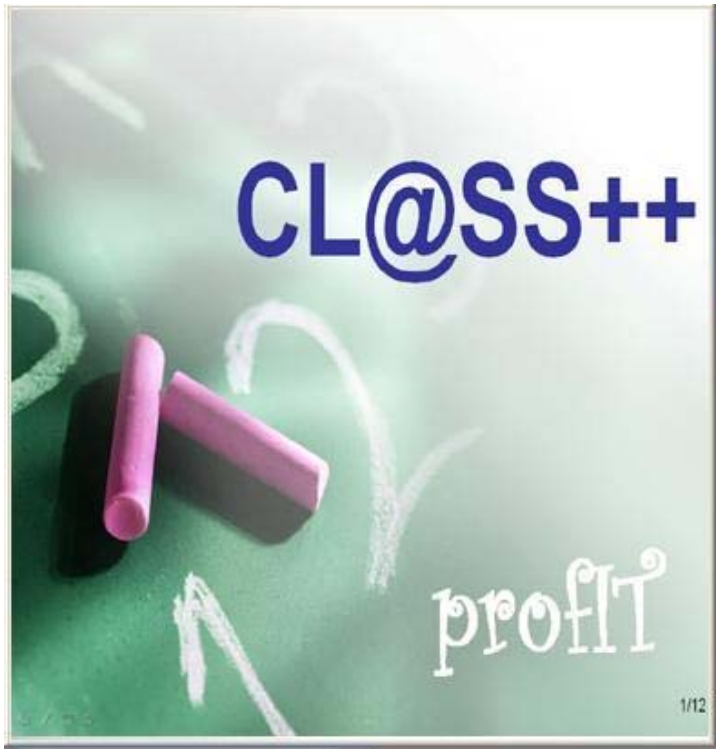


**Figure 40 Presentation Viewer GUI**

The second interface is the Presentation Controller (see Figure 41). This interface holds the Presentation Application itself and performs the necessary manipulations. The Open Button opens a new Presentation. However, it is also possible to open a presentation by dropping the presentation into the Presentation Viewer.

The Presentation Controller lets the presenter also go to the next slide or previous slide. The presenter may write the Slide Number into the textbox and upon pressing the Enter button the corresponding slide appears. Besides using these buttons the presenter may wish to use the traditional way: clicking the mouse button in the view area (on the presentation). This is controlled with the Pointer button part of the Presentation Toolbox (see Figure 42). Either the Pointer button or the "Use Whiteboard Cursor" button is selected in this Form. When the Pointer is selected, the mouse acts like in the standard PowerPoint Slide Show, whereas when the latter button is selected the current action of the Whiteboard Toolbox is performed.

This is designed this way to avoid writing the same component twice and making the application more user-friendly. While writing on the whiteboard the presenter can mark directly something on the presentation.


**Figure 41 Presentation Controller GUI**

Another feature of the Presentation Toolbox is the Copy to White Board button. The slide show is directly copied to the Whiteboard. To do this the presentation does not have to be shared. Finally, the Start/Stop Sharing button shares the presentation with the audience or restricts the broadcasting. Note that, the active buttons change color. When the button is selected again or the counter action is selected the color will return to the default button color.


**Figure 42 Presentation Toolbox GUI**

### 2.6.3 Application Sharing GUI

Figure 43 shows the Application Sharing Configuration Form. This form is used to initiate an Application Sharing or edit the properties of the Sharing process.

In the Application group box, all running processes on the local machine are displayed. Upon selecting a process, the process properties will be displayed. That is, the original position of the process and the whole actual view. The presenter can adjust the position (top-left point), because it will be displayed on the participant side on that Position by default. Besides adjusting the percentage of the view (adjusted by the presenter, but applicable to the participant

view), the presenter can limit the view Frame with the x and y coordinates relative to the window itself. This means that, the presenter can show or focus on a specific part of the application. S/he can alternatively drag the mouse on the right Panel to select the part to be displayed. In later versions, the Panel will show the application picture in this frame and the select action will be extended to just highlight the selected area in a different color.

The application is not shared before the Share button is pressed. To stop sharing the button is pressed again.

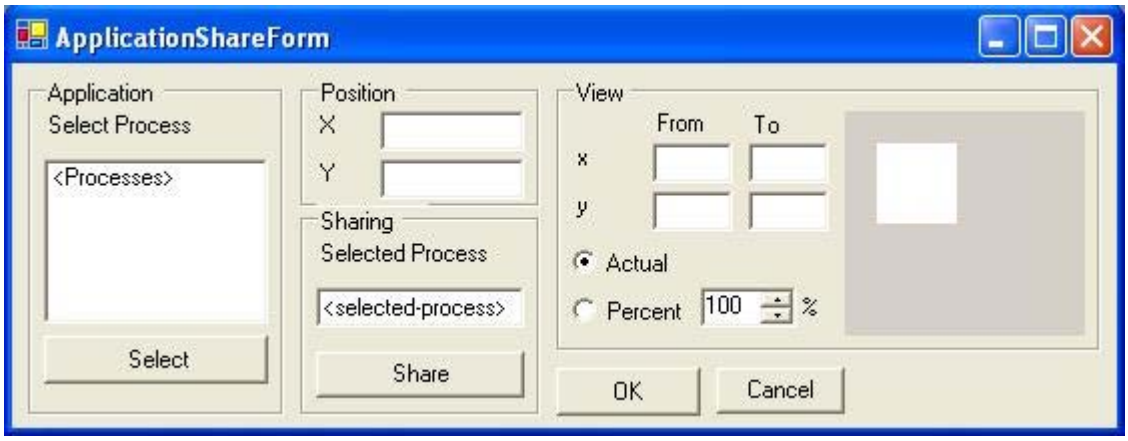


**Figure 43 Application Sharing Configuration Form**

## 2.6.4 Whiteboard GUI

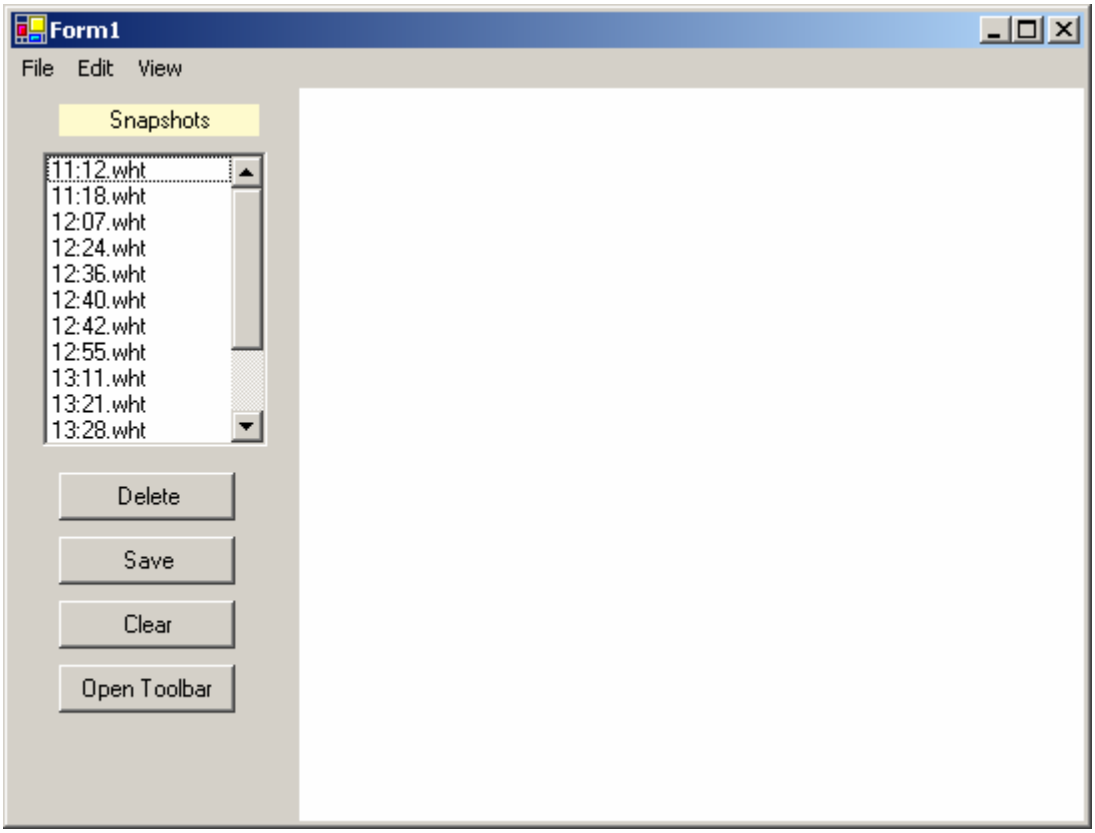


**Figure 44 White Board**

Our whiteboard GUI will have Menus for controlling the whiteboard facilities we will provide. There will be a main panel colored white for the drawings. The user of the board will be able to draw everything on this window such as lines, shapes and text. The toolbar for the different drawing styles will be in another GUI which is the Toolbar. It will be used by both whiteboard and presentations to edit. It can be opened by clicking on the OpenToolbar button.

The main menu of the GUI will contain basic operations submenus such as File and Save. The File submenu will be use for opening and closing the files as well as saving the file. In the Edit menu there will be operation such as copy, paste and cut. And also there will be a submenu for undoing changes.

In the main window of the Whiteboard GUI there will be a list of the last saved whiteboards as snapshots of the time they are taken named with the time they are saved. These files can be opened directly by double clicking on them by the user any time. Moreover there will be a delete button to delete the selected snapshot. There will be a button for saving the current board and for clearing the board.


**Figure 45 Toolbar**

As mentioned above there will be a common Toolbar providing different drawing tools for both whiteboard and the presentations. This toolbar will contain the basic tools common in most of the drawing programs such as Photoshop.

The toolbar includes buttons to choose the mouse mode. These modes define the shape to be drawn at the time of usage. These shapes can be the normal mouse pointer to draw arbitrary shapes, lines, circles, rectangles and so on. Beyond these shapes there is text mode to write on the boards. And there are listboxes and buttons for changing the text fonts.

There is a Color button to choose the drawing color. This will bring the color dialog into view and the user will be able to select the color. The current color will be shown in the colorbox in the main toolbar.

There is another button named Add Picture to open a file dialog for inserting pictures to the presentation or whiteboard. The user can choose any picture from the file dialog to add into the current window. Then he can move it in the window.

### 2.6.5 Chat GUI

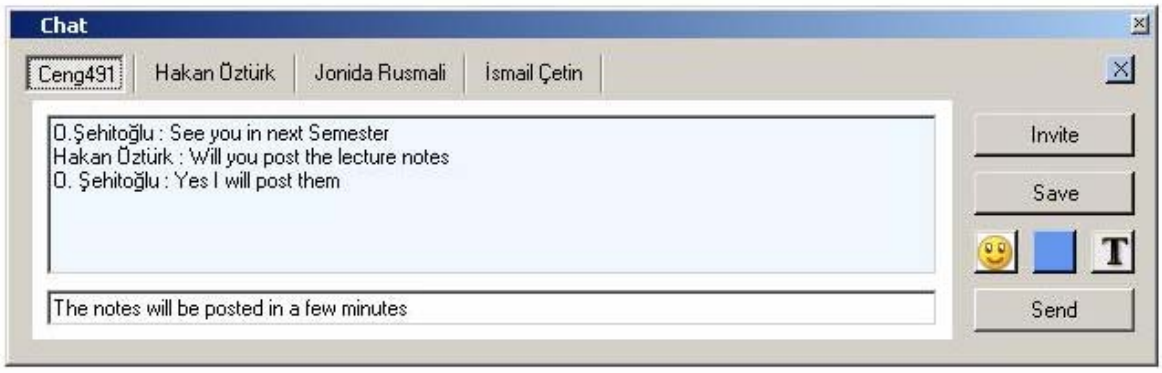


**Figure 46 Main Chat Window**

The chat module consists of many tabs each presents that that does the participant talks with. The first tab is always the classroom tab and can not be closed. The other tabs represent the private chat between the participants. The classroom tab is the public chat place and public chat place always must be present in the window so we don't allow the user to close the public chat tab. If the user wants to close the chat module he/she can do it by clicking the button which has a cross sign in the upper right corner. The button that is in the outermost closes the chat module however the button inside closes the each tab that is selected. The names of the participant are written on the tabs of the chat module. Moreover the tab will be highlighted when a new message comes from one of the participants that the user talks with.

The main user interface of the chat module consists of two main parts. The part in the right hand side shows the conversation between the users and each message line starts with the name of the user that sends the message. Furthermore there is a textbox that the user can write his/her message so that he/she can send it to the other users. The written message will be posted to the participant that the tab is down. So in this interface the message will be sent to Ceng491 which is the public chat place for this course. Also in the upper part you can easily see the messages that are sent before so you can easily look back if you forget what has happened before.

The right part of the Graphical user interface of the chat module consists of many buttons that each has a different facility. Now lets move on to these buttons and explain the facilities that each button has.


**Figure 47 Chat - User Dialog**

Invite button makes the trivial facility that it invites the user that participant wants to talk with. When clicked on the invite button a pop-up menu appears that a list of the users can be seen. By clicking on one of the user an invite message is sent to that person. He/She can join the conversation by replying that message. By the help of this facility the participant can speak more than 2 people. In the public chat part every participant can send message if he/she does not restricted by the presenter. The user list dialog also highlights the person that has the authority over this classroom. His/her name in the user list will be painted in different color other than the participants.

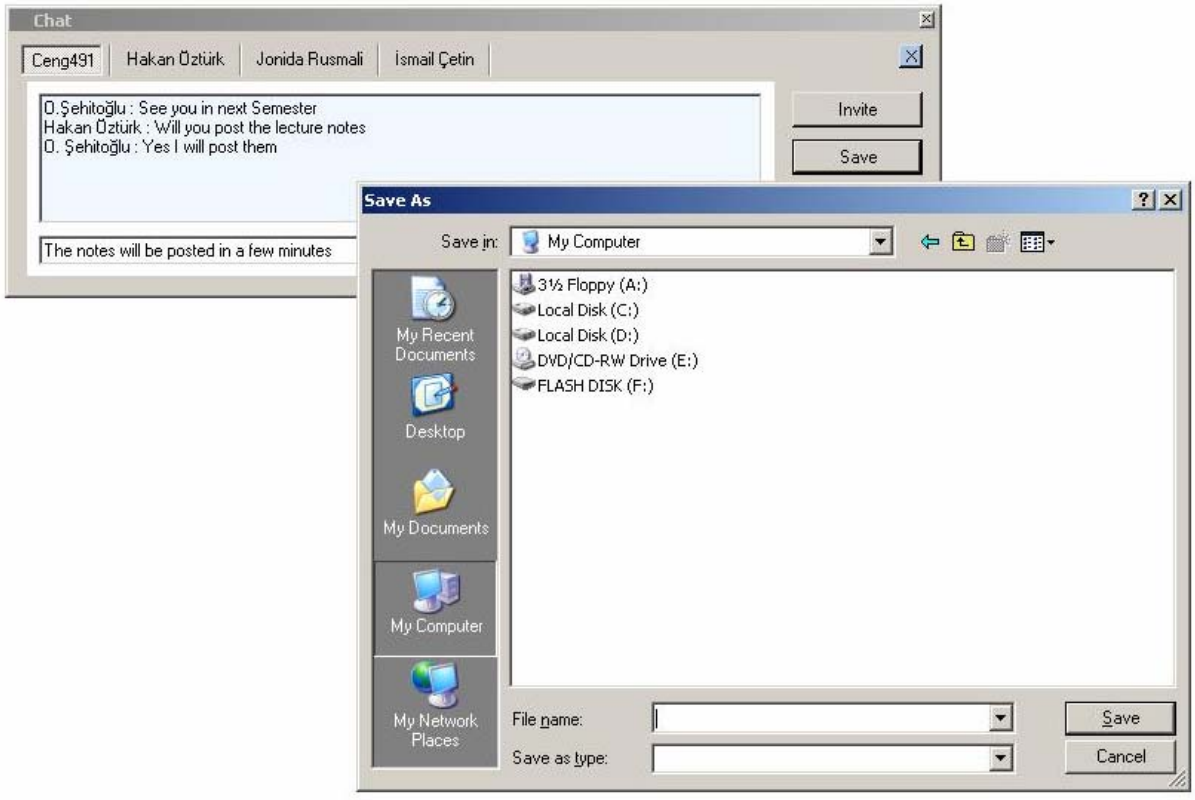


**Figure 48 Chat - Save Dialog Box**

The save button fulfills the need that one participant wants to save a conversation that is very important for him. For example, maybe in the public chat part the presenter specifies important points of the lecture or same detailed parts of the topic that he/she is presenting. So the participant will be able to save this conversation by clicking on the save button and saves the conversation to his/her hard disk. When the participant clicks the save button a save dialog appears asking where he/she wants to save his/her document in the folder tree view. He/She chooses the path and then presses the save button and conversation will be saved.

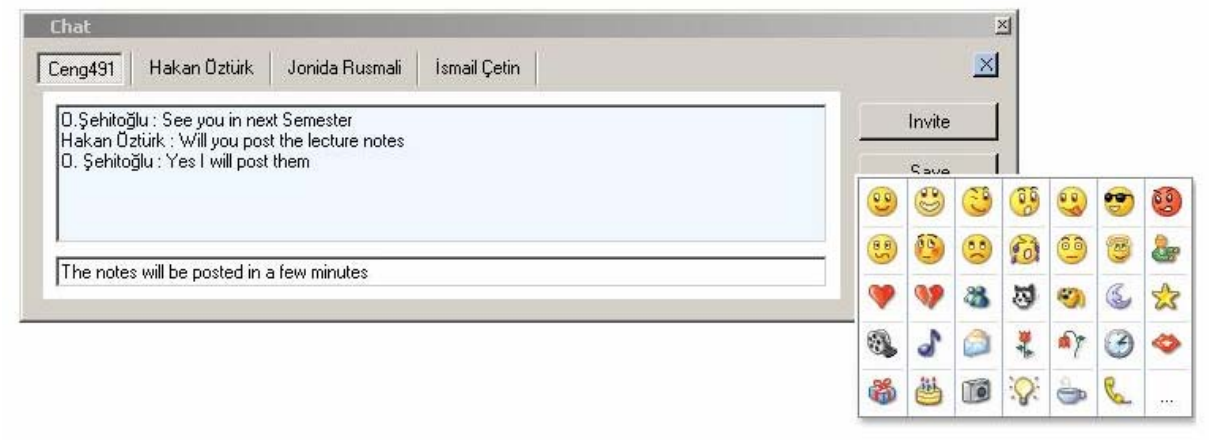


**Figure 49 Chat - Emoticon Dialog Box**

This button may not seem very important but it is spicy facility of our project. The people can not have a chance to show their emotions in chat module. However with this button the participants of the chat module can show their emotions by pictures to the person that he is talking with. When the participant presses this emoticon button, emoticon dialog box appears and he/she chooses one of the emoticons and will be able to send to the other participant.

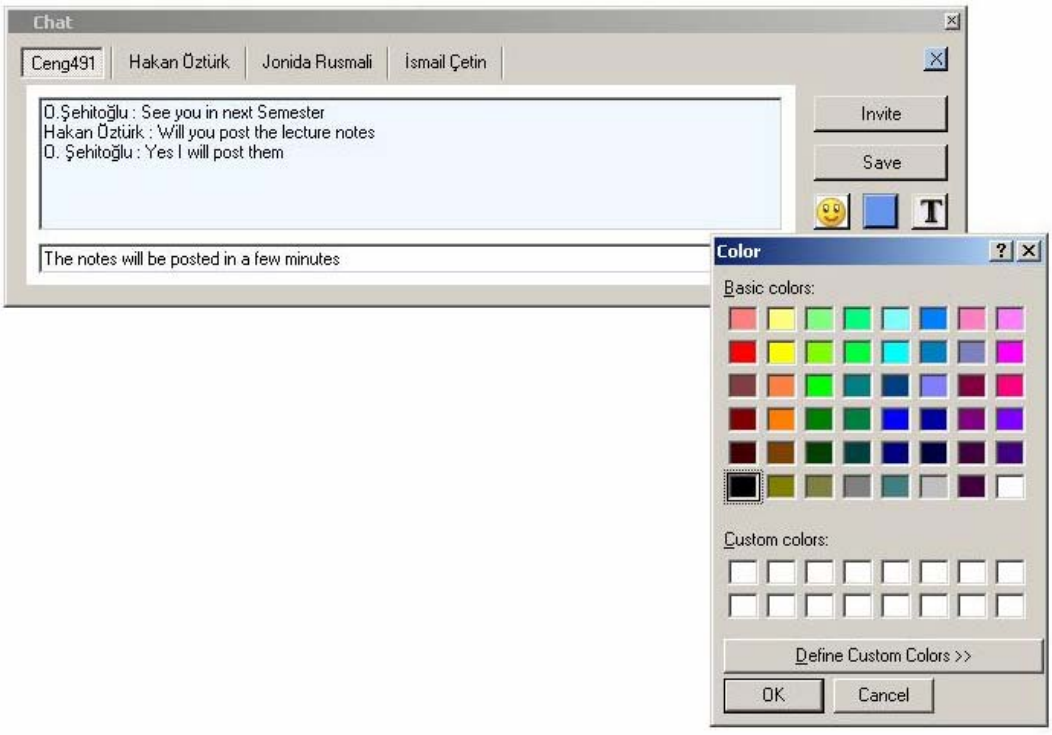


**Figure 50 Chat - Color Dialog Box**

Color button specifies the color of the written text on the message panel. This color option of the chat module also gives flexibility to send messages that are full of colors. So you can send participants that you like in different color than the classical black. A participant who wants to change the color of the text first has to the color button than a color dialog will appear. After that he/she chooses the color from dialog and can write with that color. The selected color can also be seen on the color button.

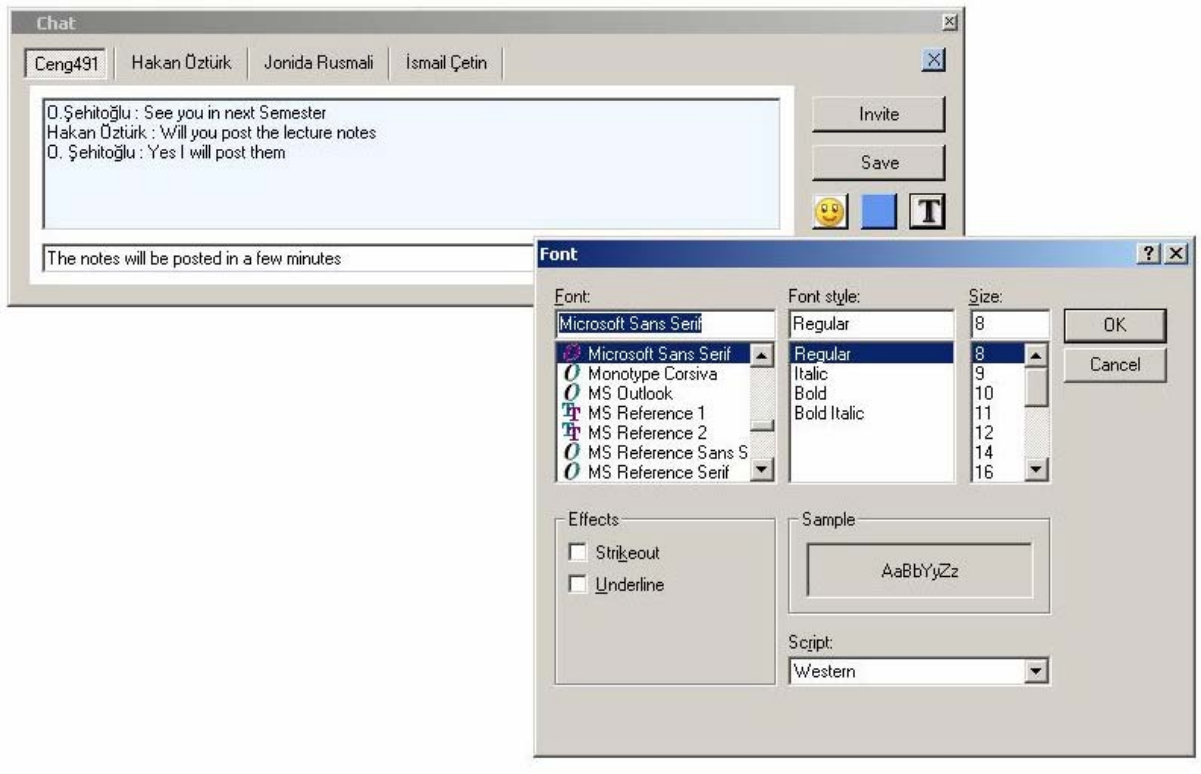


**Figure 51 Chat - Font Dialog Box**

The font button specifies the font of the written message so the participants can write with any font style that he wants. Furthermore he/she can choose the size of the font. When Participant wants to change the font of the message that he/she writes, he/she first of all presses the font button. After that a font dialog appears and he/she can choose the font type and by clicking the ok button he/she sets the font type. The messages that will be sent will appear with that font style and size.

Send button sends the written message in the textbox to the other participants. Moreover it calls a function to print the same message to the message box in the local machine. Since the same message will be transmitted to the server it will have call a broadcasting function.

### 2.6.6 Video GUI

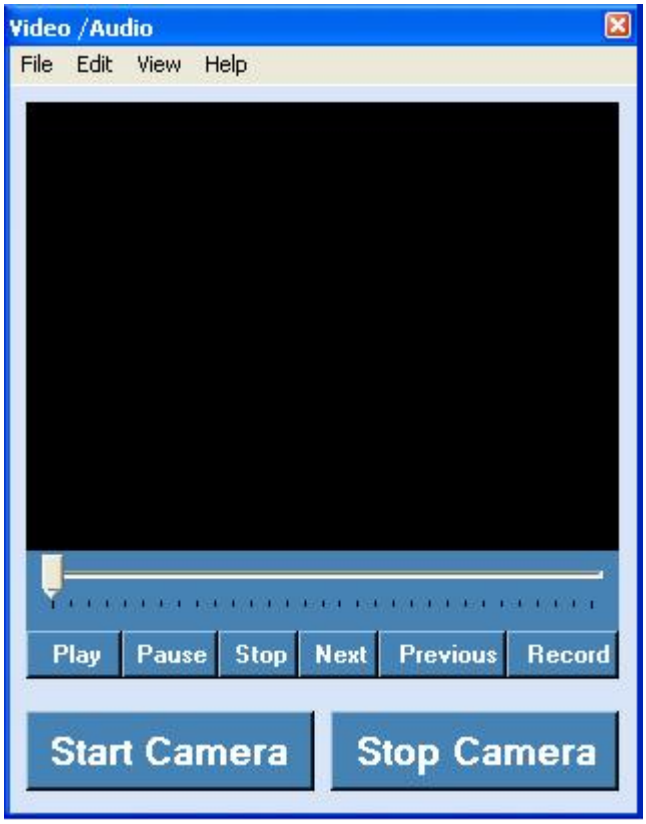


**Figure 52 Video/Audio GUI**

Our audio and video display GUIs will be separated from the main one. We have not divided the GUI in to two different GUIs which are for participant and presented. So we have composed these two in to one but with different optionally. And that window will be enlargeable if mouse is double clicked by the user when the mouse curser is on the display screen.

The buttons that are below the video displaying screen will be responsible from offline video/audio transmission. The Play button will start the video that are selected by the participant from SelectPreviousLecture list. The remaining buttons will be responsible for pausing stopping passing the next and the previous videos and recording respectively.

Above these buttons we ha a track bar which will keep the track of the video and let our users to make jumps in the offline video.

Below the control buttons we have two buttons that will only be seen by the presenters. The fist button will let our presenters to start the camera recoding. And the second one stops the recording.

### 2.6.7 User Information Panel

The connected users are displayed in this Frame. Figure 53 shows an overview of the Frame. In the frame, presenters are differentiated from the participants with color. Moreover, several conditions are displayed for the users.

The Q button stands for Question. When the users ask a question this button will change color. The H button stands for Handrising and the A button for Applause. The buttons change with user type. The presenter has for example the q button, which means that a Quiz was given by this presenter. In later versions other status information can also be displayed.

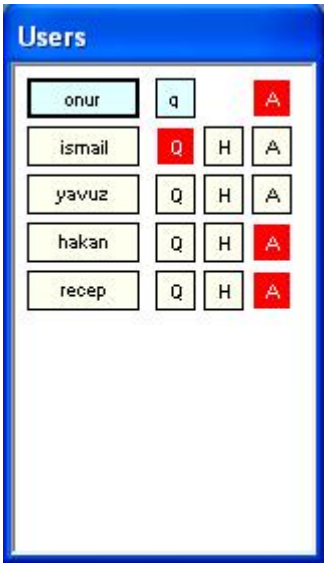


**Figure 53 Users Panel**

Another functionality of the Users Panel is the Context Menu assigned to the users. Figure 54 shows the menu for the Participants, while Figure 55 displays the Presenter menu.


**Figure 54 Users Panel - Participant Menu**

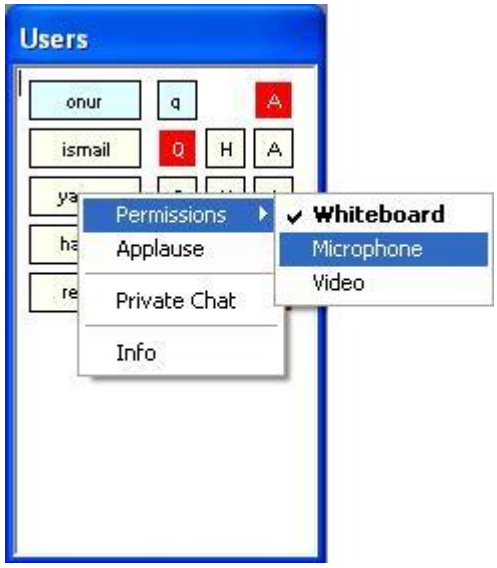


**Figure 55 Users Panel - Presenter Menu**

### 2.6.8 Virtual Class Control Form

This form delivers the basic actions to create, start, end or customize a Virtual Class. Moreover, previous classes can be opened by pressing the Open Class button.

The form is shown in Figure 56. The start button starts the virtual class. If it is a newly created class the session starts, so that other participants can join. If, on the other hand, it is a previous class, it will start the class locally.

Give Break will pop-up a dialog asking for the break time, which will count back in the textbox. This button is disabled for the participants. However, when the participants view an offline class, they can Pause the session. Tracing the offline class is also possible by using the track Bar.

The next section describes the form for creating a virtual class and viewing the properties of a virtual class.

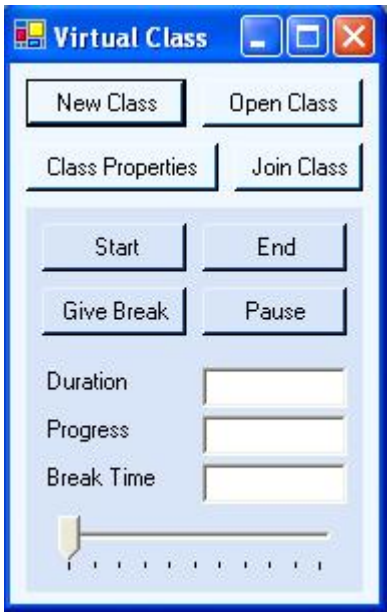


**Figure 56 Virtual Class Control Form**

### 2.6.9 Virtual Class Creation and Properties Form

The form in Figure 57 may seem rather complicated, but it provides all functionalities in a simple way. The Session Info Group box asks for the session properties. The class code can either be selected or a new one can be typed in, which will create a new class code. The name is an identifier for the session and only unique for the Code. To load the class info the Load Class Info button can be used, which displays a list of the classes associated with the current presenter.

Start Date/Time should be selected and the duration should be specified. The presenter can also specify the functionalities that will be used. However, note that, this panel can be displayed by clicking on the Class Properties Button as well, which means that the presenter can add a functionality during the class.

In the Presenters group box are the presenters displayed. The first entry shows the creator, who will manage the class. External presenters can be added directly or invited. The difference is that, when invite is selected, the presenter is sent an invitation e-mail and may or may not accept to be a presenter. With the add button a list of registered presenters are displayed, whereas with the invite button besides the registered users, it is possible to e-mail a user who is not registered. The presenter will be asked to register upon accepting the invitation.

Similarly, participants can be added. The participants can also be added as a whole class of a previous session.

The presenter can upload files either for private use or publicly available. Moreover, a previously used or created whiteboard can be added to the session.

When the OK button is clicked, the Participants will be informed through e-mail about the session.

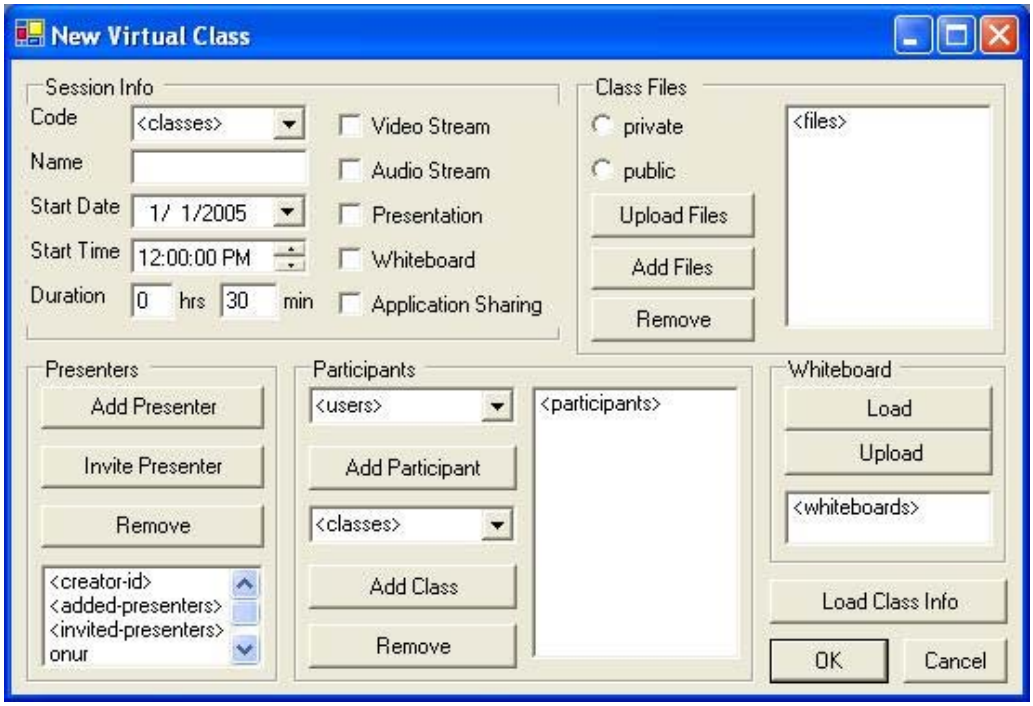


**Figure 57 New Virtual Classroom Form**

### 2.6.10 Outline Form

The outline form (see Figure 58) shows the outline of the class as well as the progress. Some buttons are disabled according to the user type.

The presenter will select the topic and press start topic to inform the system that this topic has started. Later, the participant can view the class offline and switch to a desired topic.

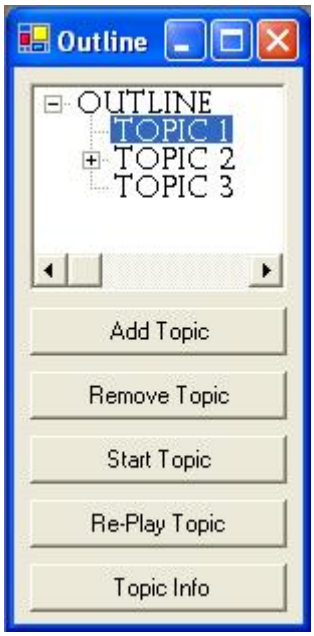


**Figure 58 Outline Form**

### 2.6.11 Tools

The tools form presents the user with some action buttons. Figure 59 and Figure 60 show the Participant and Presenter Tools respectively.


**Figure 59 Participant Tools**


**Figure 60 Presenter Tools**

### 2.6.12 Quiz Form

The presenter can create a quiz or poll using this Form (see Figure 61). The quizzes can be stored locally and opened, or uploaded to the class environment and viewed later.

The quizzes will not be visible to other users until the assignment time. Classic, Multiple Choice Quizzes or Polls can be created. The questions can be updated by selecting the related question/choice number from the combo boxes.

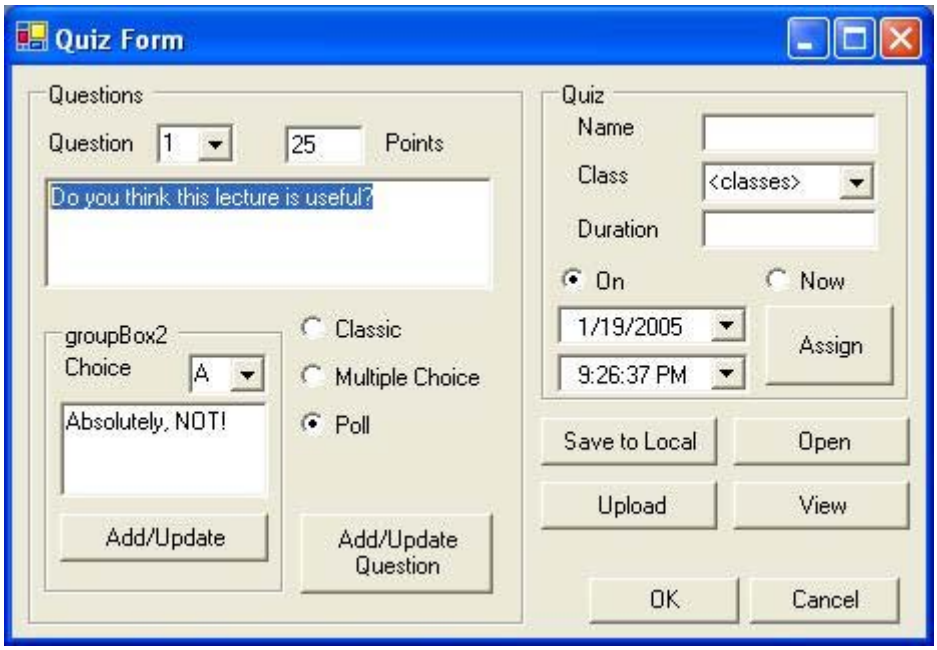


**Figure 61 Quiz Form**

# 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 Work Packages

The following table shows the Work Packages defined for the second term.

| **Work Packages** | **Implementer** | **Definition** |
|---|---|---|
| Main Module | Hakan ÖZTÜRK, Recep GÜRLEK, Yavuz GÜRCAN, İsmail ÇETİN | Integrates all of the modules in to one main module |
| Presentation Module | Yavuz GÜRCAN | Provides showing and modifying presentation facilities. |
| Whiteboard Module | Recep GÜRLEK | Provides usual blackboard facilities with additional features |
| Chat Module | İsmail ÇETİN | Provides user interconnection |
| Application Sharing Module | Yavuz GÜRCAN, Recep GÜRLEK | Supplies application sharing facility between users |
| Video/Audio Modules | Hakan ÖZTÜRK, İsmail ÇETİN | Handles the video and audio synchronization and broadcasting issues. |
| Quiz Module | Yavuz GÜRCAN | Enables Quizzes and Polls |
| Server | Yavuz GÜRCAN | In between interaction |

# 5 Conclusion

In this report we tried to give the system modules and GUIs 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.