Virtual Classroom Tool

FINAL DESIGN REPORT

CENG 491 Fall 2004

January 10, 2005

Mehmet Akif TAŞKIRMAZ
Serhat MEŞE
Abdullah ÖZTÜRK
Fırat KIZILER

Computer Engineering Department
Middle East Technical University

TABLE OF CONTENTS

1. INTRODUCTION 5

Goals and Objectives *5*

Project Scope *5*

Major Constraints *6*

Performance/Behaviour Issues 6

Management and Technical Constraints 7

2. DATA DESIGN 8

Internal Software and Data Structures *8*

Data Structures for the Instructor 8

Data Structures for the Student 10

Global Data Structures *12*

Temporary Data Structures *13*

Database Description *13*

3. ARCHITECTURAL AND COMPONENT-LEVEL DESIGN 14

Program Structure *14*

Architecture Diagram 14

Instructor State Transition Diagram 15

Student State Transition Diagram 16

Server Data Flow Diagram 17

Instructor Data Flow Diagram 18

Student Data Flow Diagram 19

Entity – Relation Diagram 20

Component Descriptions *21*

Component Descriptions for the Instructor 21

Component Descriptions for the Student 35

Details Of Processes *43*

Software Interface Description *44*

4. UML DIAGRAMS 45

Class Diagrams *45*

Instructor Class Diagram 45

Student Class Diagram 52

Collaboration Diagrams *55*

Instructor Collaboration Diagram 55

Student Collaboration Diagram 57

5. USER INTERFACE DESIGN 58

Description of the User Interface *58*

Components Available *65*

6. RESTRICTIONS, LIMITATIONS AND CONSTRAINTS 66

Performance/Behaviour Issues *66*

Program Limitations *66*

7. PROJECT SCHEDULE 67

Detailed Plan *67*

Gannt Chart *68*

8. TESTING ISSUES 69

Classes of Tests *69*

Unit Testing 69

Integration Testing 69

High-Order Testing 69

Expected Software Response *69*

General Hardware Requirements 69

Identification Of Critical Components *69*

AudioVisual Streaming 69

Synchronization of Components 69

9. GLOSSARY 71

10. REFERENCES 76

1. INTRODUCTION

Goals and Objectives

The purpose of the project is to develop a virtual classroom tool (KLAS) for synchronous distance learning with multimedia facilities. This distance learning takes place when a teacher and student(s) are separated by physical distance, and technology (i.e., voice, video, data, and print), often in concert with face-to-face communication. These types of programs can provide adults with a second chance at a college education, reach those disadvantaged by limited time, distance or physical disability, and update the knowledge base of workers at their places of employment [1; 2].

KLAS will provide network broadcasting of a multimedia lecture given by a trainer to multiple clients, interaction with the students, preparation, online editing and offline replay of the courses.

Project Scope

In a physical classroom there is a standard set of audio-visual equipment and tools available to the instructor. These might include a chalkboard, overhead projector, video cassette player, possibly a sound system, and even the textbook. The virtual classroom has equivalent equipment and tools in the form of network-based software applications [3]. This equipment and the equivalent software applications in KLAS are as follows:

- chalkboard: whiteboard
- pencil: multimedia pointer
- video cassette recorder: multimediaboard
- textbook: e-book
- sound and video teleconferencing: audiovisual system

The software will consist of a number of inputs and input files including the following:

- Video Files
- Sound Files
- MS Powerpoint Files
- Animation Files
- Bitmaps
- Camera
- Microphone

The software will consist of a number of outputs including the following:

- Audiovisual Streams
- Video Files
- Sound Files
- Text Files

- Database Files
- Snapshots

Major Constraints

Performance/Behaviour Issues

KLAS requires the following software:

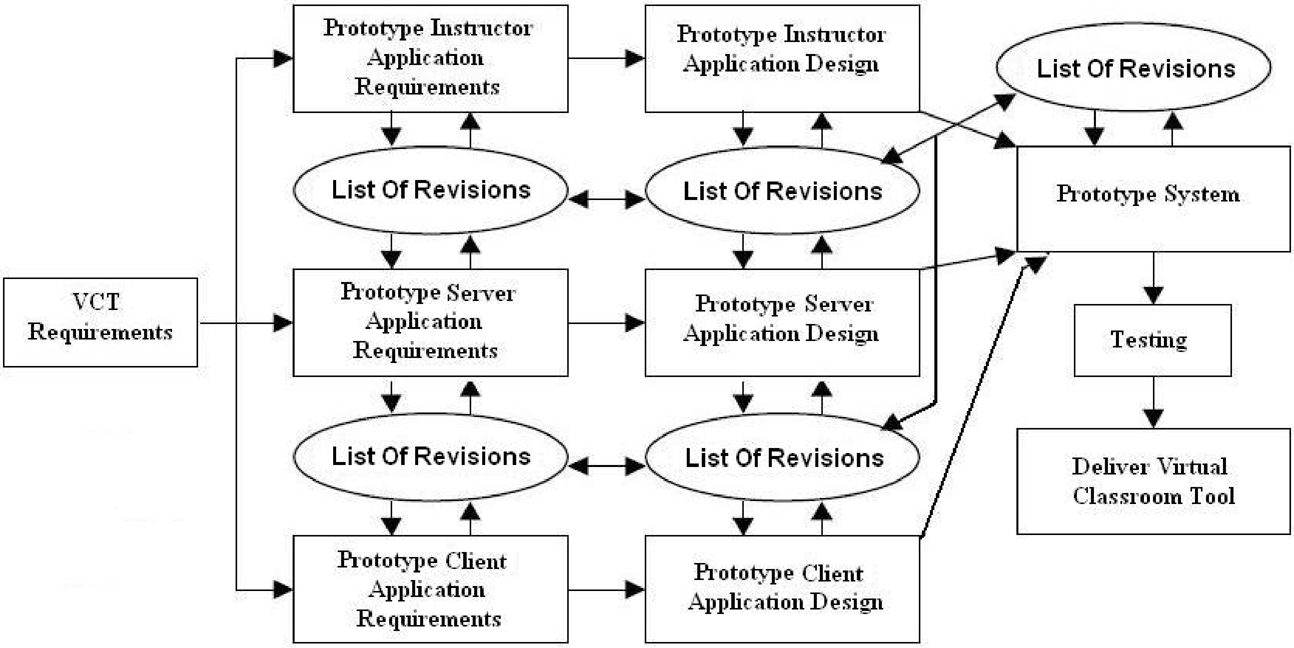
- Microsoft Windows Media SDK
- Microsoft Windows Media Server
- Microsoft Windows Media Encoder
- Microsoft Windows Media Player 9.0 or higher
- Microsoft SQL Server 7.0 or higher
- DirectX 8.1 or higher
- Microsoft Powerpoint 2000 or higher
- Microsoft Word 2000 or higher
- Microsoft Excel 2000 or higher
- Macromedia Flash Player

The server side of the KLAS can only work on Microsoft Windows 2000/2003 server operating systems because Windows Media Server is distributed only with these operating systems.

The client side of the KLAS is designed to be compatible with Microsoft Windows 2000/XP/2003 operating systems.

The reason for choosing DirectX 8.1 or higher is that the GDI+ Libraries (which is used for creating interfaces) requires at least DirectX 8.1.

Management and Technical Constraints


VCT Requirements
Prototype Instructor Application Requirements
Prototype Instructor Application Design
List Of Revisions
List Of Revisions
List Of Revisions
Prototype Server Application Requirements
Prototype Server Application Design
List Of Revisions
List Of Revisions
Prototype Client Application Requirements
Prototype Client Application Design
Prototype System
Testing
Deliver Virtual Classroom Tool

2. DATA DESIGN

Internal Software and Data Structures

Data Structures for the Instructor

broadcast

broadcast consists of the following attributes:

- **AVBroadcastState:** Determine the state of the audiovisual_broadcast object.
- **MBroadcastState:** Determine the state of the multimediaboard_broadcast object.
- **WBroadcastState:** Determine the state of the whiteboard_broadcast object.

multimediaboard

multimediaboard consists of the following attributes:

- **Content:** The content of the multimediaboard (slides, animations, etc…)
- **State:** Determine whether any file is shown or not

whiteboard

whiteboard consists of the following attributes:

- **ContentArray:** The content of the whiteboard. For the shapes it is a structure array each element of which represents a drawn shape. Each element is defined as follows:

  ```
  struct shape
  {
       int x1, y1;
       int x2, y2;
       int type;
  }
  ```

  `type` defines the shape type. The values of `x1, x2, y1` and `y2` are specified according to the value of `type`. For instance, if `type = LINE`, then `(x1, y1)` and `(x2, y2)` define the starting and ending coordinates of the line. On the other hand, if `type = CIRCLE`, then `(x1, y1)` defines the coordinates of the center, `x2` defines the radius of the circle and `y2` is not used for this type.

  For the freehand drawn figures, all the adjacent points are stored in the structure array.
- **State:** Determine whether anything is drawn or not

whiteboard_broadcast

whiteboard_broadcast consists of the following attributes:

- **SocketID:** The socket no for the coming whiteboard content

player

player consists of the following attributes:

- **Play:** The status of the play button
- **Stop:** The status of the stop button
- **State:** Determines the play state (playing, not playing)

audiovisual_data

audiovisual_data consists of the following attributes:

- **Alias:** The filename of the audiovisual_data
- **Path:** The directory that the audiovisual_data is found in
- **Buffer:** The buffer to load the audiovisual_data into
- **VolumeLevel:** Determines the volume level

course_content

course_content consists of the following attributes:

- **InstructorID:** The ID of the instructor
- **Student ID:** The array that contains the IDs of the students that are enrolled in that course
- **CourseID:** The ID of the course
- **CourseFiles:** The array that contains the names of the files that are related to the specified course

help

help consists of the following attributes:

- **Topic:** The help topic that will be displayed
- **Explanation:** The explanation of the help topic

chat

chat consists of the following attributes:

- **MessageType:** Determines the type of the message (question or chat message)
- **MessageText:** The message string
- **MessageLength:** The length of the message
- **MessageState:** Determines the state (send or get message)
- **SocketID1:** The socket number for the coming message
- **SocketID2:** The socket number for the going message

login_manager

login_manager consists of the following attributes:

- **LoginName:** The code of the error.
- **Password:** The error message.

- **ConfirmationState:** Determines the state of the confirmation info.

error

error consists of the following attributes:

- **ErrorCode:** The code of the error.
- **ErrorMessage:** The error message.

database

database consists of the following attributes:

- **RequestID:** The ID of the request coming from other objects.

control

control consists of the following attributes:

- **StatesOfButtons:** The array that contains states of all the buttons.
- **RequestID:** Determines the ID of the request.

session_manager

session_manager consists of the following attributes:

- **OnlineUsers:** The array that contains the usernames of the online users.

Data Structures for the Student

multimediaboard

multimediaboard consists of the following attributes:

- **Path:** The URL address of the multimedia stream on the server
- **Content:** The content of the multimediaboard (slides, animations, etc…)
- **State:** Determine whether any file is shown or not

whiteboard

whiteboard consists of the following attributes:

- **ContentArray:** The content of the whiteboard. For the shapes it is a structure array each element of which represents a drawn shape. Each element is defined as follows:

```
struct shape
{
     int x1, y1;
     int x2, y2;
     int type;
}
```

type defines the shape type. The values of x1, x2, y1 and y2 are specified according to the value of type. For instance, if type = LINE, then (x1, y1) and (x2, y2) define the starting and ending coordinates of the line. On the other hand, if type = CIRCLE, then (x1, y1) defines the coordinates of the center, x2 defines the radius of the circle and y2 is not used for this type.
For the freehand drawn figures, all the adjacent points are stored in the structure array.

- **State:** Determine whether anything is drawn or not
- **SocketID:** The socket no for the coming whiteboard content

player

player consists of the following attributes:

- **Play:** The status of the play button
- **Stop:** The status of the stop button
- **State:** Determines the play state (playing, not playing)

audiovisual_data

audiovisual_data consists of the following attributes:

- **Path:** The URL address of the audiovisual stream on the server
- **Buffer:** The buffer to load the audiovisual_data into
- **VolumeLevel:** Determines the volume level

course_content

course_content consists of the following attributes:

- **Student ID:** The array that contains the IDs of the students that are enrolled in that course
- **InstructorID:** The ID of the instructor
- **CourseID:** The ID of the course
- **CourseFiles:** The array that contains the names of the files that are related to the specified course

help

help consists of the following attributes:

- **Topic:** The help topic that will be displayed
- **Explanation:** The explanation of the help topic

chat_client

chat_client consists of the following attributes:

- **MessageType:** Determines the type of the message (question or chat message)
- **MessageText:** The message string

- **MessageLength:** The length of the message
- **MessageState:** Determines the state (send or get message)
- **SocketID1:** The socket number for the coming message
- **SocketID2:** The socket number for the going message

login_manager

login_manager consists of the following attributes:

- **LoginName:** The code of the error.
- **Password:** The error message.
- **ConfirmationState:** Determines the state of the confirmation info.

error

error consists of the following attributes:

- **ErrorCode:** The code of the error.
- **ErrorMessage:** The error message.

control

control consists of the following attributes:

- **StatesOfButtons:** The array that contains states of all the buttons.
- **RequestID:** Determines the ID of the request.

online_users

online_users consists of the following attributes:

- **OnlineUsers:** The array that contains the usernames of the online users.

Global Data Structures

object_handler

object_handler consists of the following attributes:

- **BroadcastControl:** State of the broadcast component
- **DatabaseControl:** State of the database component
- **QuestionControl:** State of the question component
- **HelpControl:** State of the help component
- **LoginManagerControl:** State of the login_manager component
- **ErrorControl:** State of the error component
- **ChatServerControl:** State of the chat_server component
- **CourseContentControl:** State of the course_content component
- **PlayerControl:** State of the player component

Temporary Data Structures

No temporary data structures are created.

Database Description

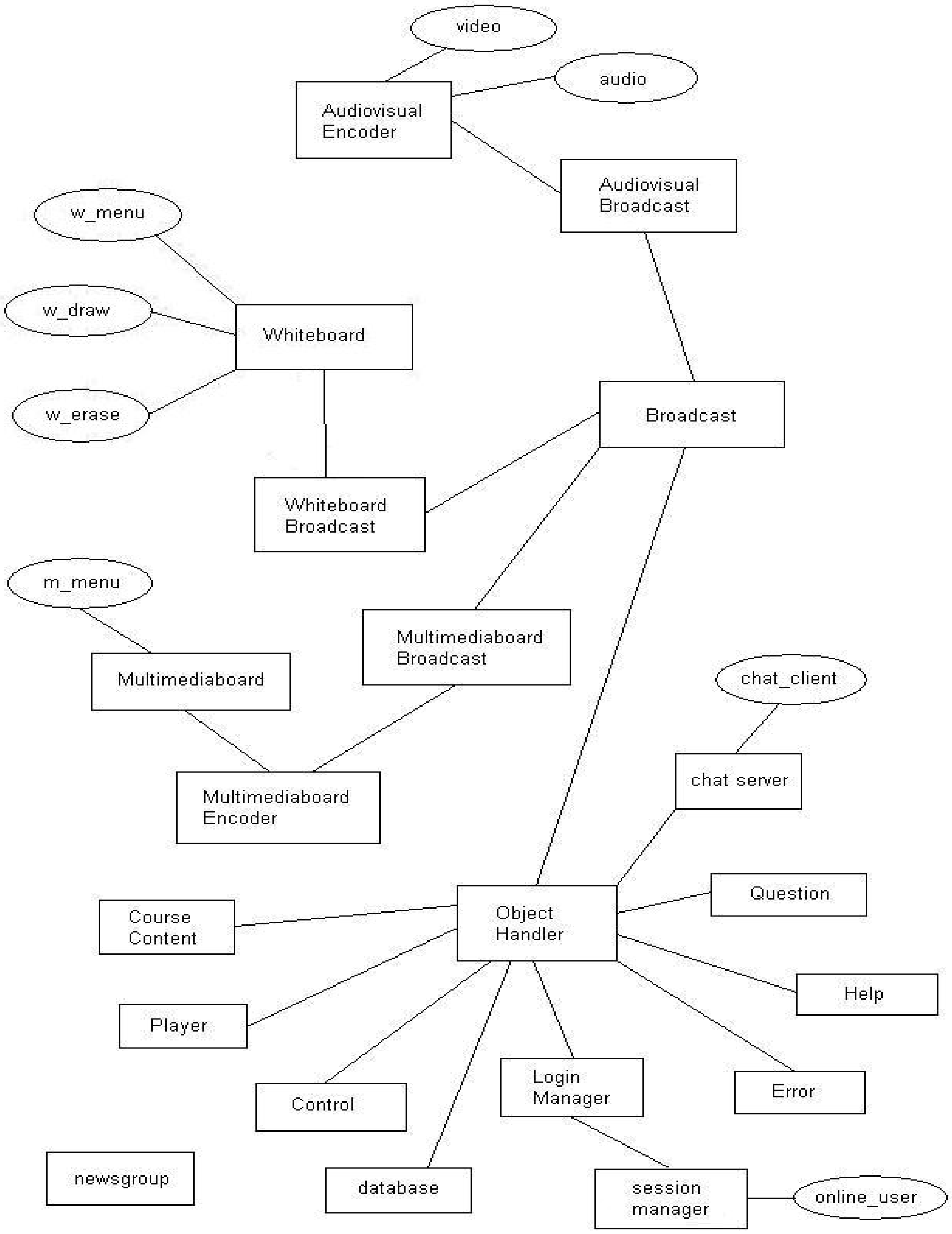
The database we are using is Microsoft SQL Server 7.0. It will hold the following information:

- Users account information
- Course information
- Help content
- Audiovisual content
- Multimedia content
- Chat messages

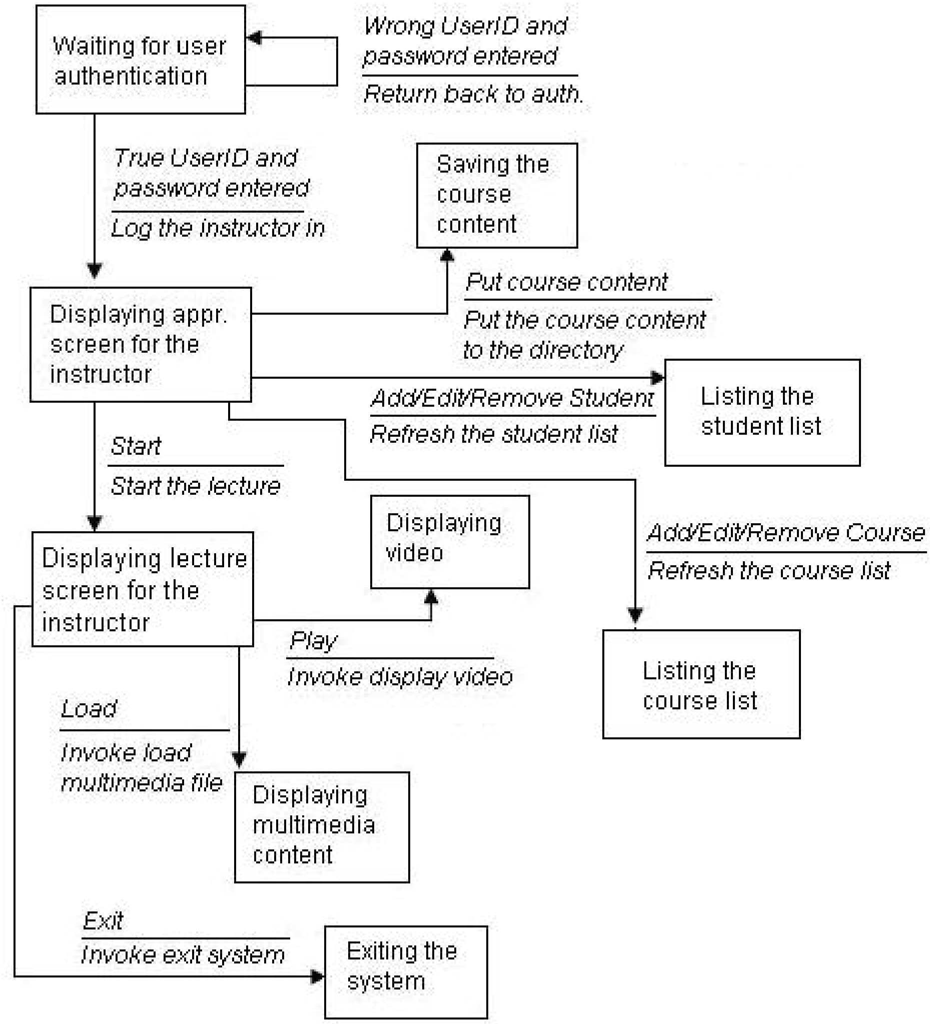
3. ARCHITECTURAL AND COMPONENT-LEVEL DESIGN

Program Structure

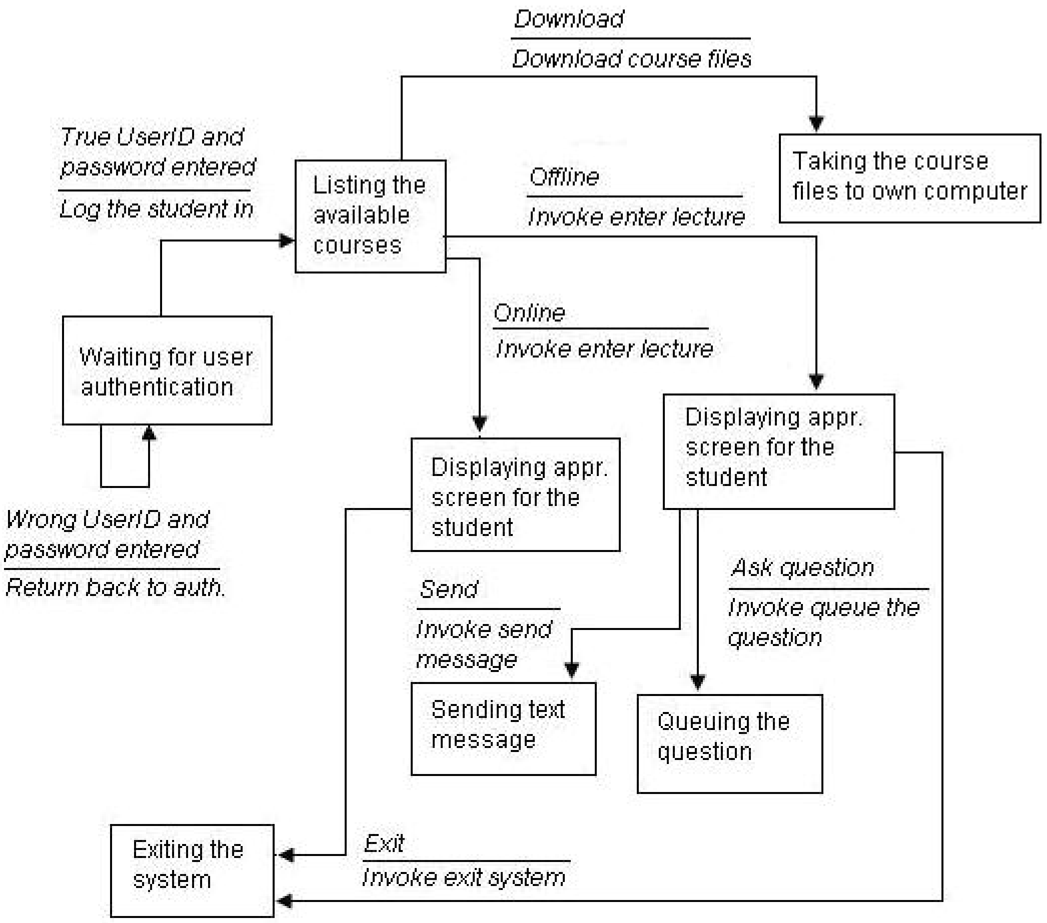
Architecture Diagram



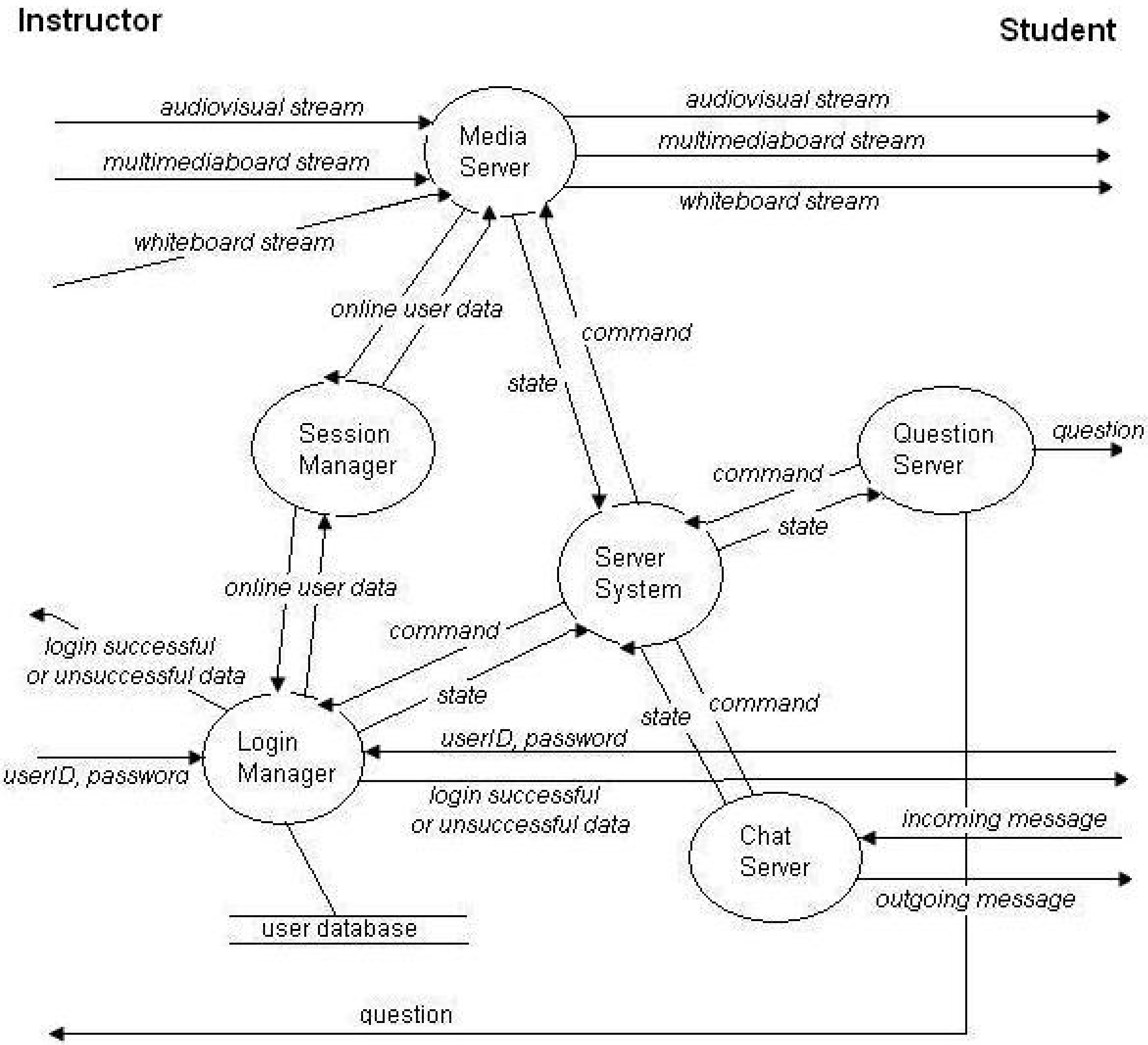
Instructor State Transition Diagram



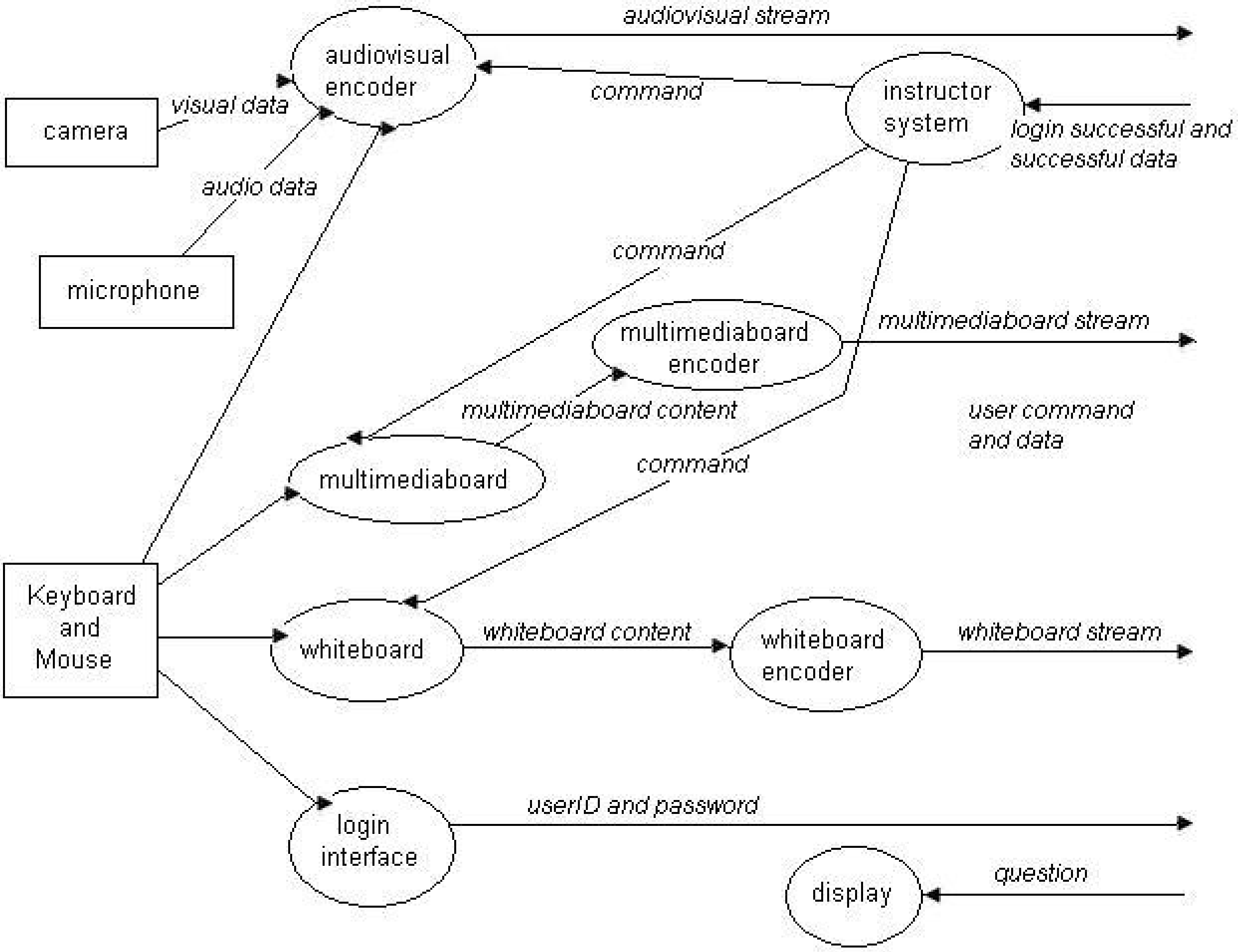
Student State Transition Diagram


Download
Download course files
True UserID and password entered
Log the student in
Listing the available courses
Offline
Invoke enter lecture
Taking the course files to own computer
Online
Invoke enter lecture
Waiting for user authentication
Displaying appr. screen for the student
Displaying appr. screen for the student
Wrong UserID and password entered
Return back to auth.
Ask question
Invoke queue the question
Send
Invoke send message
Sending text message
Queuing the question
Exiting the system
Exit
Invoke exit system

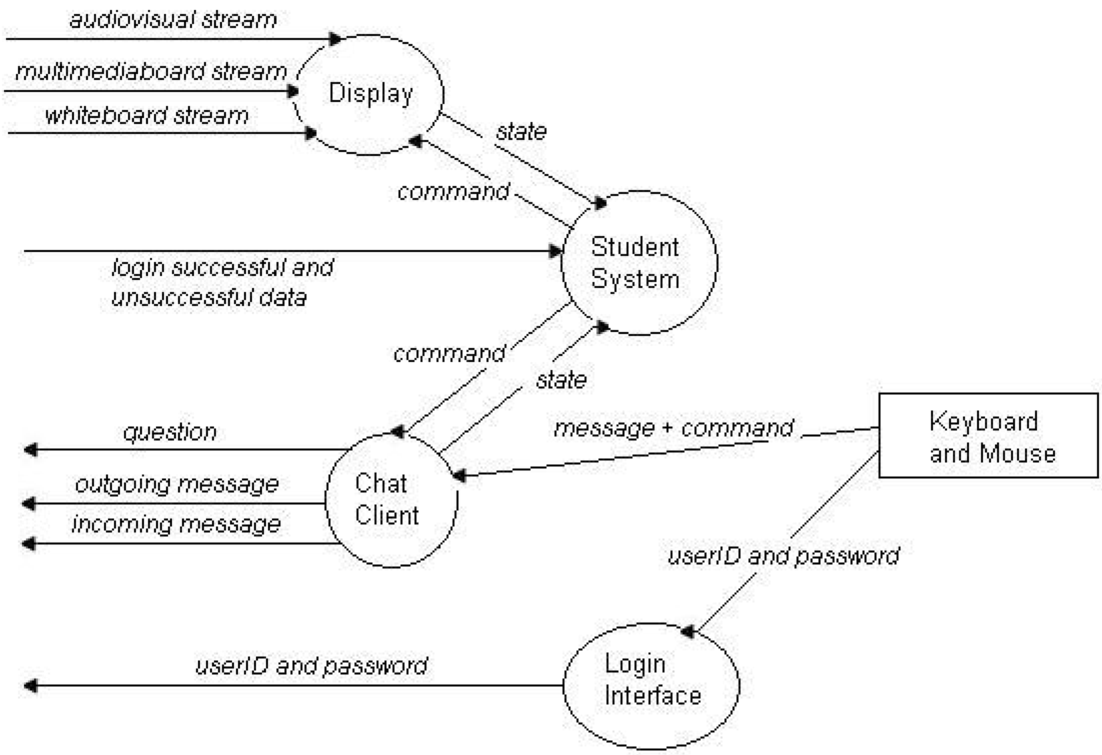
Server Data Flow Diagram



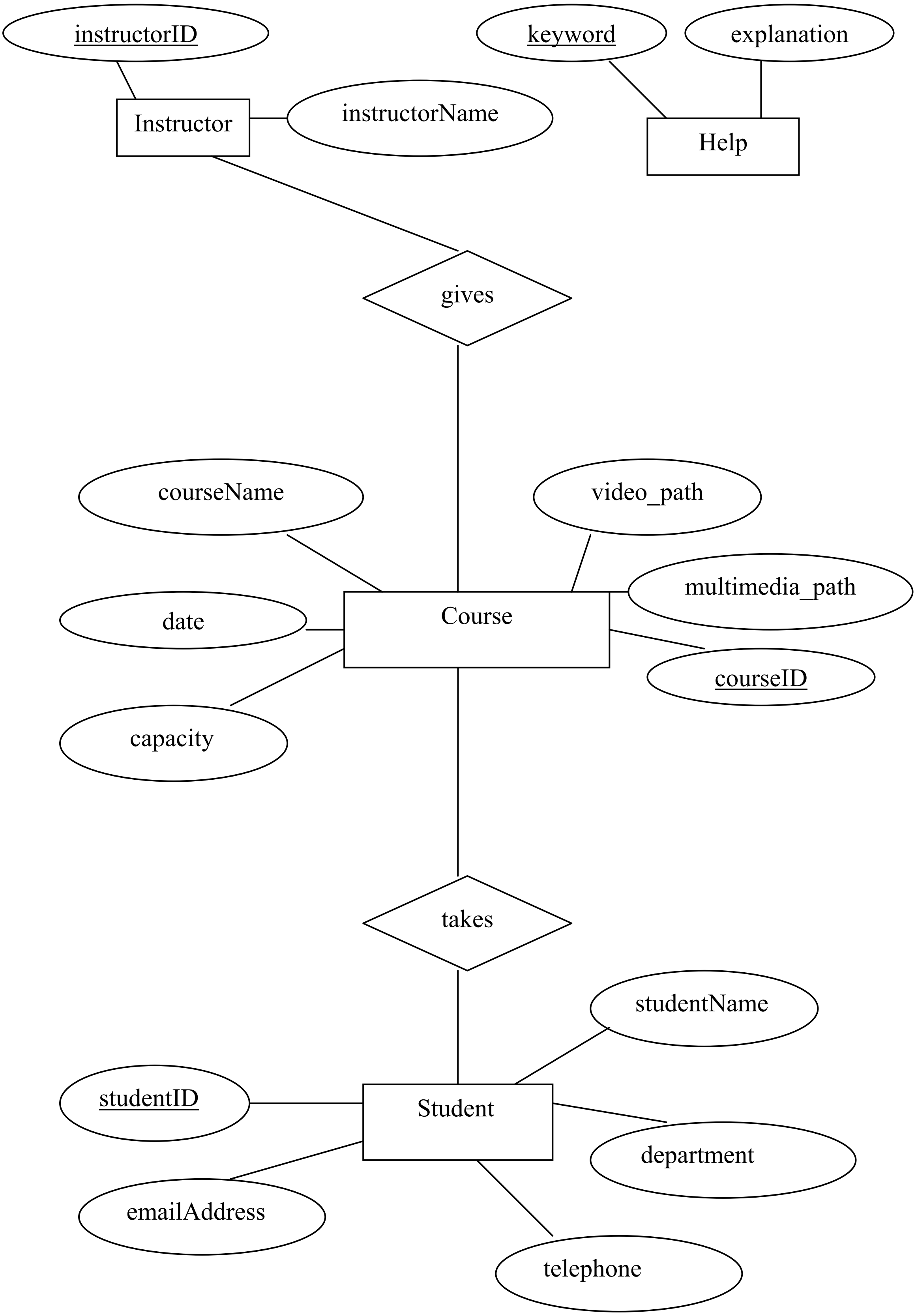
Instructor Data Flow Diagram



Student Data Flow Diagram



Entity-Relation Diagram



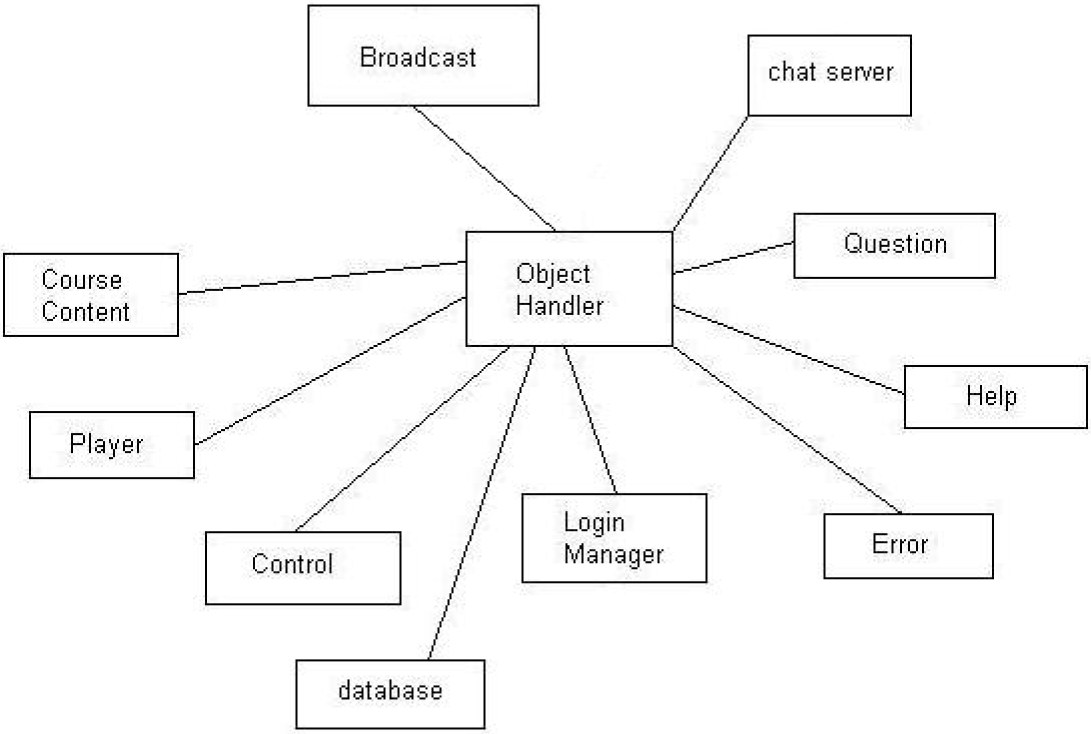
Component Descriptions

Component Descriptions for the Instructor

object_handler

- **Narrative:** Receives the states of broadcast, chat_server, question, help, player, database, error, login_manager, control, course_content objects as inputs. It controls these components.

- **Diagram:**



- **Interface:** This component interfaces with the broadcast, chat_server, question, help, player, database, error, login_manager, control, course_content objects.

- **Functions:** GetState(), SendCommand()

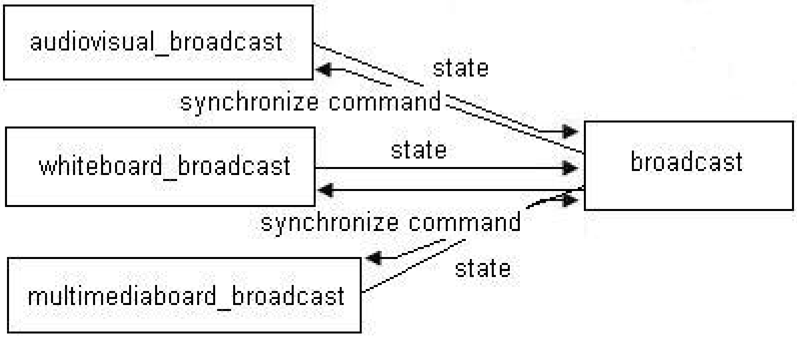
- **Issues:** This is the most critical component. It controls all the components to work together.

- **Constraints:** None.

broadcast

- **Narrative:** Receives the states of the audiovisual_broadcast, multimediaboard_broadcast and whiteboard_broadcast components. It controls (synchronizes) these objects.

- **Diagram:**



- **Interface:** This component interfaces with the audiovisual_broadcast, multimediaboard_broadcast and whiteboard_broadcast objects.

- **Functions:** GetState(), SendSynchronizeCommand()

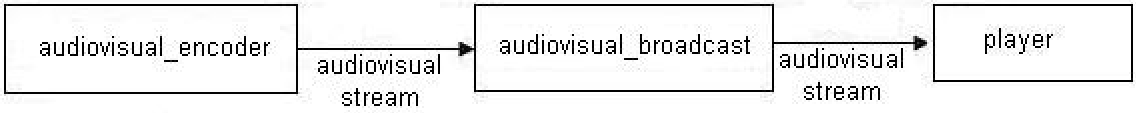
- **Issues:** None.

- **Constraints:** None.

audiovisual_broadcast

- **Narrative:** Receives the audiovisual stream as the input from the audiovisual_encoder. It broadcasts this stream.

- **Diagram:**



- **Interface:** This component interfaces with the player and audiovisual_encoder objects.

- **Functions:** BroadcastAVContStream(), GetAVContStream().

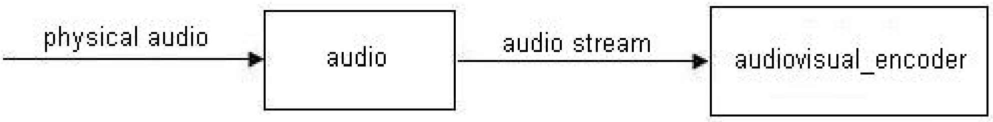
- **Issues:** None.

- **Constraints:** None.

audio

- **Narrative:** Receives physical audio. It converts the physical audio to digital audio and sends this digital audio data to the audiovisual encoder.

- **Diagram:**



- **Interface:** This component interfaces with the audiovisual encoder.

- **Functions:** GetAudio(), SendAudio()

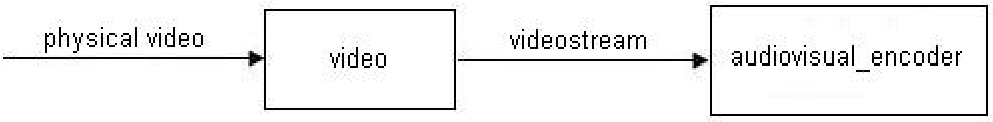
- **Issues:** None.

- **Constraints:** None.

video

- **Narrative:** Receives physical video. It converts the physical video to digital video and sends this digital video data to the audiovisual encoder.

- **Diagram:**



- **Interface:** This component interfaces with the audiovisual encoder.

- **Functions:** GetVideo(), SendVideo()

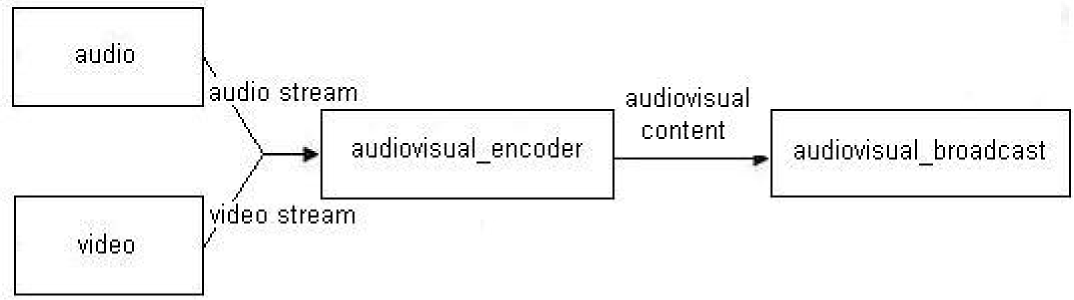
- **Issues:** None.

- **Constraints:** None.

audiovisual_encoder

- **Narrative:** Receives digital audio and digital video from the audio and video objects. It combines these and sends the audiovisual stream to the audiovisual_broadcast object.

- **Diagram:**



- **Interface:** This component interfaces with the audio, video and audiovisual_broadcast objects.

- **Functions:** GetDigitalVideo(), GetDigitalAudio(), SendAudioVideo()

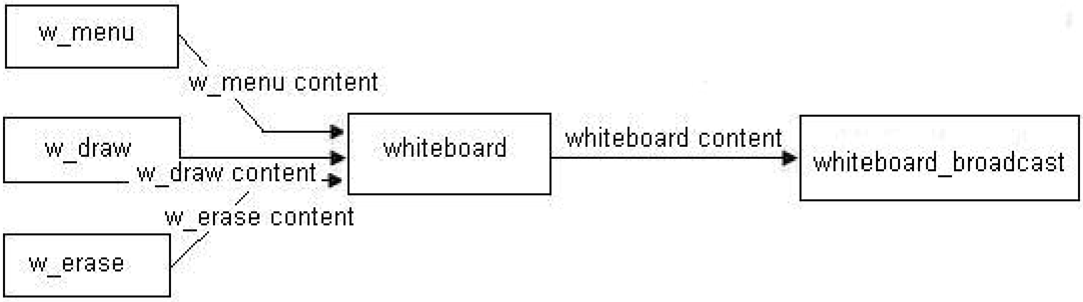
- **Issues:** The digital video files should be in the appropriate format (asf, avi, mpg and wmv). The digital audio files should be in the appropriate format (mp3, wav, wma).

- **Constraints:** None.

whiteboard

- **Narrative:** Receives w_menu, w_draw and w_erase contents as inputs. Its job is sending the whole whiteboard content to the whiteboard_broadcast component.

- **Diagram:**



- **Interface:** This component interfaces with the w_menu, w_draw, w_erase and whiteboard_encoder objects.

- **Functions:** GetWMenuContent(), DrawContent(), GetWDrawContent(), GetWEraseContent(), SendWContent().

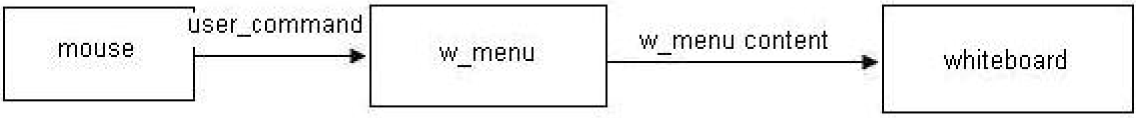
- **Issues:** None.

- **Constraints:** None.

w_menu

- **Narrative:** Receives the command that is clicked with the mouse. It performs the specified menu function.

- **Diagram:**



- **Interface:** This component interfaces with the whiteboard object and mouse.

- **Functions:** New(), Save(), Load(), SendWMenuContent().

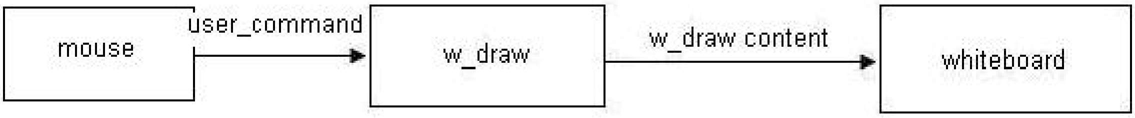
- **Issues:** None.

- **Constraints:** None.

w_draw

- **Narrative:** Receives the shape information that is clicked with the mouse. It performs the drawing jobs and it stores the content (line, rectangle, freehand drawn figures, etc…) in the structure array that is defined in the whiteboard data structure above.

- **Diagram:**



- **Interface:** This component interfaces with the whiteboard object and the mouse.

- **Functions:** DrawFreeHand(), DrawLine(), DrawCircle(), DrawEllipse(), DrawArrow(), WriteText(), SendWDrawContent().

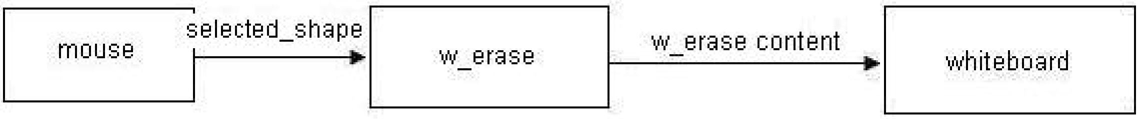
- **Issues:** None.

- **Constraints:** None.

w_erase

- **Narrative:** Receives the selected shape that is drawn on the whiteboard. It erases this shape.

- **Diagram:**



- **Interface:** This component interfaces with the whiteboard object and mouse.

- **Functions:** EraseShape(), SendEraseContent()

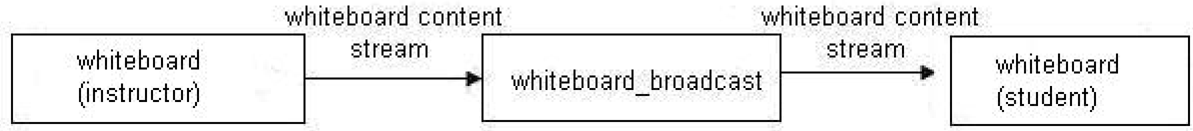
- **Issues:** None.

- **Constraints:** None.

whiteboard_broadcast

- **Narrative:** Receives the whiteboard content data structure as the input. It broadcasts this content to the students.

- **Diagram:**



- **Interface:** This component interfaces with the whiteboard components on the instructor side and student side.

- **Functions:**BroadcastWContentStream(), GetWContentStream().

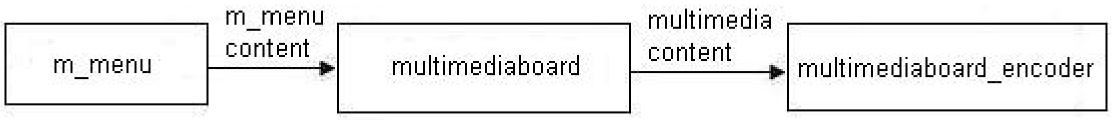
- **Issues:** None.

- **Constraints:** None.

multimediaboard

- **Narrative:** Receives multimediaboard content and m_menu content as inputs. Its job is sending the whole multimediaboard content to the multimediaboard_encoder.

- **Diagram:**



- **Interface:** This component interfaces with the m_menu and multimediaboard_encoder objects.

- **Functions:** GetMContent(), GetMMenuContent(), SendMContent().

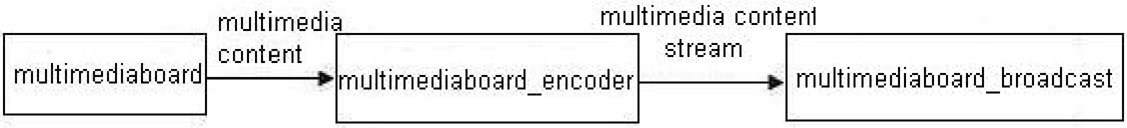
- **Issues:** None.

- **Constraints:** None.

multimediaboard_encoder

- **Narrative:** Receives the multimediaboard content as the input. It converts the multimediaboard content into stream and sends this stream to the multimediaboard _broadcast object.

- **Diagram:**



- **Interface:** This component interfaces with the multimediaboard and multimediaboard_broadcast objects.

- **Functions:** GetMContent(), SendMContentStream().

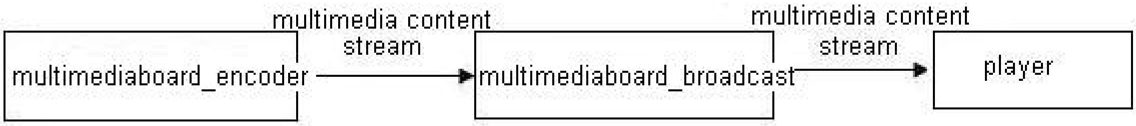
- **Issues:** None.

Constraints: None.

multimediaboard_broadcast

- **Narrative:** Receives the multimediaboard content stream as the input. It broadcasts this stream.

- **Diagram:**


multimediaboard_encoder
multimedia content stream
multimediaboard_broadcast
multimedia content stream
player

- **Interface:** This component interfaces with the player and multimediaboard_encoder objects.

- **Functions:**BroadcastMContentStream(), GetMContentStream().

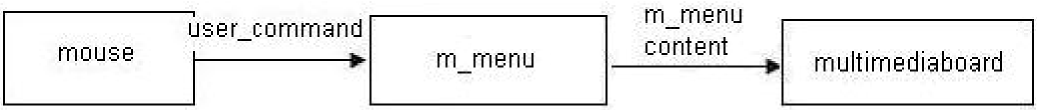
- **Issues:** None.

- **Constraints:** None.

m_menu

- **Narrative:** Receives the command that is clicked with the mouse. It performs the specified menu function.

- **Diagram:**



- **Interface:** This component interfaces with the multimediaboard object.

- **Functions:** Open(), SendMMenuContent().

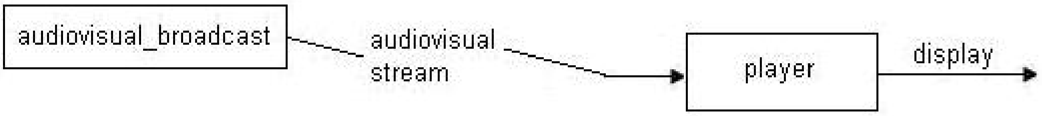
- **Issues:** None.

- **Constraints:** None.

player

- **Narrative:** Receives the audiovisual stream. It displays this stream on the screen.

- **Diagram:**



- **Interface:** This component interfaces with the audiovisual_broadcast.

- **Functions:** Display(), Play(), Stop(), SetVolumeLevel()

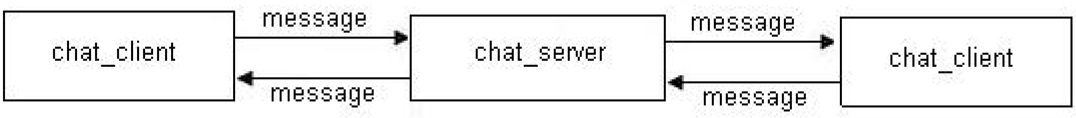
- **Issues:** The stream format will be in wmv format.

- **Constraints:** If the stream is corrupted, the player can not display it correctly.

chat_server

- **Narrative:** Receives messages as inputs. It transmits the messages between the chat clients.

- **Diagram:**



- **Interface:** This component interfaces with the chat_client objects.

- **Functions:** ServerGetMessage(), ServerSendMessage()

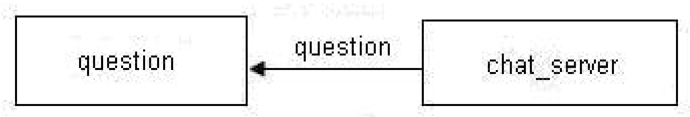
- **Issues:** None.

- **Constraints:** None.

question

- **Narrative:** Receives messages (questions) as inputs. It displays these questions.

- **Diagram:**



- **Interface:** This component interfaces with the chat_server object.

- **Functions:** GetQuestion().

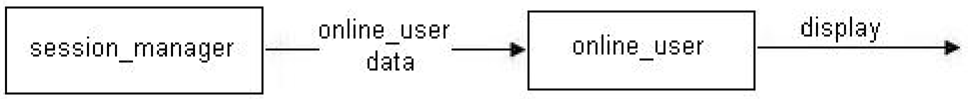
- **Issues:** None.

- **Constraints:** None.

online_users

- **Narrative:** Receives the online users data from the session_manger object as the input. It displays the clients who are online.

- **Diagram:**



- **Interface:** This component interfaces with the session_manager object.

- **Functions:** DisplayOnlineUsers()

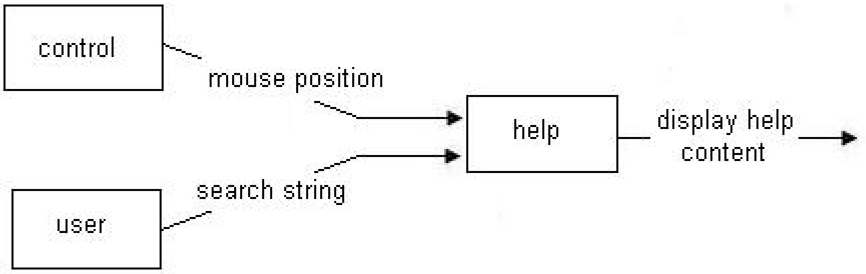
- **Issues:** None.

- **Constraints:** None.

help

- **Narrative:** Receives the mouse position from the control component or the search string from the user as the inputs. It searches the database and displays the help content dynamically.

- **Diagram:**



- **Interface:** This component interfaces with the database and control components.

- **Functions:** ConnectDatabaseServer(), Search(), DisplayHelpContent()

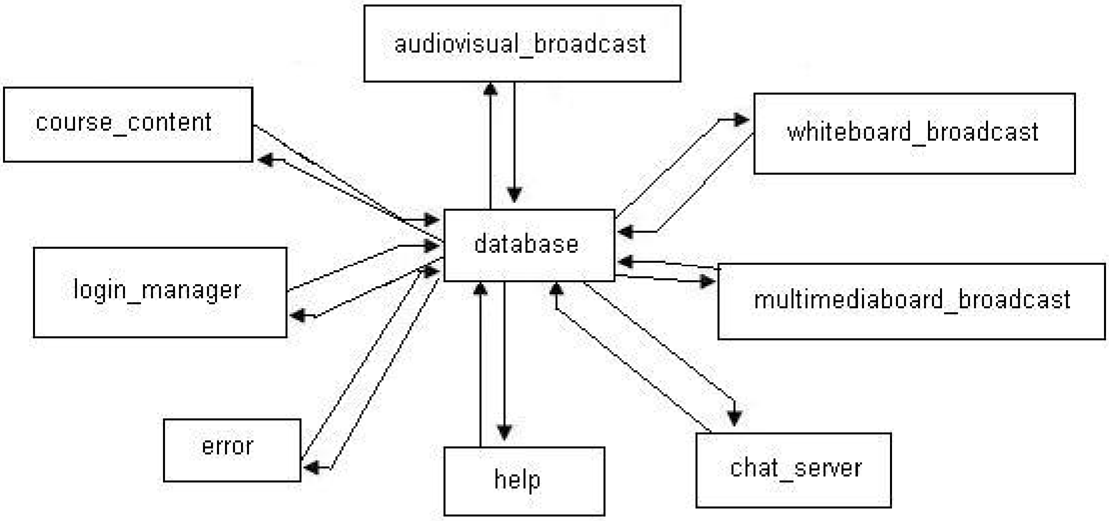
- **Issues:** The help content for all Mouse positions must be defined correctly, otherwise, the help will not work properly.

- **Constraints:** If the search string is not found in the database, nothing will be displayed.

database

- **Narrative:** Receives the requests and sends the required contents.

- **Diagram:**



- **Interface:** This component interfaces with the login_manager, error, help, chat_server, multimediaboard_broadcast, whiteboard_broadcast, audiovisual_broadcast and course_content.

- **Functions:** GetRequst(), SendContent()

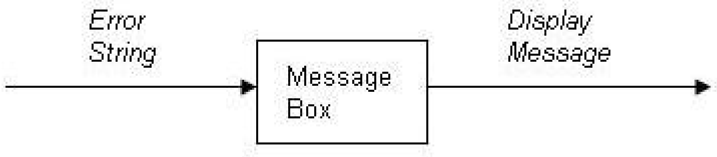
- **Issues:** If the connection fails, the required content can not be obtained.

- **Constraints:** None.

error

- **Narrative:** Receives exceptions as the input. It handles these.

- **Diagram:**

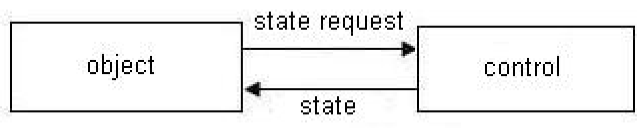


- **Interface:** This component interfaces with all of the components.
- **Functions:** GetException(), HandleException()
- **Issues:** If the errors are not handled, the program may crash.
- **Constraints:** All the error types should be handled correctly.

control

- **Narrative:** Receives state requests from the objects as input. It holds the buttons' and components' states and sends these data to those objects.

  All the necessary conditions must be satisfied when a job is to be performed. For instance, when a client wants to attend a lecture, that lecture must be given at that moment, he/she must be online and he/she must be registered for that course. All these necessary conditions are stored in this component and these conditions are checked before a job is to be performed by using this component.
- **Diagram:**

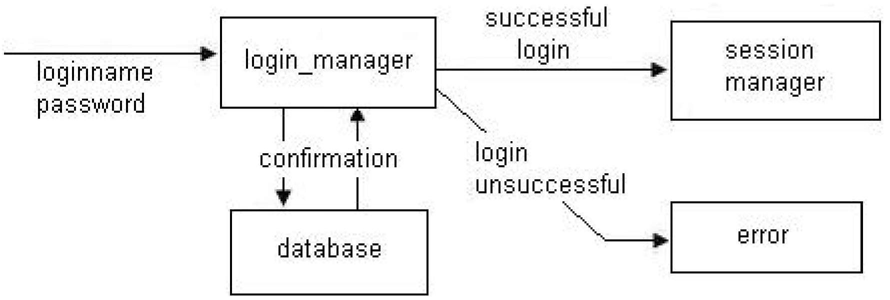


- **Interface:** This component interfaces with all of the components.
- **Functions:** GetRequest(), SendState()
- **Issues:** None.
- **Constraints:** None.

login_manager

- **Narrative:** Receives the login name and password from the user. It gives authentication or not. It sends the successful logins to the session_manager object.

- **Diagram:**



- **Interface:** This component interfaces with the database, session_manager and error objects.

- **Functions:** ConnectDatabaseServer(), ConfirmPassword()

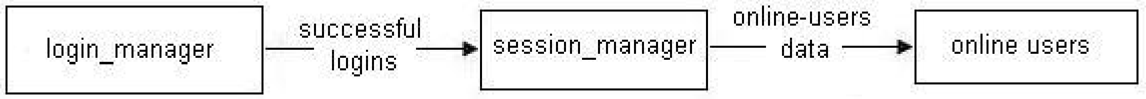
- **Issues:** If the login is unsuccessful, there will be error.

- **Constraints:** If the user is already online, then the login manager prevents logging in with the same userID and password.

session_manager

- **Narrative:** Receives the successful logins from the login manager as the input. It sends the online users data to the online_user object.

- **Diagram:**



- **Interface:** This component interfaces with the login_manager and online_users object.

- **Functions:** GetOnlineUsers(), SendOnlineUsers()

- **Issues:** None.

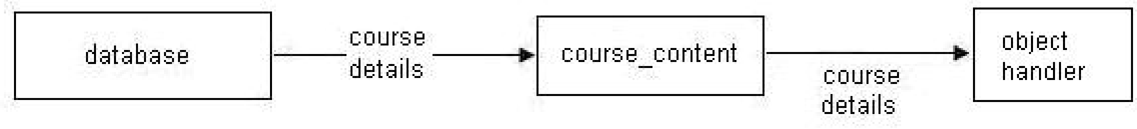
- **Constraints:** None.

newsgroup

- **Narrative:** Receives no input. This is the newsgroup on the web.
- **Diagram:** None.
- **Interface:** None (It Works independently).
- **Functions:** None.
- **Issues:** None.
- **Constraints:** None.

course_content

- **Narrative:** Receives the course details from the database as input. It sends this content to the object handler (start lecture).
- **Diagram:**



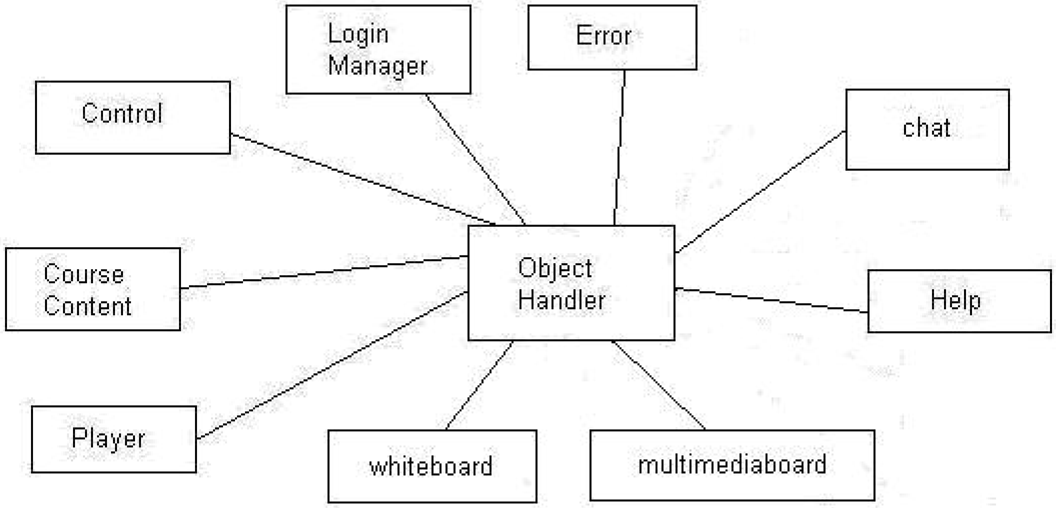
- **Interface:** This component interfaces with the database component and the object handler component.
- **Functions:** ConnectDatabaseServer(), AddNewCourse(), EditCourse(), RemoveCourse(), AddStudent(), EditStudent(), RemoveStudent(), StartLecture().
- **Issues:** None.
- **Constraints:** None.

Component Descriptions for the Student

object_handler

- **Narrative:** Receives the states of chat_client, question, help, player, error, login_manager, control, course_content objects as inputs. It controls these components.

- **Diagram:**



- **Interface:** This component interfaces with the chat_client, question, help, player, error, login_manager, control, course_content objects.

- **Functions:** GetState(), SendCommand()

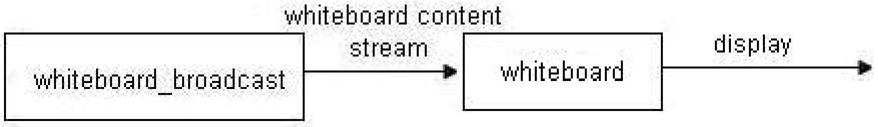
- **Issues:** This is the most critical component. It controls all the components to work together.

- **Constraints:** None.

whiteboard

- **Narrative:** Receives the whiteboard content stream from the whiteboard_broadcast component. It displays this content in the whiteboard area.

- **Diagram:**



- **Interface:** This component interfaces with the whiteboard_broadcast object.

- **Functions:** GetWContent(), DisplayWContent().

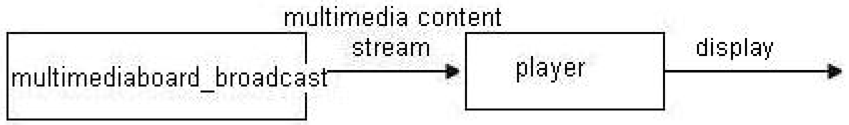
- **Issues:** None.

- **Constraints:** None.

multimediaboard

- **Narrative:** Receives the multimediaboard content stream from the multimediaboard_broadcast component. It displays this content in the multimediaboard area.

- **Diagram:**



- **Interface:** This component interfaces with the multimediaboard_broadcast object.

- **Functions:** GetMContent(), DisplayMContent().

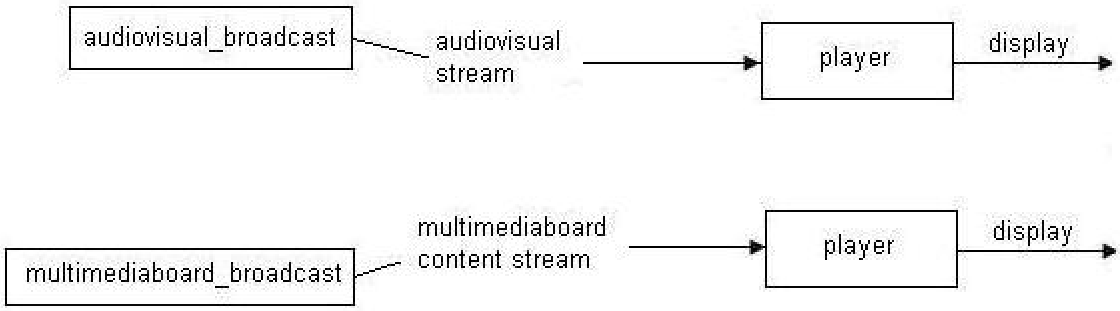
- **Issues:** None.

- **Constraints:** None.

player

- **Narrative:** Receives the audiovisual stream and multimediaboard content stream seperately. It displays these streams on different player instances.

- **Diagram:**



- **Interface:** This component interfaces with the audiovisual_broadcast, multimediaboard_broadcast objects.

- **Functions:** Display(), Play(), Stop(), SetVolumeLevel()

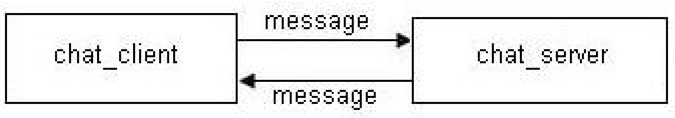
- **Issues:** The stream format will be in wmv format.

- **Constraints:** If the stream is corrupted, the player can not display it correctly.

chat_client

- **Narrative:** Receives messages as inputs. It receives and sends messages to the chat server.

- **Diagram:**



- **Interface:** This component interfaces with the chat_server object.

- **Functions:** ClientGetMessage(), ClientSendMessage().

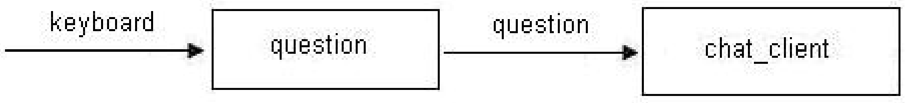
- **Issues:** None.

- **Constraints:** None.

question

- **Narrative:** Receives the question string from the keyboard as the input. It sends the question string to the chat_client object.

- **Diagram:**



- **Interface:** This component interfaces with the keyboard and chat_client objects.

- **Functions:** GetQuestion(), SendQuestion()

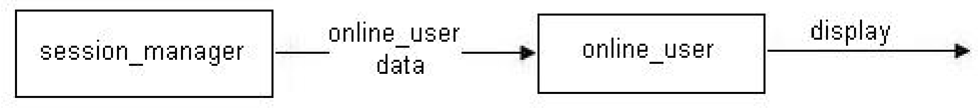
- **Issues:** None.

- **Constraints:** None.

online_users

- **Narrative:** Receives the online users data from the session_manger object as the input. It displays the clients who are online.

- **Diagram:**



- **Interface:** This component interfaces with the session_manager object.

- **Functions:** DisplayOnlineUsers()

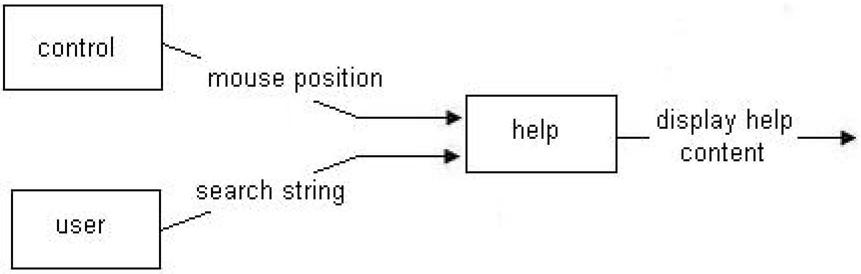
- **Issues:** None.

- **Constraints:** None.

help

- **Narrative:** Receives the mouse position from the control component or the search string from the user as the inputs. It searches the database and displays the help content dynamically.

- **Diagram:**



- **Interface:** This component interfaces with the database and control components.

- **Functions:** ConnectDatabaseServer(), Search(), DisplayHelpContent()

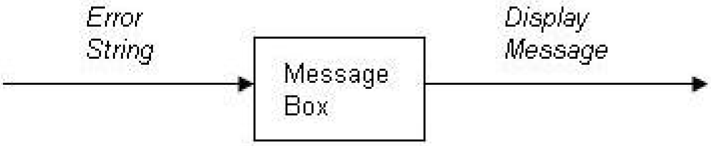
- **Issues:** The help content for all Mouse positions must be defined correctly, otherwise, the help will not work properly.

- **Constraints:** If the search string is not found in the database, nothing will be displayed.

error

- **Narrative:** Receives exceptions as the input. It handles these.

- **Diagram:**



- **Interface:** This component interfaces with all of the components.

- **Functions:** GetException(), HandleException()

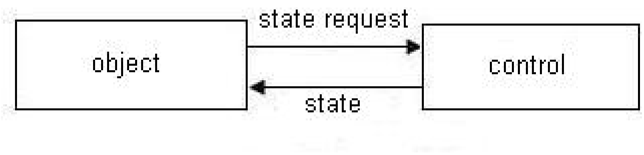
- **Issues:** If the errors are not handled, the program may crash.

- **Constraints:** All the error types should be handled correctly.

control

- **Narrative:** Receives state requests from the objects as input. It holds the buttons' and components' states and sends these data to those objects.

  All the necessary conditions must be satisfied when a job is to be performed. For instance, when a client wants to attend a lecture, that lecture must be given at that moment, he/she must be online and he/she must be registered for that course. All these necessary conditions are stored in this component and these conditions are checked before a job is to be performed by using this component.
- **Diagram:**



- **Interface:** This component interfaces with all of the components.

- **Functions:** GetRequest(), SendState()

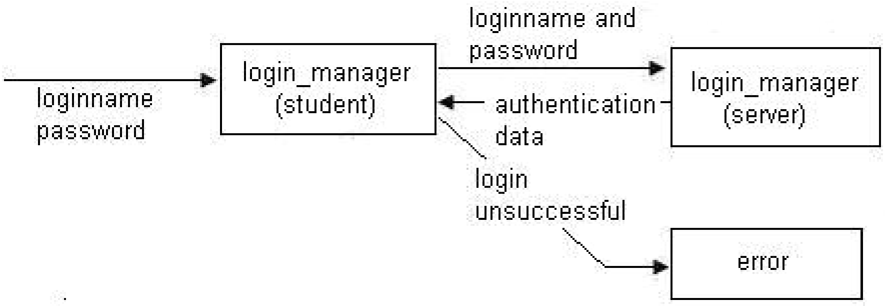
- **Issues:** None.

- **Constraints:** None.

login_manager

- **Narrative:** Receives the login name and password from the user. It sends these to the login manager on the server side. If the login operation is unsuccessful, it sends error message to the error component.

- **Diagram:**



- **Interface:** This component interfaces with the login_manager (server side) and error objects.

- **Functions:** GetLoginPassword(), SendLoginPassword(), HandleAuthInfo().

- **Issues:** If the login is unsuccessful, there will be error.

- **Constraints:** If the user is already online, then the login manager prevents logging in with the same userID and password.

newsgroup

- **Narrative:** Receives no input. This is the newsgroup on the web.

- **Diagram:** None.

- **Interface:** None (It Works independently).

- **Functions:** None.

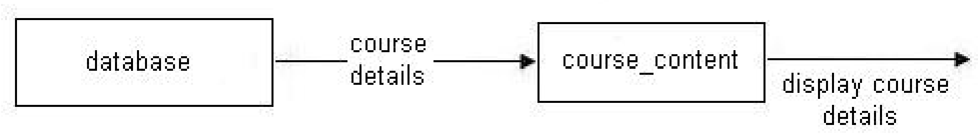
- **Issues:** None.

- **Constraints:** None.

course_content

- **Narrative:** Receives the course details from object handler on the server side. Then the lecture is started accordingly (online or offline) by the object handler on the student side.

- **Diagram:**



- **Interface:** This component interfaces with the database component and the object handler component.

- **Functions:** GetCourseDetail(), SendCourseDetail(), EnterLecture().

- **Issues:** None.

- **Constraints:** None.

Details of Processes

Broadcast:

Generally, broadcasting is the transmission of the realtime streams from one machine to another. There are three main broadcasting in our project:

Audiovisual Broadcast:

Audiovisual broadcast starts with clicking the “broadcast” button by the instructor. When the instructor clicks this button, an instance of windows media encoder is created. The encoder gets the realtime audiovisual data from the camera and the microphone. Then it converts this data into audiovisual stream and sends this stream to the windows media server. In the server, the coming stream is broadcasted via the broadcast point that is created previously for that course.

In the student side, when the student wants to attend an online lecture, a new windows media player instance is created. The URL address (the address of the broadcast point on the server) for the audiovisual broadcast is assigned to this instance. The player can directly play the coming stream from the server.

Whiteboard Broadcast:

Whiteboard broadcast starts with clicking the “broadcast” button by the instructor. The content of the whiteboard is sent periodically to the server. On the server, the data coming from the instructor’s computer is sent to the clients. This data is also saved on the disk on the server for offline usage.

In the student side, when the student enters the system, a new whiteboard instance is created. It gets the whiteboard content coming from the server and decomposes it (draws the content on the whiteboard).

MultimediaBoard Broadcast:

As in the audiovisual broadcast, the broadcast point for the multimediaboard is created on the server previously. The process again starts with clicking the “broadcast” button by the instructor. When the instructor clicks this button, an instance of windows media encoder is created. The broadcast event starts with clicking the “share” buton on the multimediaboard. The encoder captures the multimediaboard window and converts it into stream. Then it sends this stream to the windows media server. In the server, the coming stream is broadcasted via the broadcast point that is created previously for the multimediaboard.

In the student side, when the student wants to attend an online lecture, a new windows media player instance is created. The URL address (the address of the broadcast point on the server) for the multimediaboard broadcast is assigned to this instance. The player displays this stream.

Chat:

When the server application starts, the chat server starts on the server side. On the client side, when the client enters the system, a new chat client is created. The chat client connects to the chat server. There are three cases for chatting:

- When the client writes message and clicks the ”send all” button, the message is sent to the chat server. The chat server sends this message to all the clients. On the client side, the chat client displays the coming message on the public chat area.
- When the client writes message and clicks the ”send priv” button, the message is sent to the chat server. The chat server sends this message to the specified client. On the client side, the chat client displays the coming message on the private chat area.
- When the client writes message and clicks the ”question” button, the message is sent to the chat server. The chat server sends this message to the instructor. On the instructor side, the chat client displays the coming question on the question queue.

Synchronization:

Synchronization is the most critical part. Two types of synchronization should be considered in this project:

- The synchronization between audio and video will be performed by the windows media server.
- The synchronization among the audio-video, whiteboard and multimediaboard will be performed by using the timestamps. For instance, if the audio-video is sent at a rate of 30 frames/second and if the whiteboard content is refreshed every 3 seconds, then the whiteboard content will be displayed if the timestamp of the audio-video is 30 * k (where k = 1, 2, 3, …). If latency occurs, for example in audio video delivery, the whiteboard content will not be refreshed until (30 * k)th frame of the audio-video is arrived. The reverse of this scenario is also valid. If latency occurs in the delivery of the whiteboard content, the next 30 frames of the audio-video will not be displayed until whiteboard content is arrived. The same idea will be used in the synchronization of the multimediaboard.

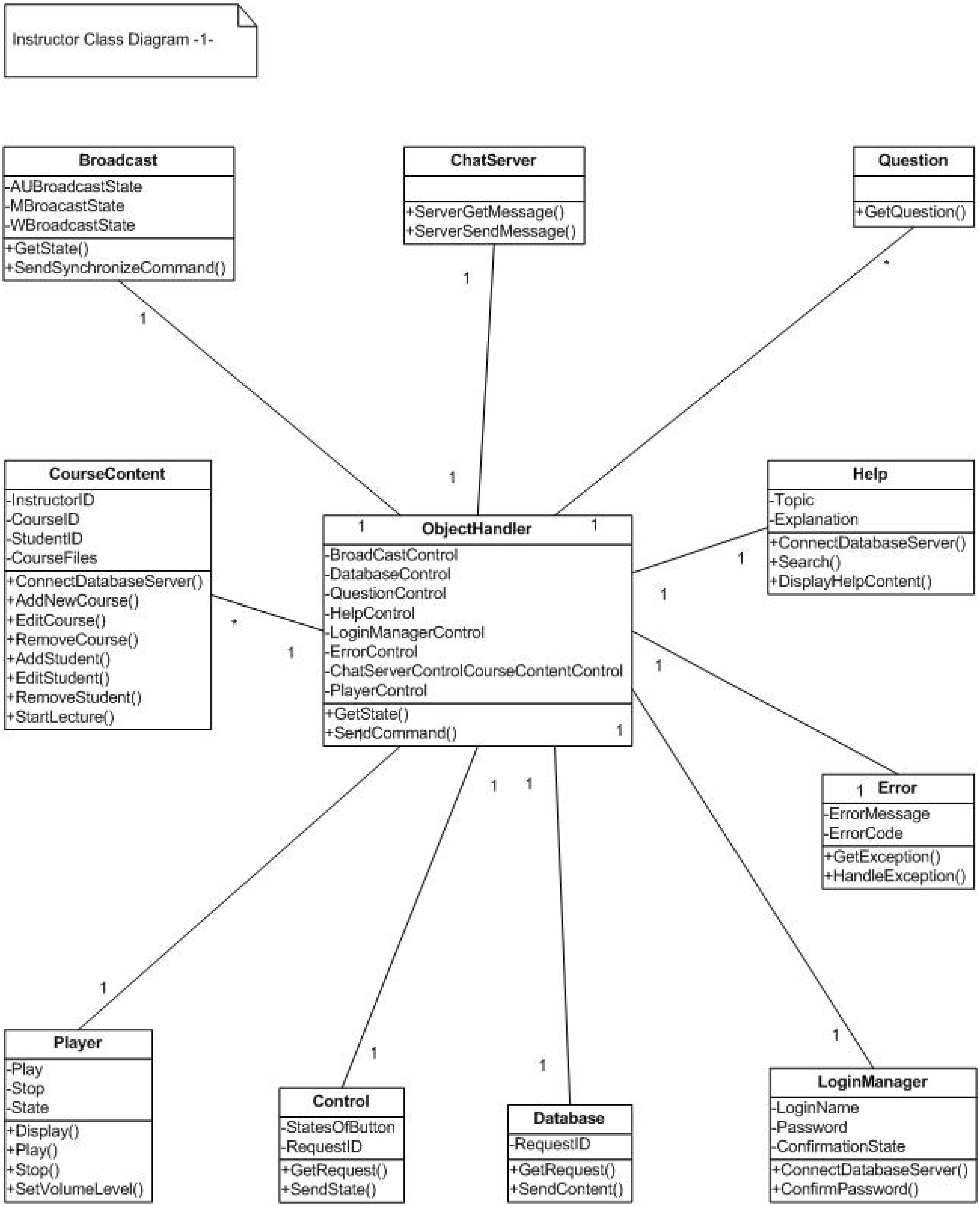
Software Interface Description

The interface that we have chosen to use will be designed in Microsoft Visual C# .Net. The interface will utilize many of the common controls included in Visual Visual C# .Net, and the majority of windows-based applications. The interface will be a graphical user interface that provides the instructor to give lectures efficiently and that provides the user to follow the lectures easily. For a more detailed description of the interface, please refer to the User Interface Design section found below.

4. UML DIAGRAMS

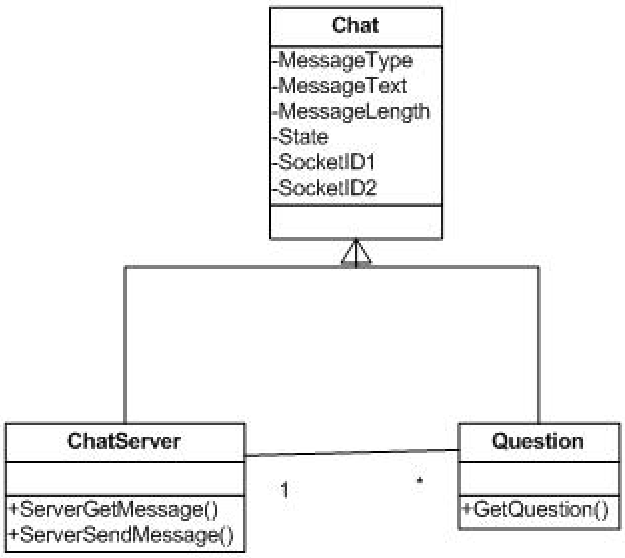
Class Diagrams

Instructor Class Diagram

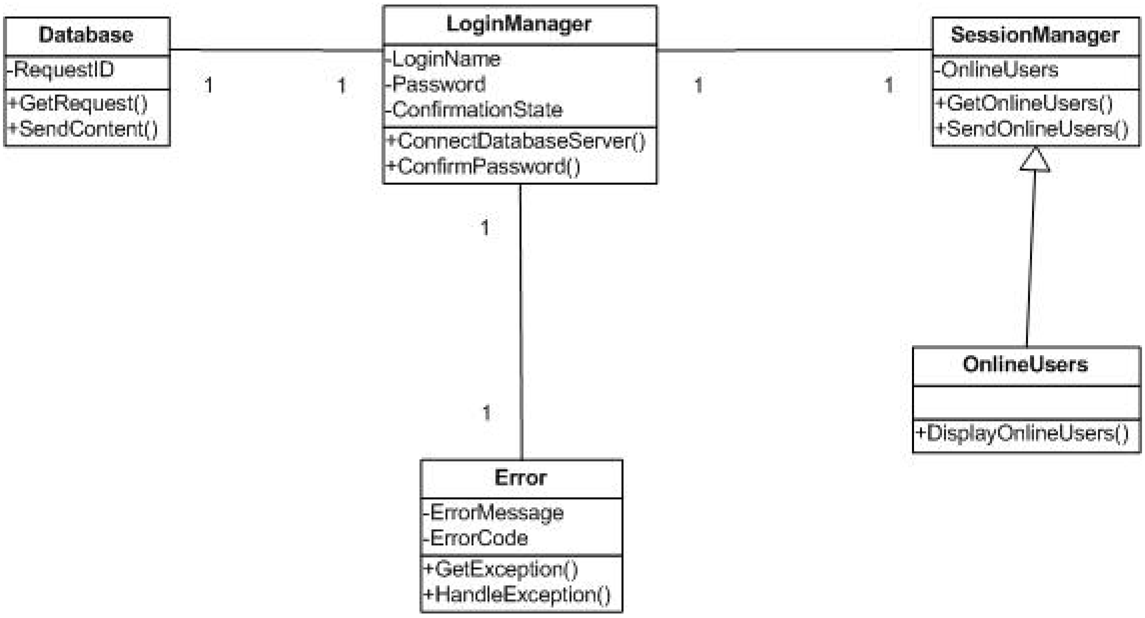


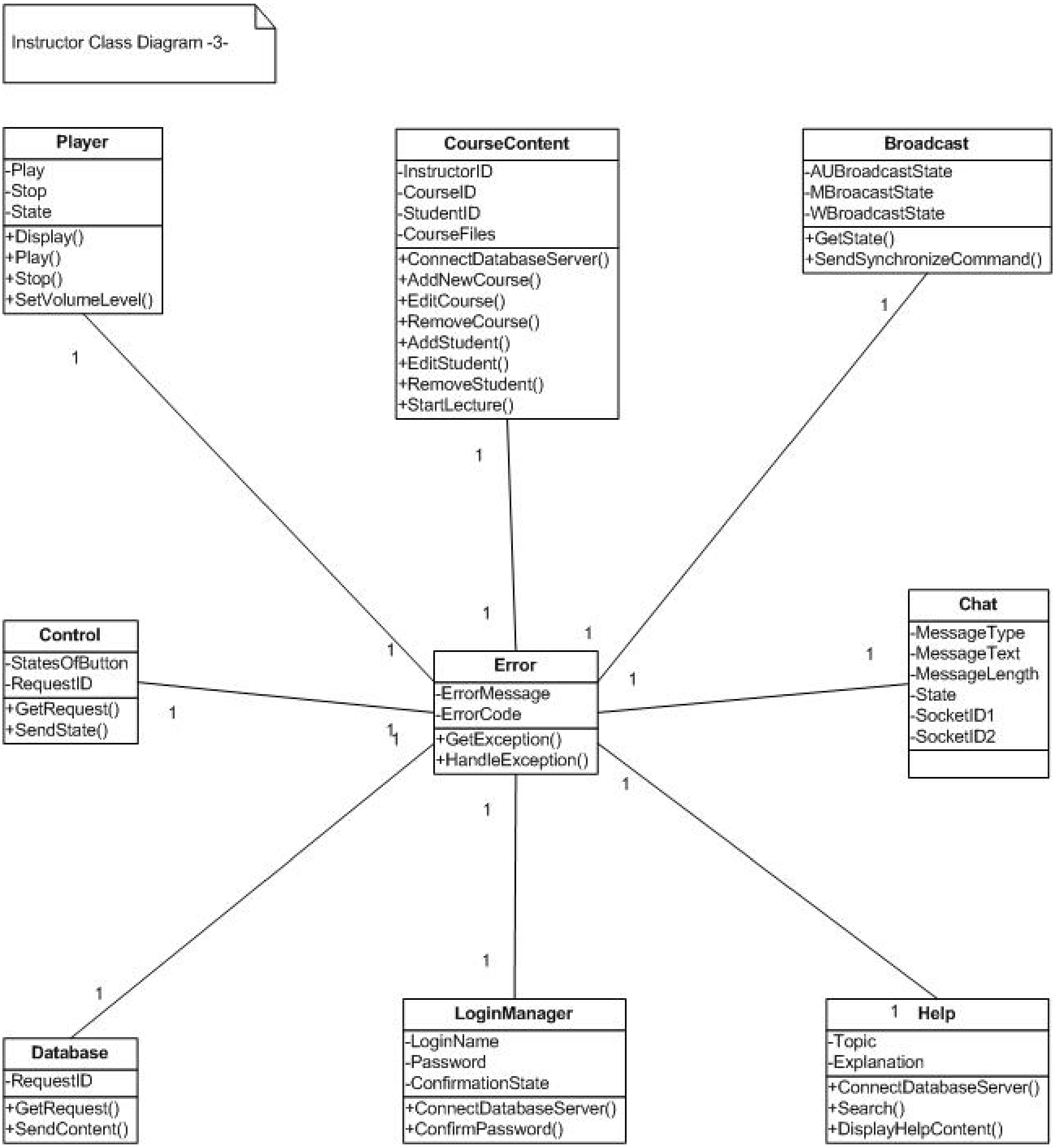
Instructor Class Diagram -2-

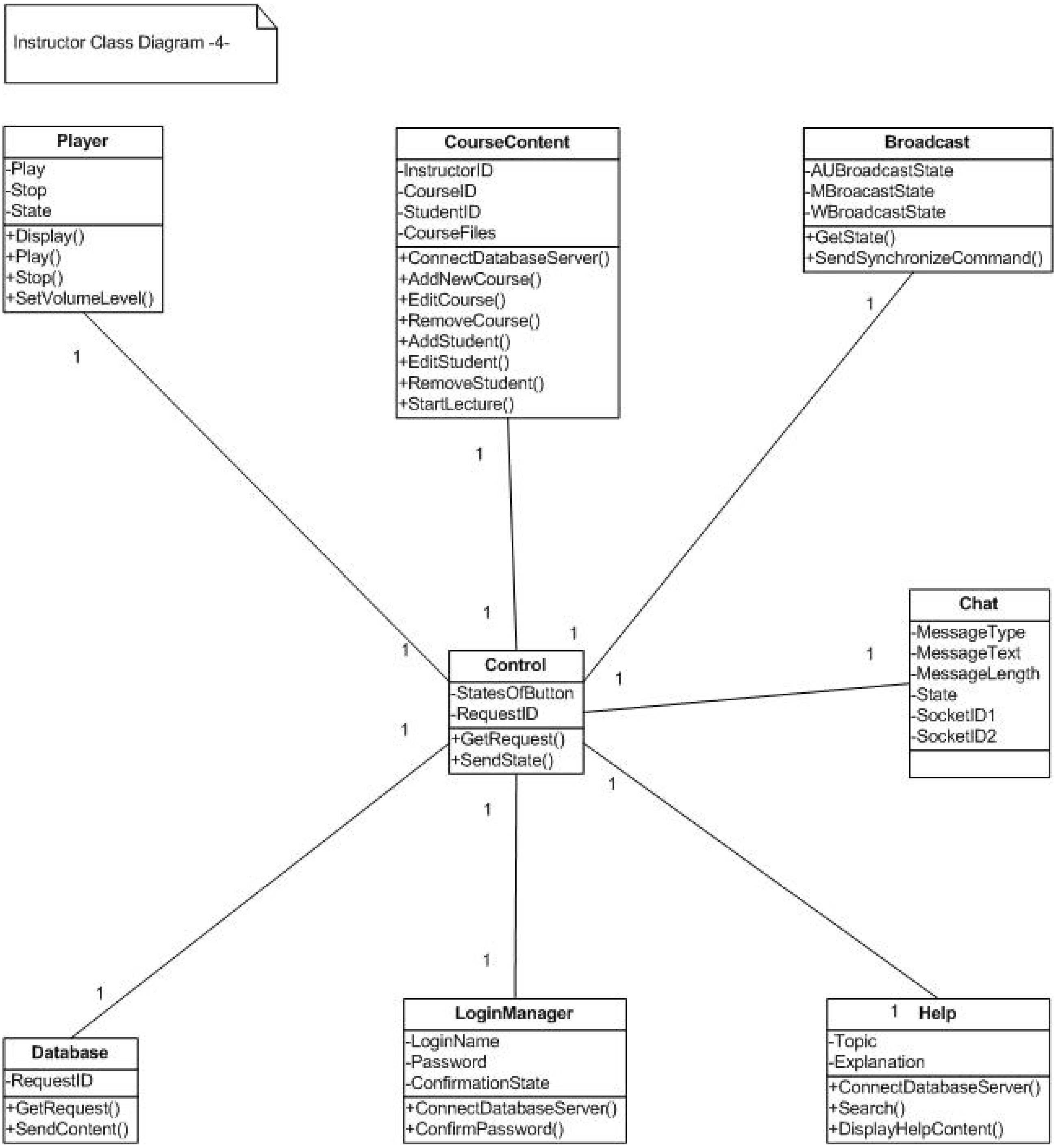
Chat

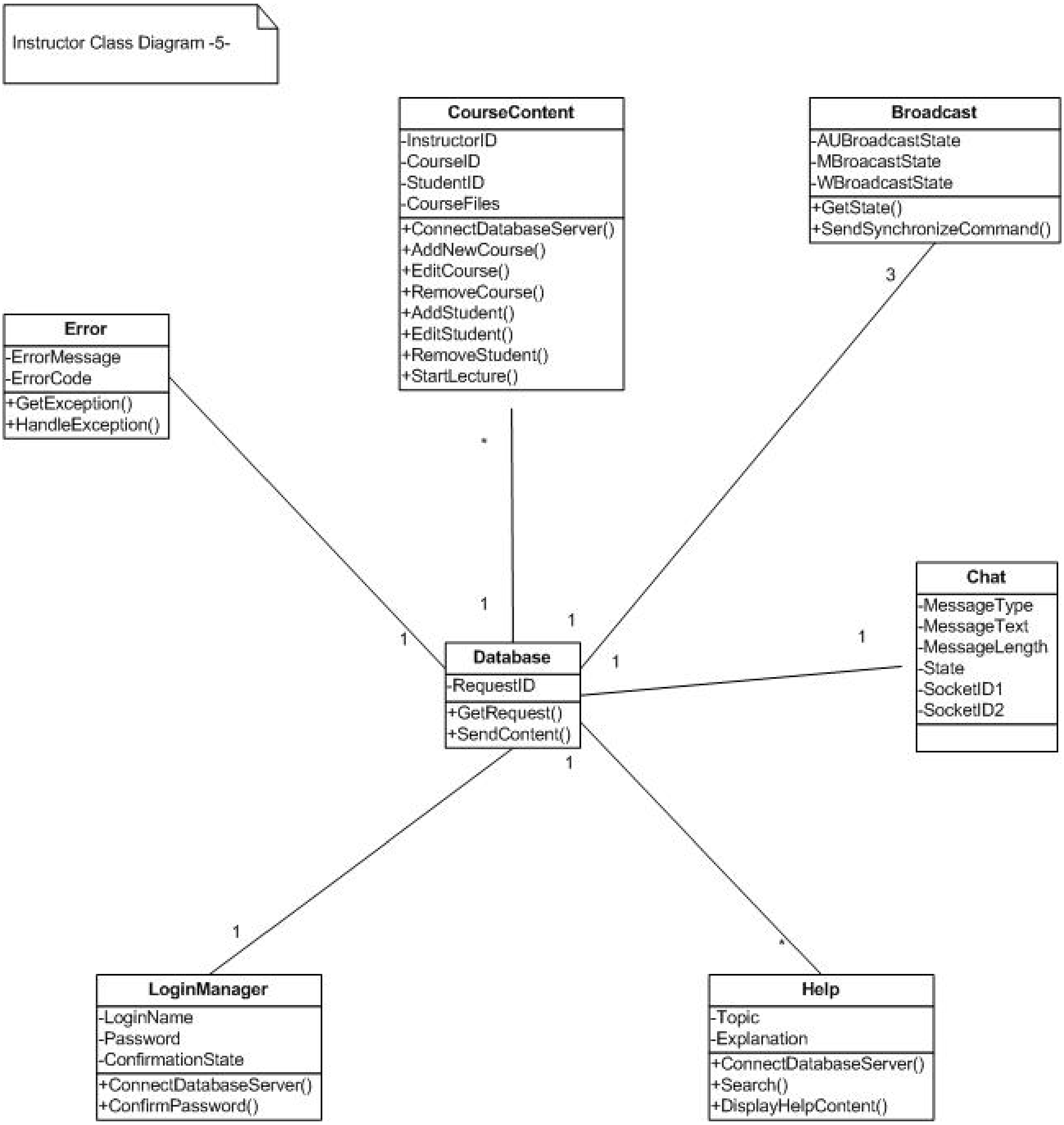


Login Manager

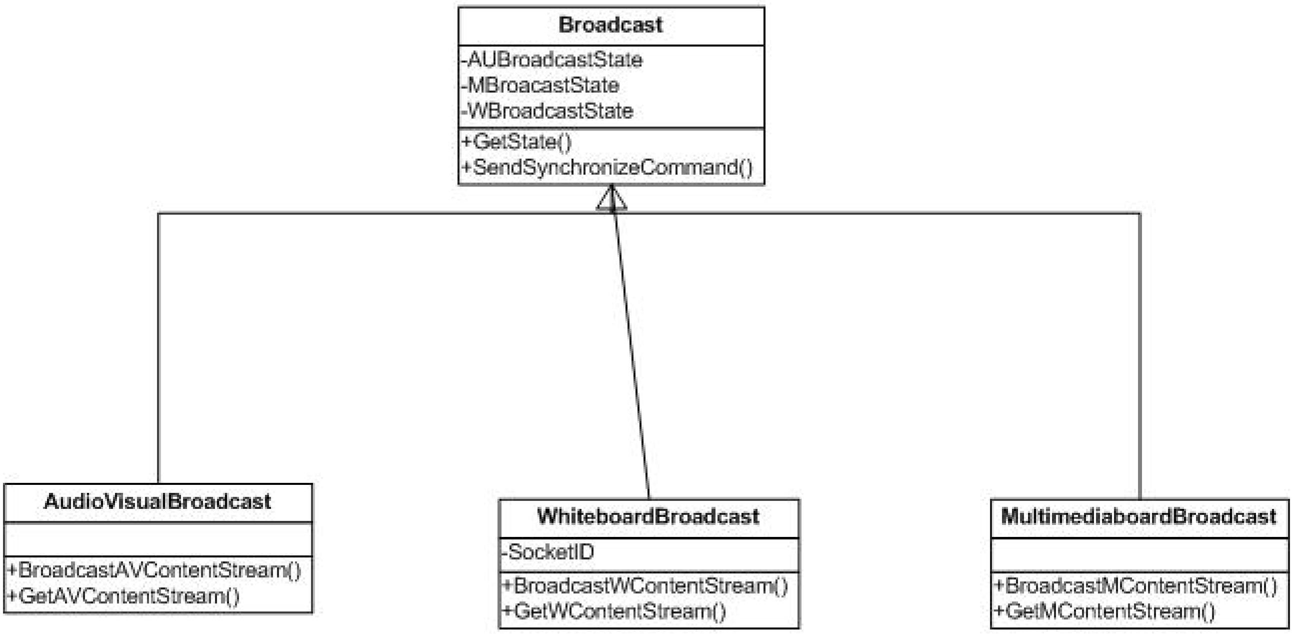



Instructor Class Diagram -3-
Player
-Play
-Stop
-State
+Display()
+Play()
+Stop()
+SetVolumeLevel()
CourseContent
-InstructorID
-CourseID
-StudentID
-CourseFiles
+ConnectDatabaseServer()
+AddNewCourse()
+EditCourse()
+RemoveCourse()
+AddStudent()
+EditStudent()
+RemoveStudent()
+StartLecture()
Broadcast
-AUBroadcastState
-MBroacastState
-WBroadcastState
+GetState()
+SendSynchronizeCommand()
Control
-StatesOfButton
-RequestID
+GetRequest()
+SendState()
Error
-ErrorMessage
-ErrorCode
+GetException()
+HandleException()
Chat
-MessageType
-MessageText
-MessageLength
-State
-SocketID1
-SocketID2
Database
-RequestID
+GetRequest()
+SendContent()
LoginManager
-LoginName
-Password
-ConfirmationState
+ConnectDatabaseServer()
+ConfirmPassword()
Help
-Topic
-Explanation
+ConnectDatabaseServer()
+Search()
+DisplayHelpContent()
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1

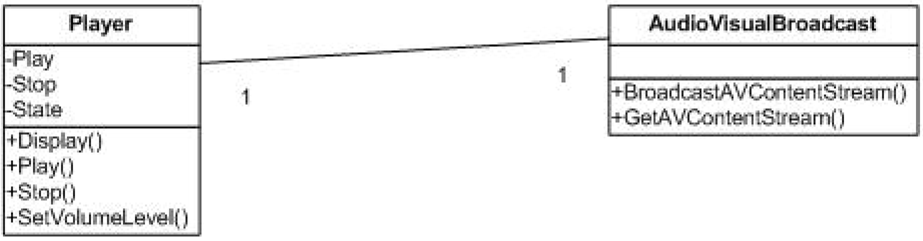

Instructor Class Diagram -4-
Player
-Play
-Stop
-State
+Display()
+Play()
+Stop()
+SetVolumeLevel()
CourseContent
-InstructorID
-CourseID
-StudentID
-CourseFiles
+ConnectDatabaseServer()
+AddNewCourse()
+EditCourse()
+RemoveCourse()
+AddStudent()
+EditStudent()
+RemoveStudent()
+StartLecture()
Broadcast
-AUBroadcastState
-MBroacastState
-WBroadcastState
+GetState()
+SendSynchronizeCommand()
Chat
-MessageType
-MessageText
-MessageLength
-State
-SocketID1
-SocketID2
Control
-StatesOfButton
-RequestID
+GetRequest()
+SendState()
Database
-RequestID
+GetRequest()
+SendContent()
LoginManager
-LoginName
-Password
-ConfirmationState
+ConnectDatabaseServer()
+ConfirmPassword()
Help
-Topic
-Explanation
+ConnectDatabaseServer()
+Search()
+DisplayHelpContent()
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1


Instructor Class Diagram -5-
CourseContent
-InstructorID
-CourseID
-StudentID
-CourseFiles
+ConnectDatabaseServer()
+AddNewCourse()
+EditCourse()
+RemoveCourse()
+AddStudent()
+EditStudent()
+RemoveStudent()
+StartLecture()
Broadcast
-AUBroadcastState
-MBroacastState
-WBroadcastState
+GetState()
+SendSynchronizeCommand()
3
Error
-ErrorMessage
-ErrorCode
+GetException()
+HandleException()
*
1
1
1
1
1
Chat
-MessageType
-MessageText
-MessageLength
-State
-SocketID1
-SocketID2
Database
-RequestID
+GetRequest()
+SendContent()
1
1
*
LoginManager
-LoginName
-Password
-ConfirmationState
+ConnectDatabaseServer()
+ConfirmPassword()
Help
-Topic
-Explanation
+ConnectDatabaseServer()
+Search()
+DisplayHelpContent()

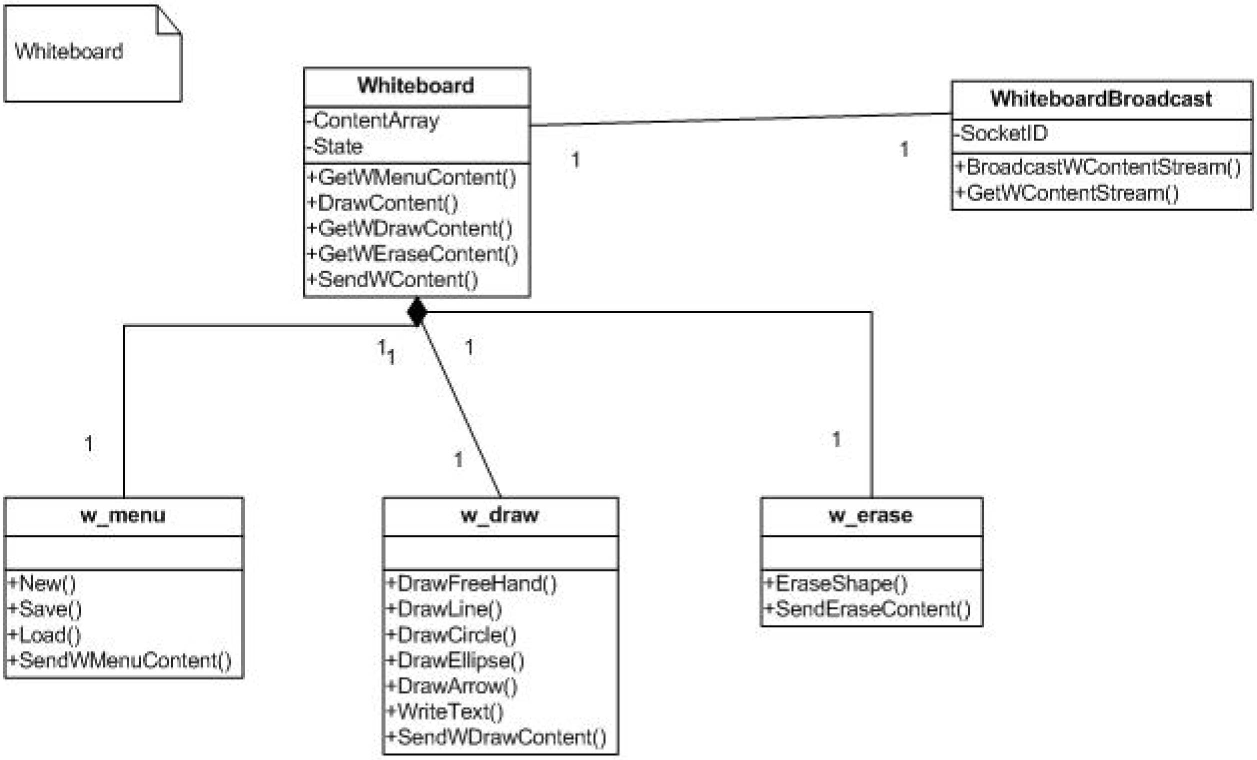
Instructor Class Diagram -6-

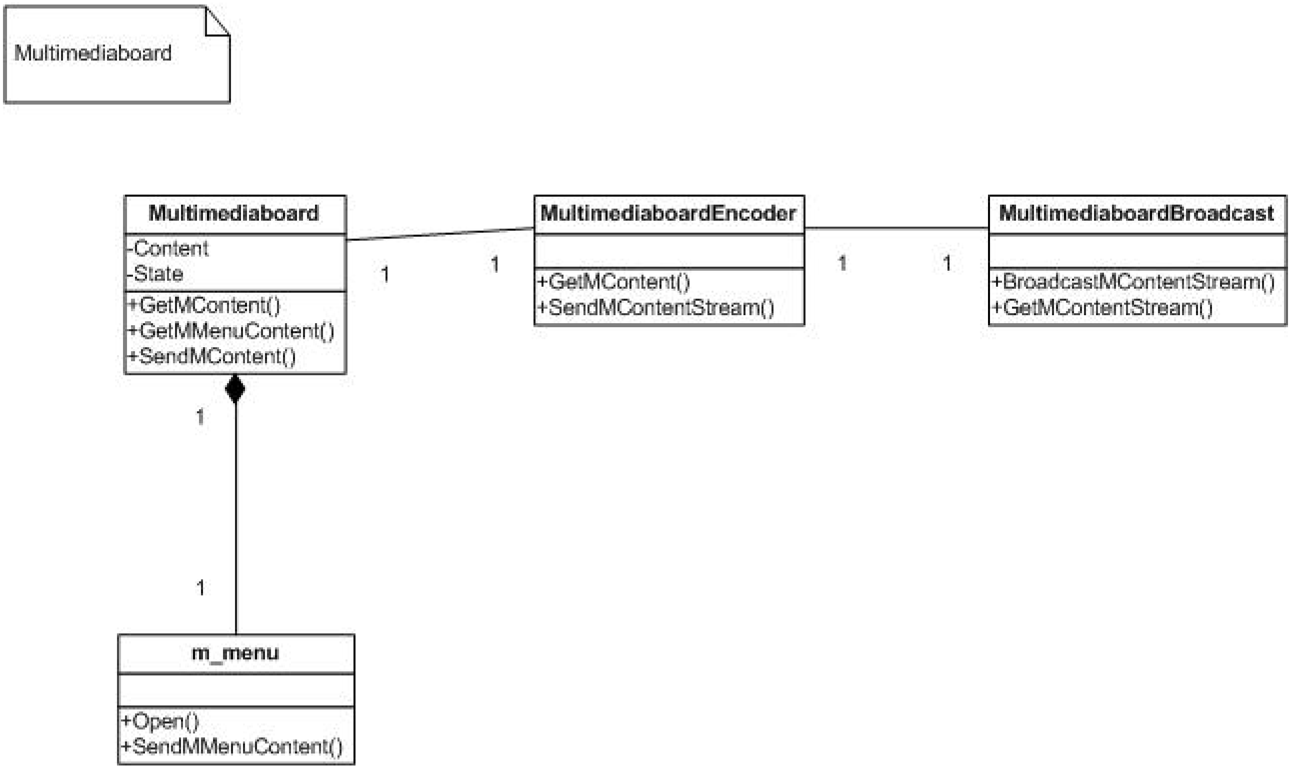


Player

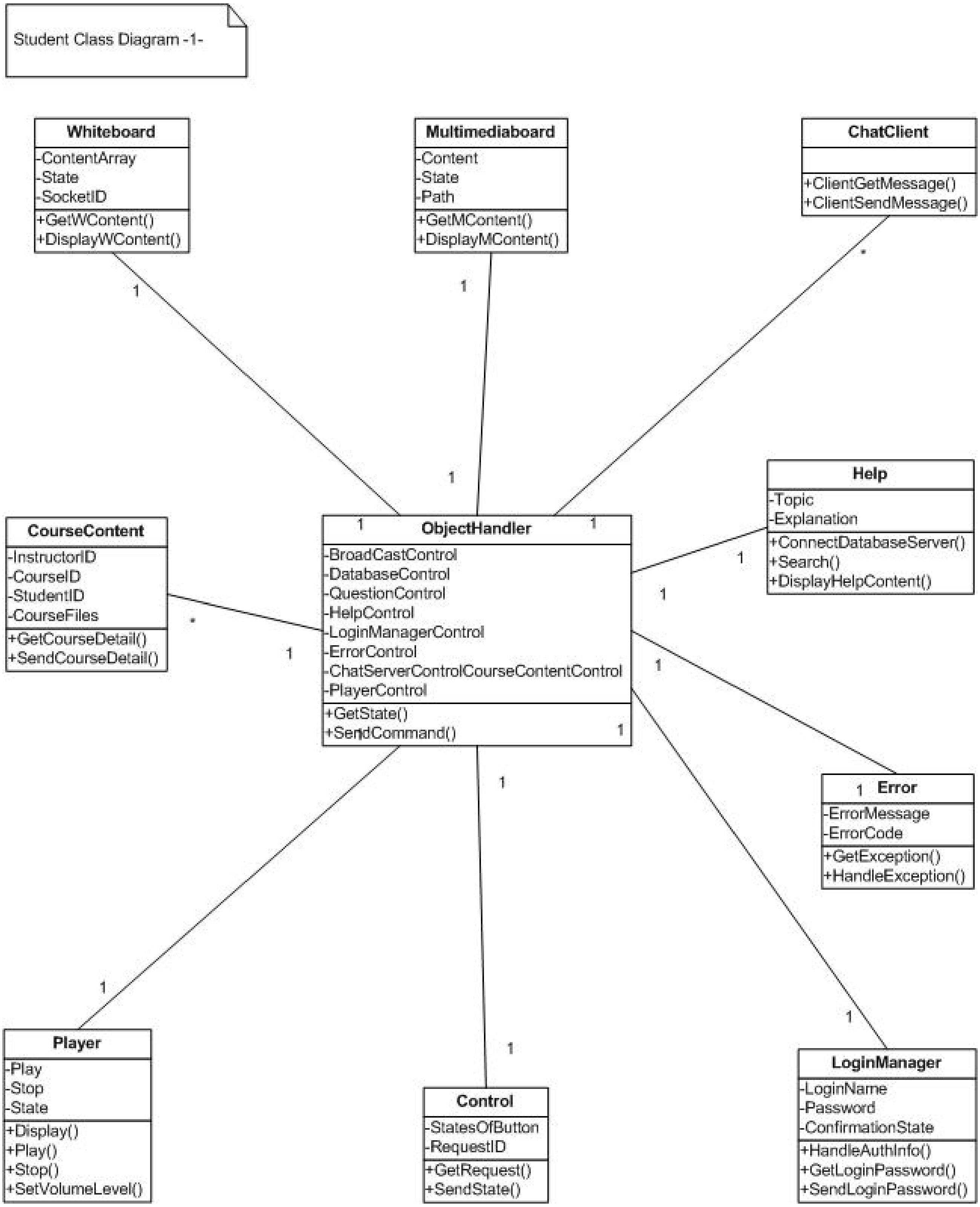


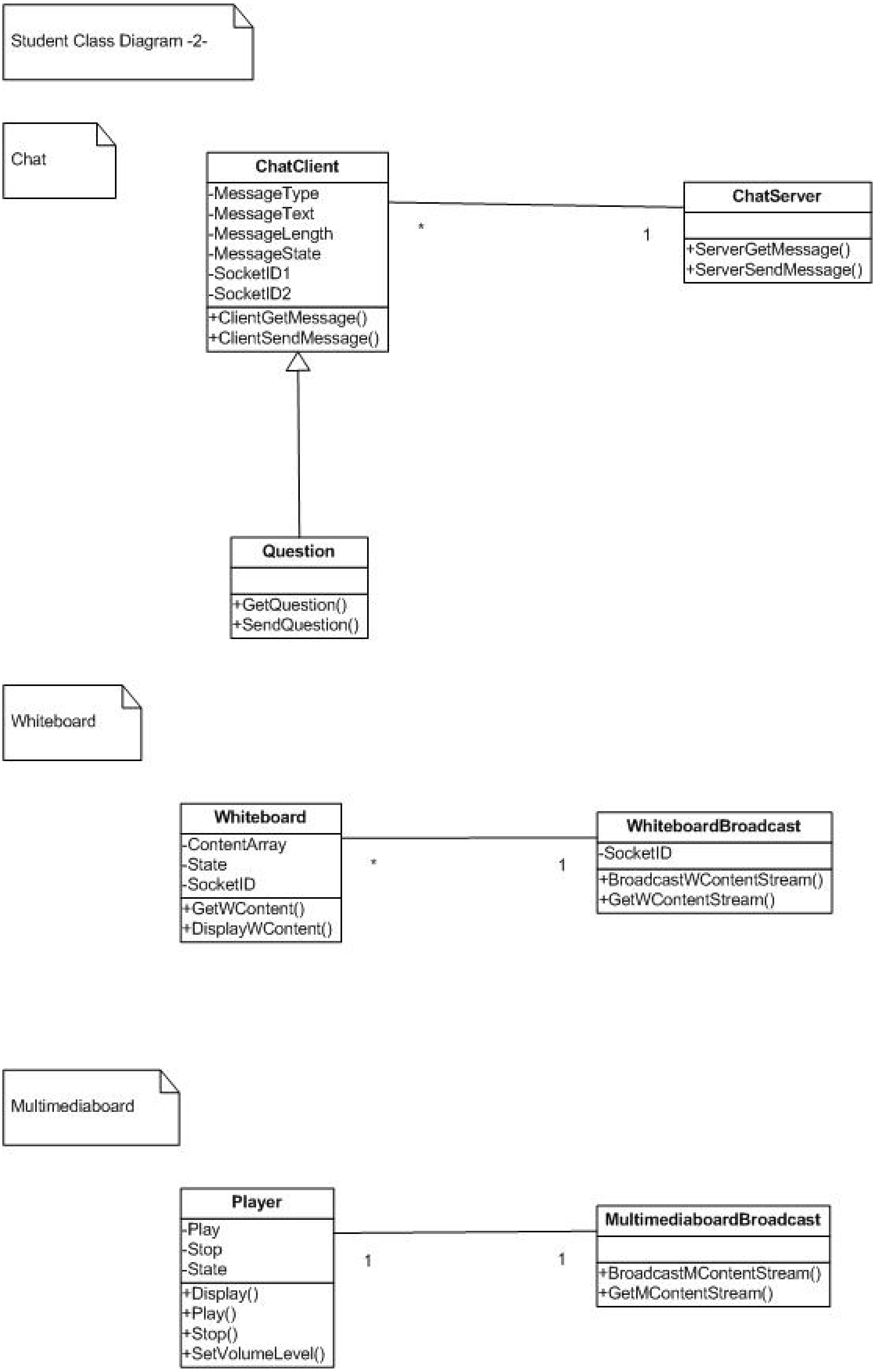
Instructor Class Diagram -7-



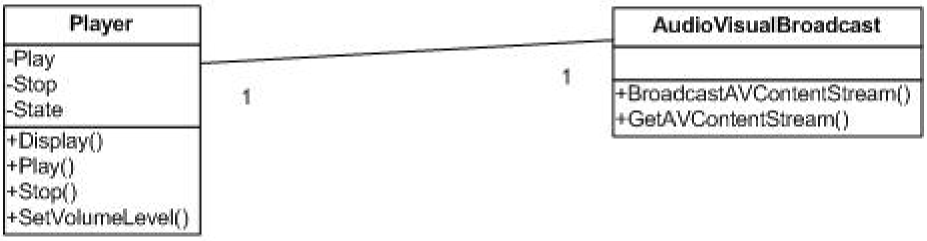


Student Class Diagram

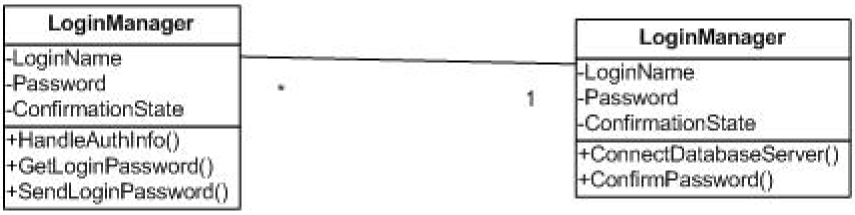



Student Class Diagram -2-
Chat
ChatClient
-MessageType
-MessageText
-MessageLength
-MessageState
-SocketID1
-SocketID2
+ClientGetMessage()
+ClientSendMessage()
*
1
ChatServer
+ServerGetMessage()
+ServerSendMessage()
Question
+GetQuestion()
+SendQuestion()
Whiteboard
Whiteboard
-ContentArray
-State
-SocketID
+GetWContent()
+DisplayWContent()
*
1
WhiteboardBroadcast
-SocketID
+BroadcastWContentStream()
+GetWContentStream()
Multimediaboard
Player
-Play
-Stop
-State
+Display()
+Play()
+Stop()
+SetVolumeLevel()
1
1
MultimediaboardBroadcast
+BroadcastMContentStream()
+GetMContentStream()

Student Class Diagram -3-

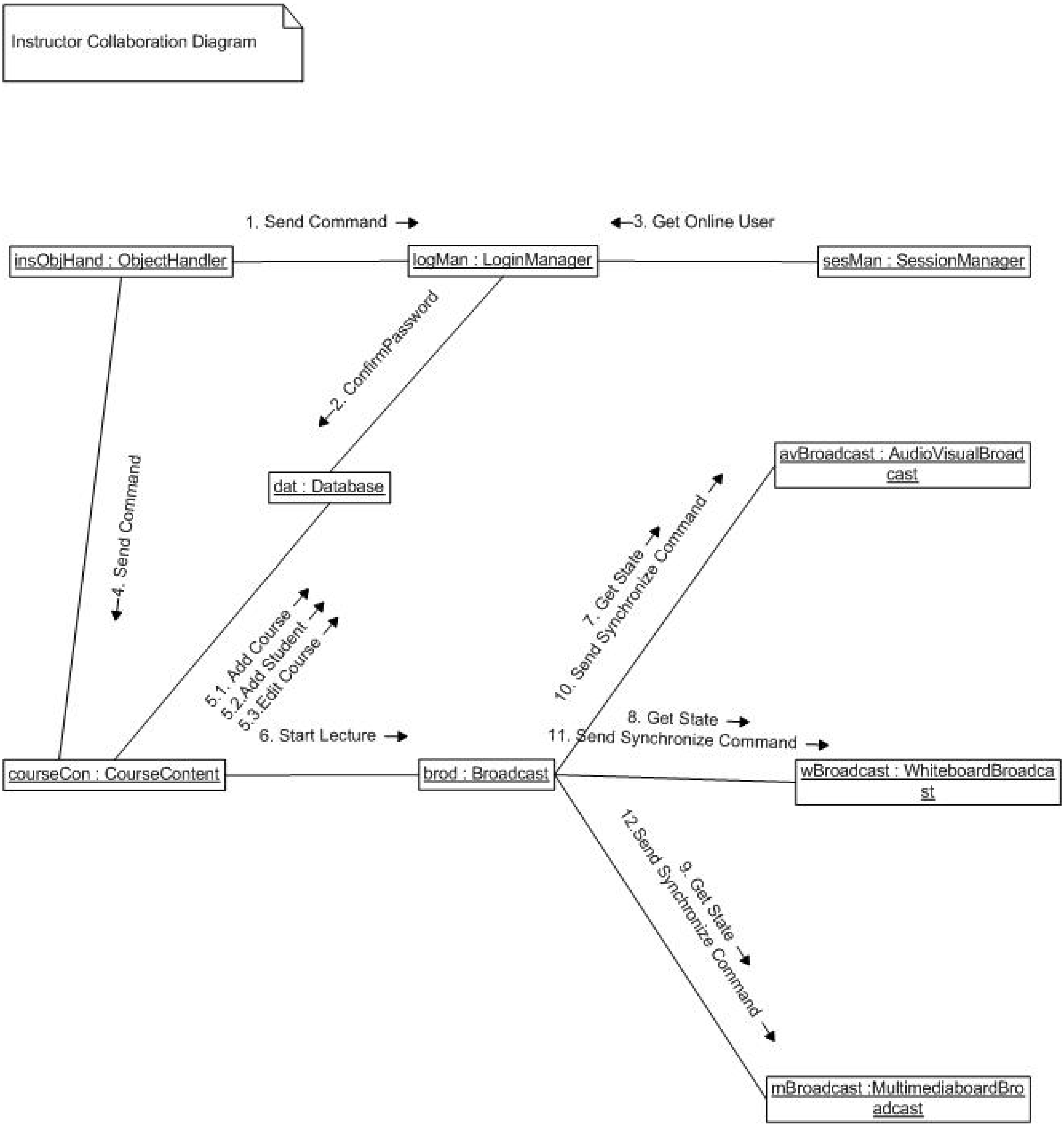


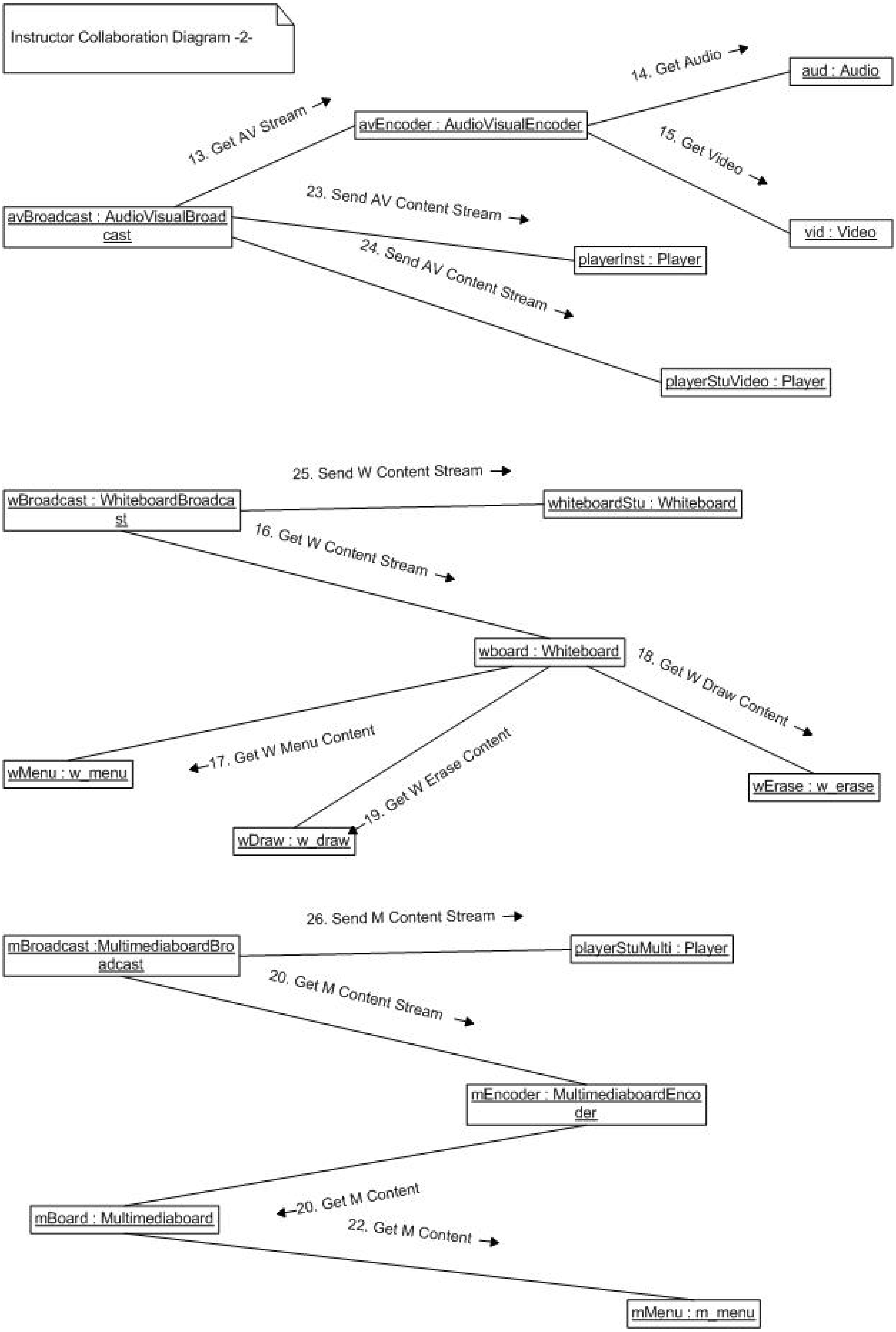
Login Manager



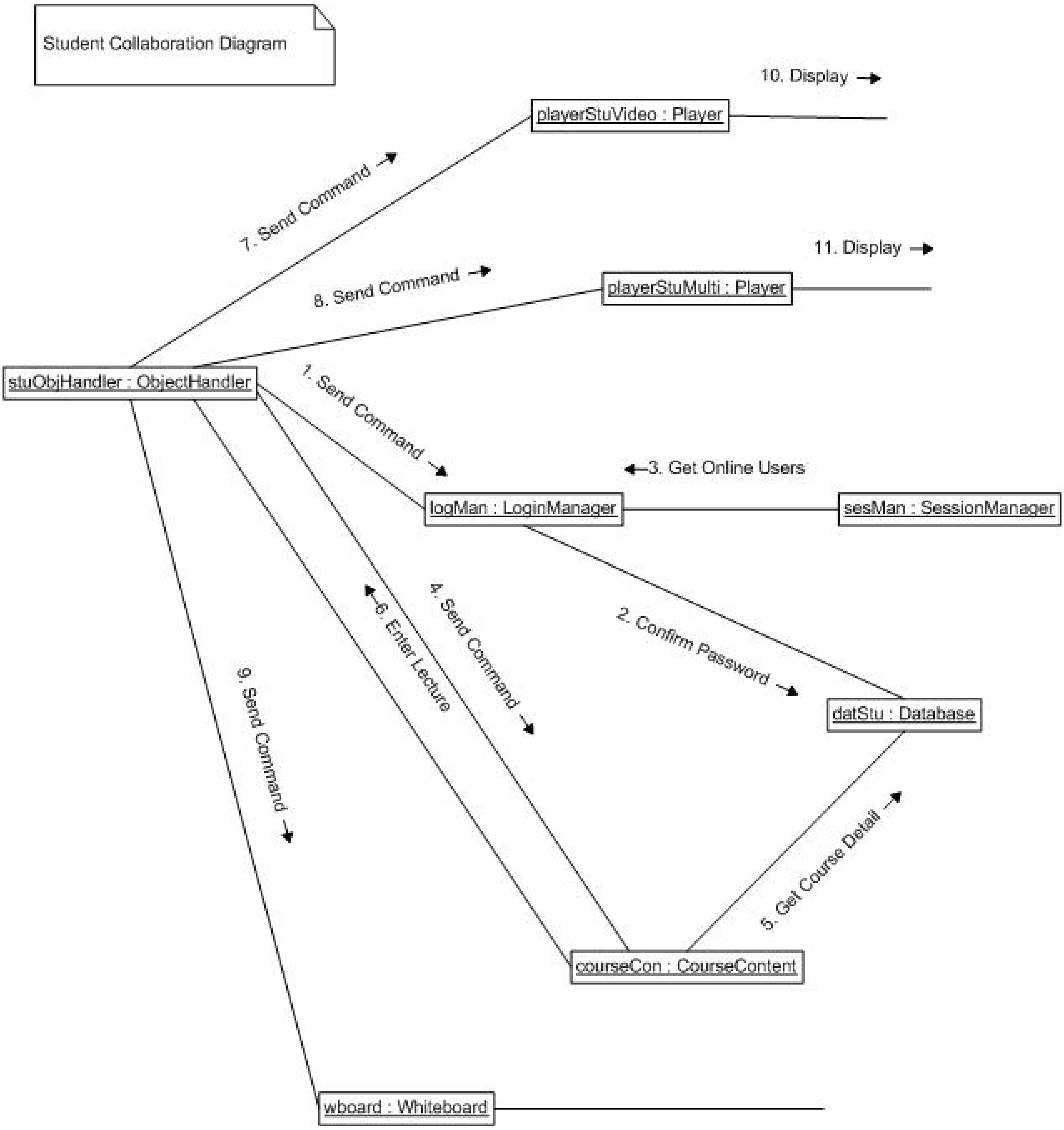
Collaboraiton Diagrams

Instructor Collaboration Diagram


Instructor Collaboration Diagram
1. Send Command
3. Get Online User
insObjHand : ObjectHandler
logMan : LoginManager
sesMan : SessionManager
2. ConfirmPassword
dat : Database
4. Send Command
avBroadcast : AudioVisualBroadcast
7. Get State
10. Send Synchronize Command
5.1. Add Course
5.2.Add Student
5.3.Edit Course
6. Start Lecture
8. Get State
11. Send Synchronize Command
courseCon : CourseContent
brod : Broadcast
wBroadcast : WhiteboardBroadcast
9. Get State
12.Send Synchronize Command
mBroadcast :MultimediaboardBroadcast


Instructor Collaboration Diagram -2-
avBroadcast : AudioVisualBroadcast
avEncoder : AudioVisualEncoder
aud : Audio
vid : Video
playerInst : Player
playerStuVideo : Player
13. Get AV Stream
14. Get Audio
15. Get Video
23. Send AV Content Stream
24. Send AV Content Stream
wBroadcast : WhiteboardBroadcast
whiteboardStu : Whiteboard
wboard : Whiteboard
wMenu : w_menu
wDraw : w_draw
wErase : w_erase
25. Send W Content Stream
16. Get W Content Stream
17. Get W Menu Content
18. Get W Draw Content
19. Get W Erase Content
mBroadcast :MultimediaboardBroadcast
playerStuMulti : Player
mEncoder : MultimediaboardEncoder
mBoard : Multimediaboard
mMenu : m_menu
26. Send M Content Stream
20. Get M Content Stream
20. Get M Content
22. Get M Content

Student Collaboration Diagram



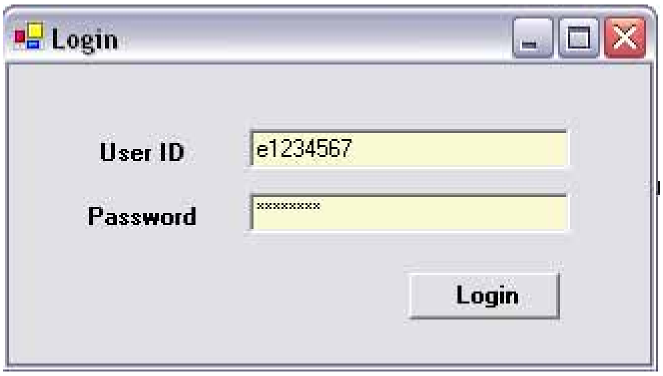
5. USER INTERFACE DESIGN

Description of the User Interface

There are two types of user interfaces: instructor interface and student interface. These interfaces show all the contents (player, multimediaboard, whiteboard, etc) simultaneously. In the instructor interface broadcasting process is started by using the “broadcast” command in the main menu or context menu. There is a question queue window in this interface for intrustor to see the questions asked by the students. In the student interface, all the content sended by the multimedia server is displayed to the student. The student can chat with the other students or ask questions using the chat window. Moreover, there is a help window in both of the interfaces.

Login Interface

The users (instructor and the students) log in the system by entering their userID and password in the login window.

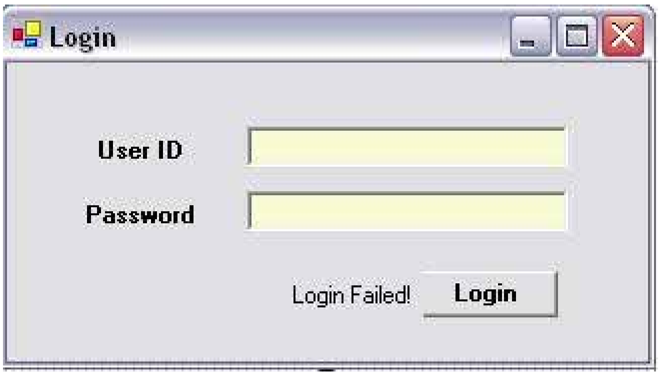


User ID: e1234567

Password: ********

Login Failed Interface

If login process fails, the login failed interface appears. The reason (wrong userID or password, the user is already online, database connection failure, etc.) for the failure is displayed in the interface.

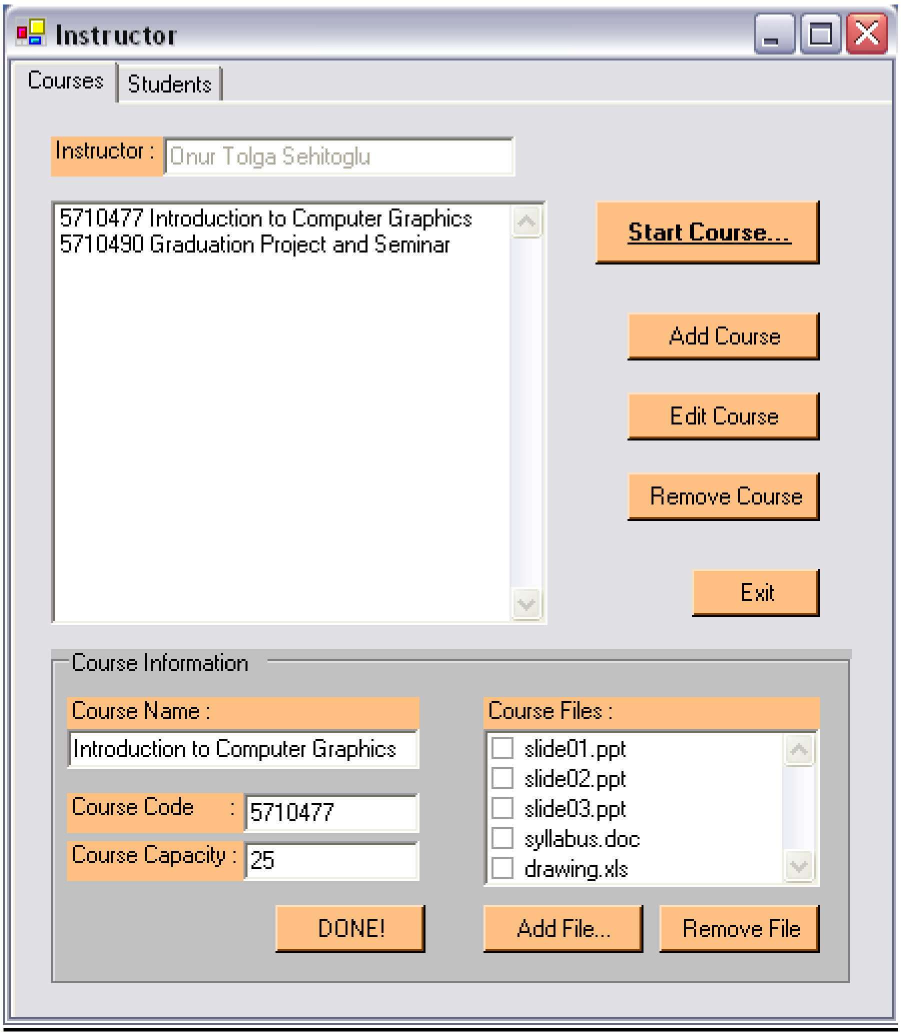


Login Failed!

The Instructor Pre - Lecture Interface

After the instructor log in the system by entering right user ID and password, the instructor pre-lecture interface appears which has two tabs, courses and students.

Courses tab



Instructor

Instructor : Onur Tolga Sehitoglu

5710477 Introduction to Computer Graphics
5710490 Graduation Project and Seminar

Course Information

Course Name : Introduction to Computer Graphics

Course Code : 5710477

Course Capacity : 25

Course Files :

- □ slide01.ppt
- □ slide02.ppt
- □ slide03.ppt
- □ syllabus.doc
- □ drawing.xls

Courses tab displays courses which are given by instructor entered the system. The course information part below is initially disabled and empty. There are four main course operations on this tab:

Add new course:

After the instructor presses the ‘add course’ button, the text boxes on the ‘course information’ part allows the instructor to enter the name, code and capacity of the new course. Also instructor can add course related files which can be lecture slides, drawings, lecture notes, or other related materials to the database by pressing the ‘add file’ button or remove the selected ones from the database by pressing ‘remove file’ button. When all the text boxes are filled and new files are added, or removed, then the instructor presses the ‘done’ button to save them to the database as a newly created course. The new course is added to the course list.

Edit course:

After the instructor selects one course from the course list and presses the ‘edit course’ button, the text boxes on the ‘course information’ part allows the instructor to change the name, code and capacity of the already existed course. Also, when the instructor only selects the course, the course files appear in the list below, that the instructor can add new files there or remove some of them. When all the changes are done, then the instructor presses the ‘done’ button to save them to the database.

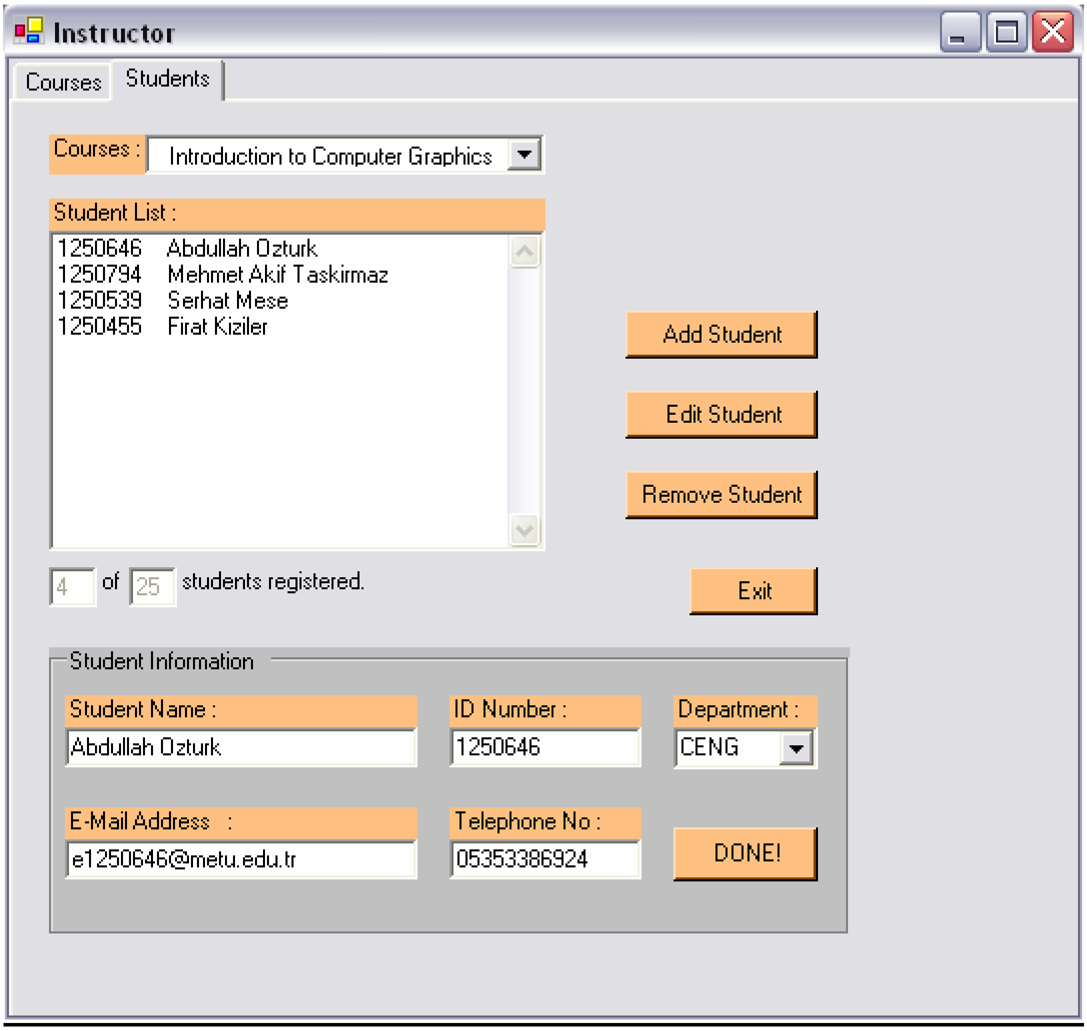
Remove course:

After the instructor selects one course from the course list and presses the ‘remove course’ button, the selected course is all deleted with its course files and past lecture replay files from the database. The course is removed from the course list.

Exit:

The instructor presses the ‘exit’ button to exit from the system.

Students tab



Courses : Introduction to Computer Graphics

Student List :

```
1250646   Abdullah Ozturk
1250794   Mehmet Akif Taskirmaz
1250539   Serhat Mese
1250455   Firat Kiziler
```

4 of 25 students registered.

Student Information

Student Name :	ID Number :	Department :
Abdullah Ozturk	1250646	CENG
E-Mail Address :	Telephone No :	
e1250646@metu.edu.tr	05353386924	

Students tab displays students registered to the course which are given by instructor entered the system after selecting the course from the combo box at top. The student information part below is initially disabled and empty. There are three main student operations on this tab:

Add student:

After the instructor presses the ‘add student’ button, the text boxes on the ‘student information’ part allows the instructor to enter the name, ID number, department, e-mail address and the telephone number of the new student. When all the text boxes are filled, then the instructor presses the ‘done’ button to save them to the database as a newly registered student to the course selected. The new student is added to the student list.

Edit student:

After the instructor selects one student from the student list of one course and presses the ‘edit student’ button, the text boxes on the ‘student information’ part allows the instructor to the name, ID number, department, e-mail address and the telephone number of the already existed student. When all the changes are done, then the instructor presses the ‘done’ button to save them to the database.

Remove student:

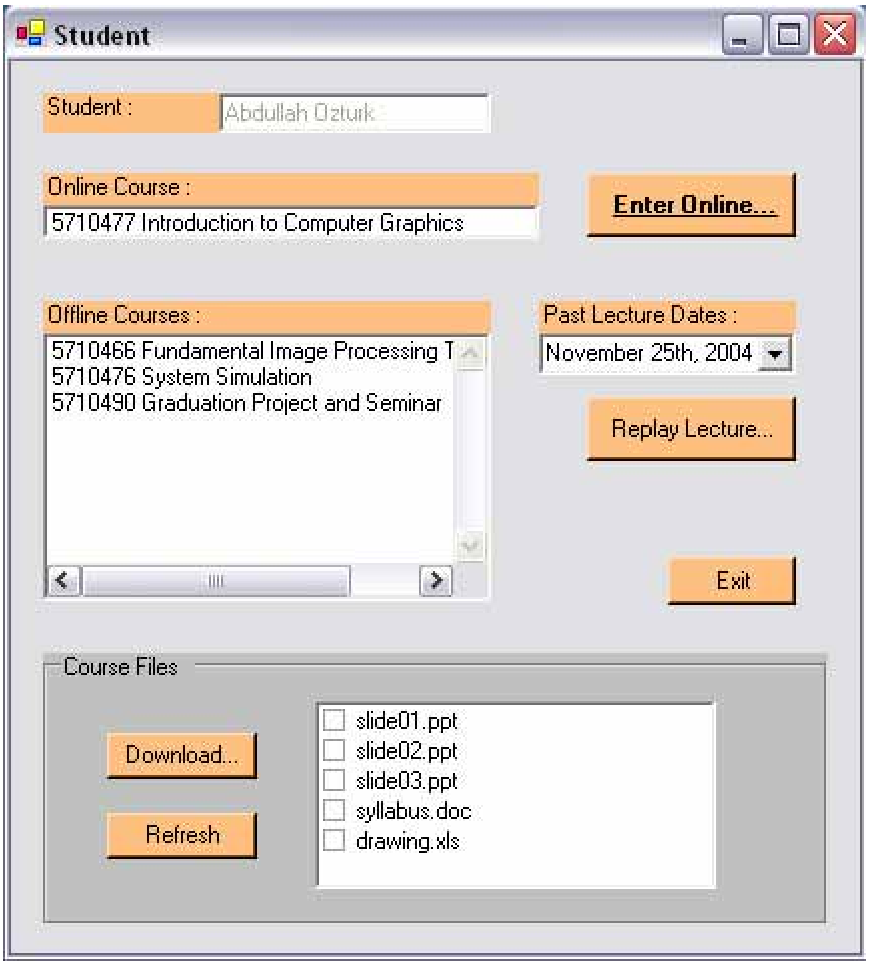
After the instructor selects one student from the student list and presses the ‘remove student’ button, the selected student is removed from the course, but the same student can still be registered other courses. The student is removed from the course list.

Exit:

The instructor presses the ‘exit’ button to exit from the system.

The Student Pre - Lecture Interface

After the student log in the system by entering right user ID and password, the student pre-lecture interface appears. In this window the student can see the courses to which he had been registered already.



Student : Abdullah Ozturk

Online Course : 5710477 Introduction to Computer Graphics

Offline Courses :
5710466 Fundamental Image Processing T
5710476 System Simulation
5710490 Graduation Project and Seminar

Past Lecture Dates : November 25th, 2004

Course Files
□ slide01.ppt
□ slide02.ppt
□ slide03.ppt
□ syllabus.doc
□ drawing.xls

Online Course

At any time, maximum one online lecture can be given by the system. If there is an online lecture at the moment the student enters the system, the online course section displays this online course. After selecting the course, the student presses the ‘enter online’ button to enter the online lecture.

Offline Courses

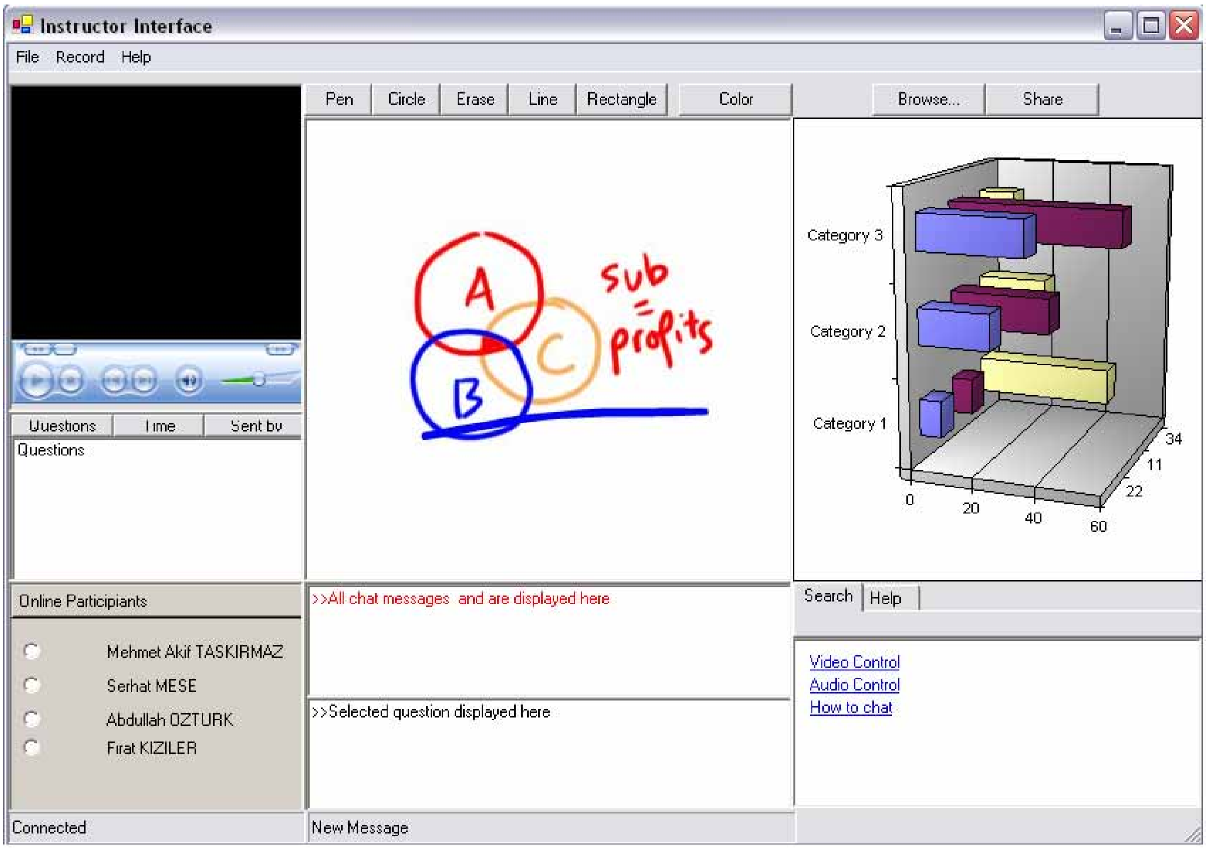
Replay lecture:

At the offline courses list, the student can see all the courses he is registered including the online one at this moment. From this list the student chooses one course and then chooses the date of the past lecture listed in the combo box next to it. Then the student presses the ‘replay lecture’ to watch offline replay of the course.

Download files:

When the student selects one course from the offline course list, the course files are appeared below in the course files list. The student can select one or more files from this list and then presses the ‘download’ button to download them to his computer. ‘refresh’ button refreshes the list if pressed.

The Instructor Interface



Mehmet Akif TASKIRMAZ
Serhat MESE
Abdullah OZTURK
Firat KIZILER

>>All chat messages and are displayed here

>>Selected question displayed here

Player

Displays the audiovisual streams coming from the media server. The instructor sees his/her image in this part.

Multimediboard Window

The instructor uses this part to show and send the multimedia content to the students.

Whiteboard Window

The instructor uses this part as the chalkboard in a real classroom [3].

Chat Window

The instructor uses this part to see the questions asked by the students and all the public chat messages.

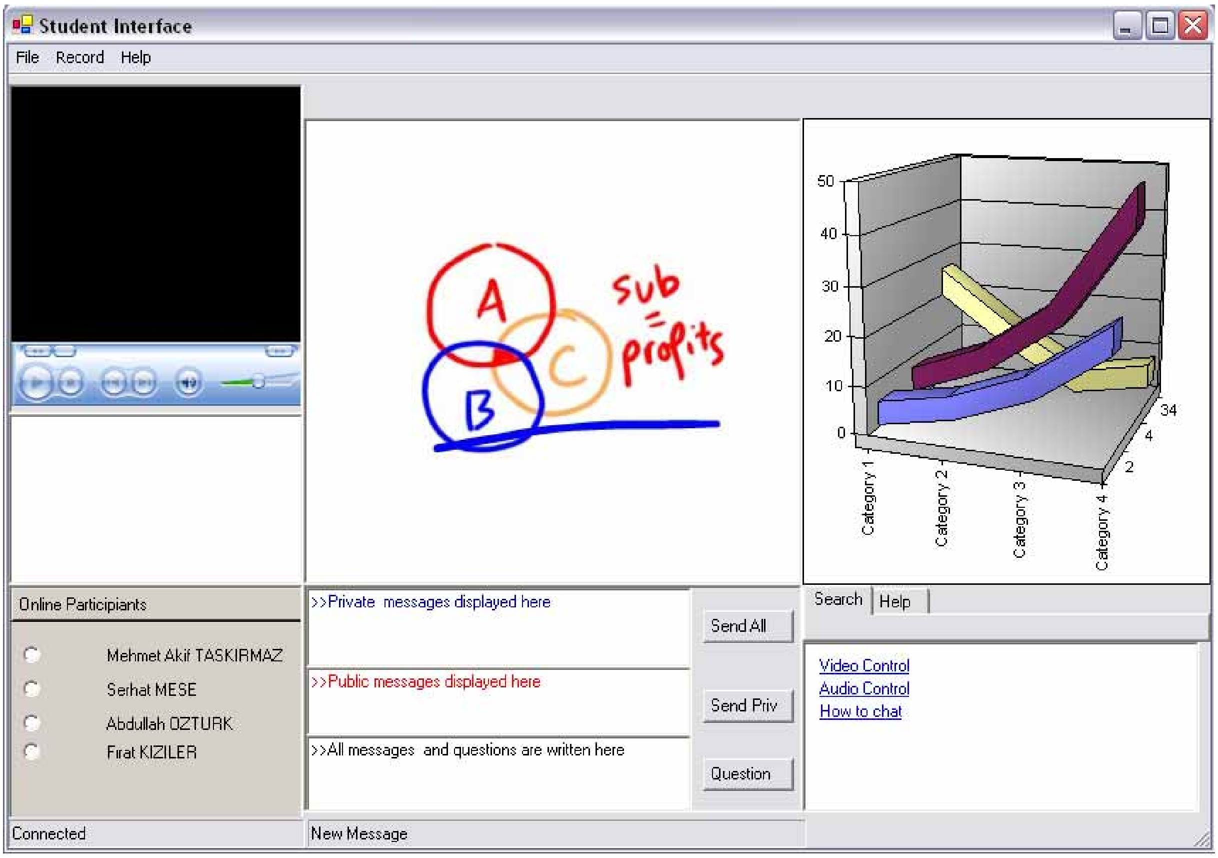
Online Users Window

The instructor uses this part to see the online users.

Help Window

The instructor uses this part for getting help about the usage of the application. It consists of two parts: dynamic help and search.

The Student Interface



Mehmet Akif TASKIRMAZ
Serhat MESE
Abdullah OZTURK
Firat KIZILER

>>Private messages displayed here

>>Public messages displayed here

>>All messages and questions are written here

Player

Displays the audiovisual streams coming from the media server.

Multimediboard Window

Displays the multimediaboard streams coming from the media server.

Whiteboard Window

Displays the whiteboard streams coming from the media server.

Chat Window

Used for chatting and asking questions.

Online Users Window

Used to see the online users.

Help Window

Used for getting help about the usage of the application. It consists of two parts: dynamic help and search.

Components Available

Microsoft Visual C#.Net is a programming language that allows the user to create complex applications for Windows, without all of the overhead required using other languages. It allows the user to pick from a list of practically thousands of controls, and draw them on the screen. These controls then have certain events, methods, and properties that can be set. When a particular event fires, the code associated with that event is executed. See the MSDN Library for Microsoft C#.Net for a complete list of all components available, including each component's properties, methods and events.

6. RESTRICTIONS, LIMITATIONS AND CONSTRAINTS

Performance/Behaviour Issues

The server side of the KLAS can only work on Microsoft Windows 2000/2003 server operating systems because Windows Media Server is distributed only with these operating systems.

The client side of the KLAS is designed to be compatible with Microsoft Windows 2000/XP/2003 operating systems (earlier versions will not be supported).

The reason for choosing DirectX 8.1 or higher is that the GDI+ Libraries (which is used for creating interfaces) requires at least DirectX 8.1.

Program Limitations

- The resolution of the player is 232 by 200.
- All the streams are in wmv format.
- The students do not have permissions to change the contents of the whiteboard, multimediaboard and the player.
- The user can not connect again immediately. Connections are refreshed every five minutes.
- KLAS works on LAN.

7. PROJECT SCHEDULE

Detailed Plan

Phase 1: Design and Analysis

- Database design
  - Design tables and entities of database
- Architectural design for client – server system
  - Designing client interface
  - Designing server interface
- Preparation of the design report

Phase 2: Implementation

- Configuration and testing of the required software
- Creating database
  - Population of the database with sample data
- Development of network connection among the applications
- Implementation
  - Implementation of the chat service
  - Implementation of the whiteboard
  - Implementation of the multimedia board
  - Implementation of video and audio streaming
  - Implementation of user interface
  - Implementation of help service
  - Implementation of newsgroup on the web and WAP
- Preparation of implementation document

Phase 3: Testing

- Integration of all modules and testing of integrated system
- Testing of all features
- Preparation test document

Gannt Chart

PROJECT PLAN

			Months															
Tasks	**Start Date**	**End Date**	**Feb**				**Mar**				**Apr**				**May**			
Implementation	14/02/05	22/05/05			🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩				
Configuration and testing of the required software	14/02/05	18/02/05			🟨													
Creating Database	19/02/05	21/02/05			🟨													
Development of network connection among the applications	22/02/05	23/02/05				🟨												
Implementation of the chat service	24/02/05	28/02/05				🟨												
Specifying the coding standards	1/03/05	7/03/05					🟨											
Implementation of the whiteboard	8/03/05	14/03/05						🟨										
Implementation of the multimedia board	15/03/05	25/03/05							🟨									
Implementation of video and audio streaming	26/03/05	8/04/05								🟨	🟨							
Implementation of user interface	9/04/05	13/04/05										🟨						
Implementation of help service and newsgroup	14/04/05	21/04/05											🟨					
Preparation of implementation document	22/04/05	24/04/05											🟧					
Integration of all modules and deployment "premature release"	25/04/05	2/05/05												🟨				
Testing	3/05/05	15/05/05													🟩	🟩		
Testing of all features	3/05/05	8/05/05													■			
Preparation of test document	9/05/04	15/05/05														🟧		

Research	Documentation	Phase Timeline	Testing	Implementation
🟥	🟧	🟩	■	🟨

8. TESTING ISSUES

Classes of Tests

Unit Testing

Individual components (database, error, help, …) will be tested seperately. All components can be tested through the object handler. All unit testing will be done in Whit Box fashion.

Integration Testing

Combined components will be tested as a whole. To maintain maximum control over the testing criteria, all data files will be made specifically for testing purposes.

High-Order Testing

The High-Order testing will be performed on the complete, integrated system.

Expected Software Response

General Hardware Requirements

- Processor: PIII 733 MHz
- System Memory: 256 MB
- Video Memory: 32 MB

Identification Of Critical Components

AudioVisual Streaming

The audiovisual data should be transmitted continously without errors. Windows Media Server performs this operation.

Synchronization of Components

The synchronization between the three main components (player, whiteboard and multimediaboard) should be provided. This is the most critical part in this project. The broadcast object performs this task.

Unpredictable network errors while transmissions of audio and video data packets lead to latency between audio and video while processing data packets on students side. For real-time streaming data, using strategy which will ignore loss of some data packets, especially for audio and video, will not affect the performance of audio-visual processing on the student side. This obstacle will be overcome by using Windows Media Server.

On the other hand, transmission of multimedia content requires an approach will provide almost-lossless transmissions of data packets over network. Namely, retransmission of data packets which were not received by the student-side is necessary. This does not consumes bandwidth as much as retransmission of large video frames over network.

9. GLOSSARY

American Standard Code for Information Interexchange (ASCII)

A computer language used to convert letters, numbers, and control codes into a digital code understood by most computers.

Application Service Provider (ASP)

A specialized form of an Internet service provider (ISP) that allows a company to have a software application hosted via a rental fee. An ASP sells access to a “packaged application” on a fee basis. ASPs provide IT operations expertise (offering the necessary application functionality, hardware, database and networking services, etc.) and frequently also business operation expertise in a particular market niche or in a particular functional area (such as HR or logistics management).

Asynchronous

Communication in which interaction between parties does not take place simultaneously.

Asynchronous Transfer Mode (ATM)

A high bandwidth,low delay, packet-like switching and multi-plexing technique.

Audio Bridge

A device used in audioconferencing that connects multiple telephone lines.

Audioconferencing

Voice only connection of more than two sites using standard telephone lines.

Backbone

A primary communication path connecting multiple users.

Band

A range of frequencies.

Bandwidth

Information carrying capacity of a communication channel.

Browser

Software that allows you to find and see information on the Internet.

Codec (COder/DECoder)

Device used to convert analog signals to digital signals for transmission and reconvert signals upon reception at the remote site while allowing for the signal to be compressed for less expensive transmission.

Collaborative Tools

Allow learners to work with others via e-mail, threaded discussions or chat. In some cases, collaboration is used to facilitate team-based projects. Collaborative tools can sometimes provide the ability to host moderated discussion groups, where students and instructors can collaborate on course-related materials or assignments in an asynchronous environment. In addition, real-time synchronous chat allows learners to communicate with their peers and instructors, emulating a physical classroom setting.

Compressed Video Television signals transmitted with much less than the usual bit rate. The lower bit rates typically involve some compromise in picture quality, particularly when there is rapid motion on the screen.

Computer-Based Training (CBT) Course or educational material presented on a computer, primarily via CDROM or floppy disk. Unlike Web-based training, computer-based training does not require a computer connected to a network and does typically not provide links to learning resources outside of the course.

Desktop Videoconferencing Videoconferencing on a personal computer.

Dial-Up Teleconference Using public telephone lines for communication links among various locations.

Distance Education The process of providing instruction when students and instructors are separated by physical distance, and technology, often in tandem with face-toface communication, is used to bridge the gap.

Distance Learning The desired outcome of distance education.

Echo Cancellation The process of eliminating the acoustic echo in a videoconferencing room.

E-Learning/Technology-Based Learning Covers a wide set of applications and processes such as Web-based learning, computer-based learning, virtual classrooms, and digital collaboration. It includes the delivery of content via Internet, intranet/extranet (LAN/WAN), audio/video tape, satellite broadcast, interactive TV, and CD-ROM.

Fiber Optic Cable Glass fiber that is used for laser transmission of video, audio, and/or data.

File Transfer Protocol A protocol that allows you to move files from a

(FTP) distant computer to a local computer using a network like the Internet.

Full Motion Video Signal which allows transmission of complete action taking place at the origination site.

Fully Interactive Video (Two way interactive video) Two sites interact with audio and video as if they were colocated.

Host A network computer that can receive information from other computers.

Instructional Television Fixed Service (ITFS) Microwave-based, high-frequency television used in educational program delivery.

Integrated Services Digital Network (ISDN) A telecommunications standard allowing communications channels to carry voice, video, and data simultaneously.

Interactive Media Frequency assignment that allows for a two-way interaction or exchange of information.

Internet-Based Training/Web-Based Training (WBT)/Online Training Delivery of educational content via a web browser over the public Internet, a private intranet, or an extranet (LAN/WAN). Internet-based training provides links to learning resources outside of the course, such as references, e-mail, bulletin boards, and discussion groups. It provides the advantages of computer-based training (CBT) while retaining advantages of instructor-led training. The term Internet-based training is used synonymously with Web-based training and online training.

Internet Protocol (IP): The international standard for addressing and sending data via the Internet.

Local Area Network (LAN) Two or more local computers that are physically connected.

Learning Management System (LMS): Internet-based software that deploys, manages, tracks and reports on interaction between a) the learner and the content, and b) the learner and the instructor. In particular, training management systems perform student registration, track learner progress, record test scores, and indicate course completions, and finally allow instructors/trainers to assess the performance of their students. Learning management systems administer and track both online and classroom-based learning events, as well as other training processes.

Learning Portal

A Web site that offers learners or organizations consolidated access to learning and training resources from multiple sources. Learning portals can be grouped into content consolidation portals, embedded technology portals, internal portals, community & collaboration portals, and affiliation portals. Operators of learning portals are also called content aggregators, distributors or hosts.

Learning Service Provider (LSP)

An LSP is a specialized type of ASP offering learning management and training delivery software on a hosted/rental basis via diverse business models. There are four different types of LSPs: 1) full service LSPs (customizing, implementing, and hosting a complete software solution via a private network); 2)content specific LSPs (licensing content to an organization and providing a level of learning management services to the buyer); 3) tool specific LSPs (licensing and hosting their specific system to an organization); and 4) portal LSPs (hosting a portal site and bundle the learning system in the background). LSPs also include value-added resellers (VAR) and companies providing certification and testing services, online collaboration services, media production and delivery services, and online tutoring.

Multimedia

Any document which uses multiple forms of communication, such as text, audio, and/or video.

Multi-Point Control Unit (MCU)

Computerized switching system which allows point-to-multipoint videoconferencing.

Network

A series of points connected by communication channels in different locations.

Protocol

A formal set of standards, rules, or formats for exchanging data that assures uniformity between computers and applications.

Server

A computer with a special service function on a network, generally receiving and connecting incoming information traffic.

Slow Scan Converter

Transmitter/receiver of still video over narrow band channels. In real time, camera subjects must remain still for highest resolution.

Synchronous Communication in which interaction between participants is simultaneous.

Teleconferencing Two way electronic communication between two or more groups in separate locations via audio, video, and/or computer systems.

Technology-Based Training Includes the delivery of content via Internet, intranet/extranet (LAN/WAN), satellite broadcast, audio/video tape, interactive TV, and CD-ROM. Technology-based training includes computer-based training (CBT) and Web-based training (WBT).

Training Management Systems See Learning Management Systems.

Transmission Control Protocol (TCP) A protocol which makes sure that packets of data are shipped and received in the intended order.

Video Teleconferencing A teleconference including two way video.

Web-Based Training (WBT) See Internet-Based Training

Note
The following resources were reviewed and consulted in the preparation of this publication:

- Glossary: http://152.30.11.86/DEER/Houghton/Committees/distancelearn/GlossaryDistEd.html
- Glossary of Terms. http://www.ctcnet.com/tips/glossary.htm
- Newton, H. (1991). Newton's telecom dictionary.
- Telecom Library Inc: New York.
- Reed, J. (1996). Videoconferencing for learning glossary. http://www.kn.pacbell.com/wired/vidconf/glossary.html.
- The EdWeb Dictionary. http://k12.cnidr.org:90/dic.html
- Willis, B. (Ed.) (1994). Distance education: Strategies and tools. Educational Technology Publications, Inc.: Englewood Cliffs, N. J.

10. REFERENCES

1. Moore, M.G. & Thompson, M.M., with Quigley, A.B., Clark, G.C., & Goff, G.G. (1990). *The effects of distance learning: A summary of the literature. Research Monograph No. 2*. University Park, PA: The Pennsylvania State University, American Center for the Study of Distance Education. (ED 330 321)
2. Verduin, J.R. & Clark, T.A. (1991). *Distance education:The foundations of effective practice*. San Francisco, CA: Jossey-Bass Publishers.
3. http://www.tc.cornell.edu/~dwyer/