CENG 491

SENIOR PROJECT

Initial Design Report

PDC

(PROJECT DEVELOPMENT CENTER)

PROJECT TITLE:	VIRTUAL CLASSROOM	
COMPANY STAFF:	Harun Alpak	1249903
	Ömer Faruk Aygün	1297506
	Ufuk Biçen	1297555
	Ali Osman Tekin	1250802

1. INTRODUCTION

This document is written for establishing a starting step for the designing phase of Virtual Classroom project. This document covers the initial design approaches to the system and establishes a basis for the detailed design processes. We tried to handle all points of our system. Since this is an initial design report we left some points more general on the other hand we deeply focused on many aspects. We did not deeply focus on the following points:

- Converting presentation slides to the any image file format and use during classes
- Video and audio streaming

As a difference from analysis we changed the working style of our system. We have designed our first system as working in web browser, at this stage we faced many problems and changed the system as working with installed software.

In our Initial Design Report we firstly focused on data design of our system. Data design part covers revised versions of the previous ER schema of the system and the data dictionary. Furthermore it covers database design, data flow diagrams with control flow. Secondly we investigated the structural model of our system with class diagrams and their descriptions. And thirdly we focused on behavioral model with Use Case, State Transition Diagram and Sequence diagrams.

2. DATA MODEL

2.1 DATABASE DESIGN AND ER DIAGRAM

User table stores the profile of all the users (admin, instructor, student). The username and password of a user is unique.

```
CREATE TABLE USER (
userid number,
username varchar (15),
password varchar (15),
name varchar (15),
surname varchar (15),
address varchar (100),
phonenumber number,
UNIQUE (username, password),
PRIMARY KEY (userid));
```

Admin table stores the profile of administrators that are also users as shown in the ER diagram. The column adid uniquely identifies an admin.

```
CREATE TABLE ADMIN (
adid number,
username varchar(15),
password varchar(15),
PRIMARY KEY (adid),
FOREIGN KEY (username, password) REFERENCES USER (username, password)
                                        ON DELETE CASCADE);
```

Instructor table stores the profile of instructors that are also users as shown in the ER diagram. The column insid uniquely identifies an admin.

```
CREATE TABLE INSTRUCTOR (
insid number,
username varchar(15),
password varchar(15),
PRIMARY KEY (insid),
FOREIGN KEY (username, password) REFERENCES USER (username, password)
                                        ON DELETE CASCADE);
```

Student table stores the profile of students that are also users as shown in the ER diagram. The column studentid uniquely identifies an admin.

```
CREATE TABLE STUDENT (
studentid number,
username varchar(15),
password varchar(15),
PRIMARY KEY (studentid),
FOREIGN KEY (username, password) REFERENCES USER (username, password)
                                        ON DELETE CASCADE);
```

News table stores the messages posted by users. Newsid uniquely identifies a posted message. The title of the message is stored as a string named title, the name of the person who posted the message is stored in the column sender and the course name that the message is written about is stored in the column of cname. The message is stored as a plain text in the column of content.

```
CREATE TABLE NEWS (
newsid number,
title varchar(15),
sender varchar(15),
cname varchar(15),
content text,
PRIMARY KEY(newsid));
```

Course table stores information about the courses. Courseid uniquely identifies a course. Coursename stores the name of the course and capacity is the number of students taking the course.

```
CREATE TABLE COURSE (
courseid number,
coursename varchar(50),
capacity number,
PRIMARY KEY (courseid));
```

Documents table stores necessary information for a document. Address uniquely identifies a document. The type stores the type information of the document (presentation, text file, video, audio, etc.) The column name stores the name of the document.

```
CREATE TABLE DOCUMENTS (
folderaddress varchar(100),
type varchar(15),
name varchar(15),
PRIMARY KEY (folderaddress));
```

Teaches table stores the information of which instructor teaches which course. Insid is the instructor id and cid is the course id.

```
CREATE TABLE TEACHES (
insid number,
cid number,
PRIMARY KEY (insid, cid),
FOREIGN KEY (insid) REFERENCES INSTRUCTOR (insid) ON DELETE CASCADE,
FOREIGN KEY (cid) REFERENCES COURSE (courseid)  ON DELETE CASCADE);
```

Downloads Document table stores information about which student downloads which document. Folderaddress is the address of the document to be downloaded. Sid is the id of the student who downloads the document and coursename is the name of the course that the document is about. Together with the sid, folderaddress uniquely identifies a document to be downloaded.

```
CREATE TABLE DOWNLOADSDOCUMENT (
folderaddress varchar(100),
sid number,
coursename varchar(15),
PRIMARY KEY (folderaddress, sid),
FOREIGN KEY (folderaddress) REFERENCES DOCUMENTS (folderaddress)
                                        ON DELETE CASCADE,
FOREIGN KEY (sid) REFERENCES STUDENT (studentid) ON DELETE CASCADE);
```

Has table stores information about which course has which documents to be downloaded. Course with courseid (cid) has the documents stored at the folder address (fadd).

```
CREATE TABLE HAS (
cid number,
fadd varchar (100),
PRIMARY KEY (cid, fadd),
FOREIGN KEY (cid) REFERENCES COURSE (courseid),
FOREIGN KEY (fadd) REFERENCES DOCUMENTS (folderaddress));
```

Attends table stores information about which student attends to which course and is present at what dates. Sid is the student's id number, cid is the courseid that the student is attending to and coursedate stores the dates when the course is given. The column present stores whether the student is present or absent the dates the course is given.

```
CREATE TABLE ATTENDS (
sid number,
cid number,
coursedate date,
present int,
PRIMARY KEY (sid, cid, coursedate),
FOREIGN KEY (sid) REFERENCES STUDENT (studentid) ON DELETE CASCADE,
FOREIGN KEY (cid) REFERENCES COURSE (courseid) ON DELETE CASCADE);
```

The table AddsDocument stores the information which documents are added by which instructors. Folderaddress is the address that the document is stored. Insid is the instructor id of the instructor who adds the document. Together with the insid, folderaddress uniquely identifies the document to be added. The column coursename stores the name of the course that the added document is about.

```
CREATE TABLE ADDSDOCUMENT (
folderaddress varchar(100),
insid number,
coursename varchar(15),
PRIMARY KEY (folderaddress, insid),
FOREIGN KEY (folderaddress) REFERENCES DOCUMENTS (folderaddress)
                                        ON DELETE CASCADE,
FOREIGN KEY (insid) REFERENCES INSTRUCTOR (insid) ON DELETE CASCADE);
```

The AddsCourse table stores information of which student attends to which course. The student with student id (sid) adds the course with courseid (cid) in order to attend during the semester. Together with cid, sid uniquely identifies such courses.

```
CREATE TABLE ADDSCOURSE (
sid number,
cid number,
PRIMARY KEY (sid, cid),
FOREIGN KEY (sid) REFERENCES STUDENT (studentid) ON DELETE CASCADE,
FOREIGN KEY (cid) REFERENCES COURSE (courseid) ON DELETE CASCADE);
```

The AddNews table stores information about which user sends which message to the newsgroup. The user with userid (insid) sends the message with newsid (nid).

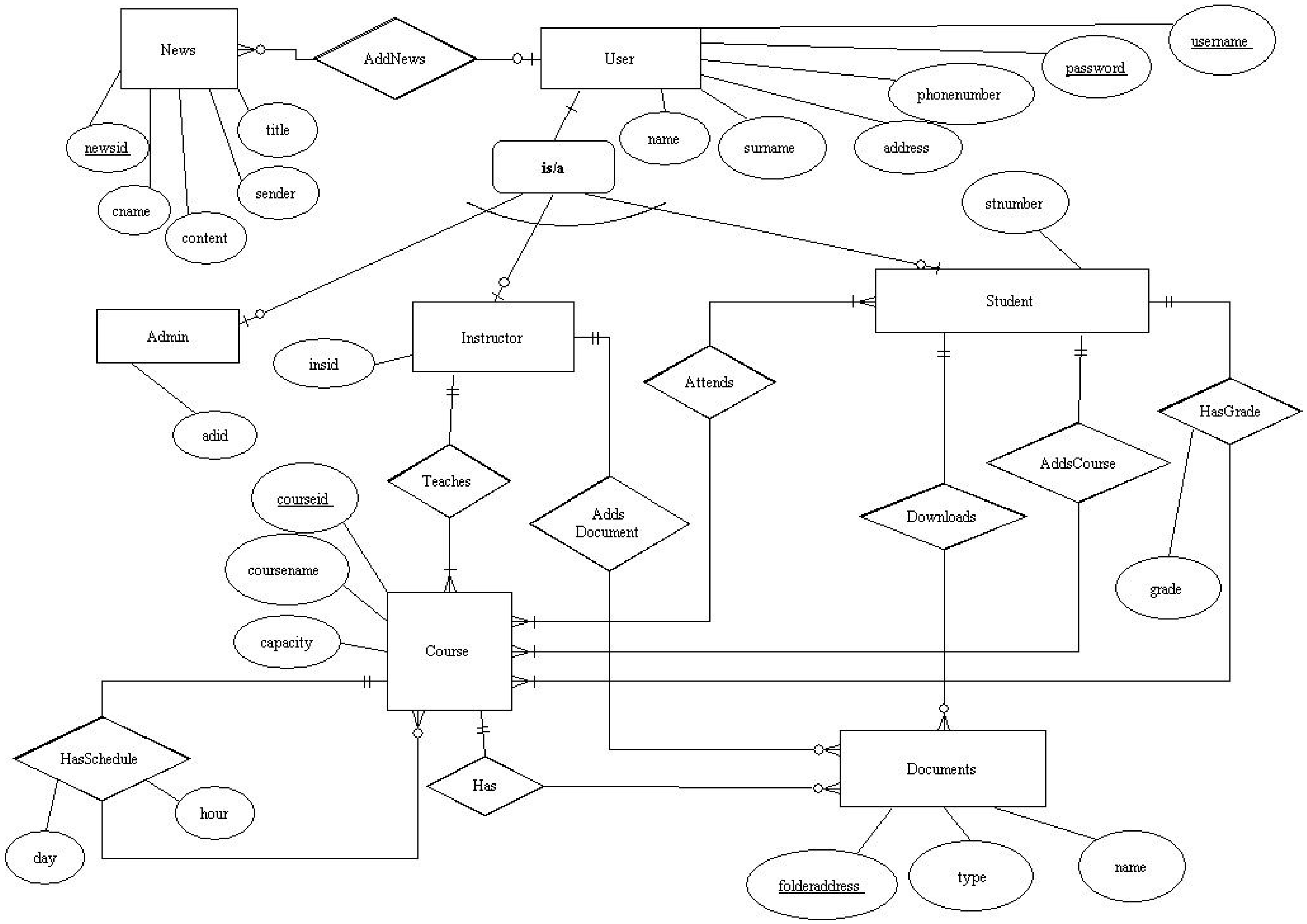
```
CREATE TABLE ADDNEWS (
insid number,
nid number,
PRIMARY KEY (insid, nid),
FOREIGN KEY (insid) REFERENCES USER (userid) ON DELETE CASCADE,
FOREIGN KEY (nid) REFERENCES NEWS (newsid) ON DELETE CASCADE);
```

The table Schedule stores the schedule of a course. Hour stores the hour of the course and day stores the day that the course is given.

```
CREATE TABLE HASSCHEDULE (
hour varchar (6),
day varchar (10),
cid number,
PRIMARY KEY (cid)
FOREIGN KEY (cid) REFERENCES COURSE (courseid) ON DELETE CASCADE);
```

HasGrade table stores the grade of the student with studentid (sid) from the course with courseid (cid). The columns sid and cid forms the key for this table.

```
CREATE TABLE HASGRADE (
grade varchar (3),
cid number,
sid number,
PRIMARY KEY (cid , sid),
FOREIGN KEY (cid) REFERENCES COURSE (courseid) ON DELETE CASCADE,
FOREIGN KEY (sid) REFERENCES STUDENT (studentid) ON DELETE CASCADE,
);
```


News
AddNews
User
username
password
phonenumber
newsid
title
sender
cname
content
name
surname
address
is/a
stnumber
Student
Admin
Instructor
insid
adid
Attends
HasGrade
AddsCourse
Teaches
Adds Document
Downloads
courseid
coursename
capacity
Course
grade
HasSchedule
Has
Documents
hour
day
folderaddress
type
name

2.2 REVIEW OF DFD & CFD

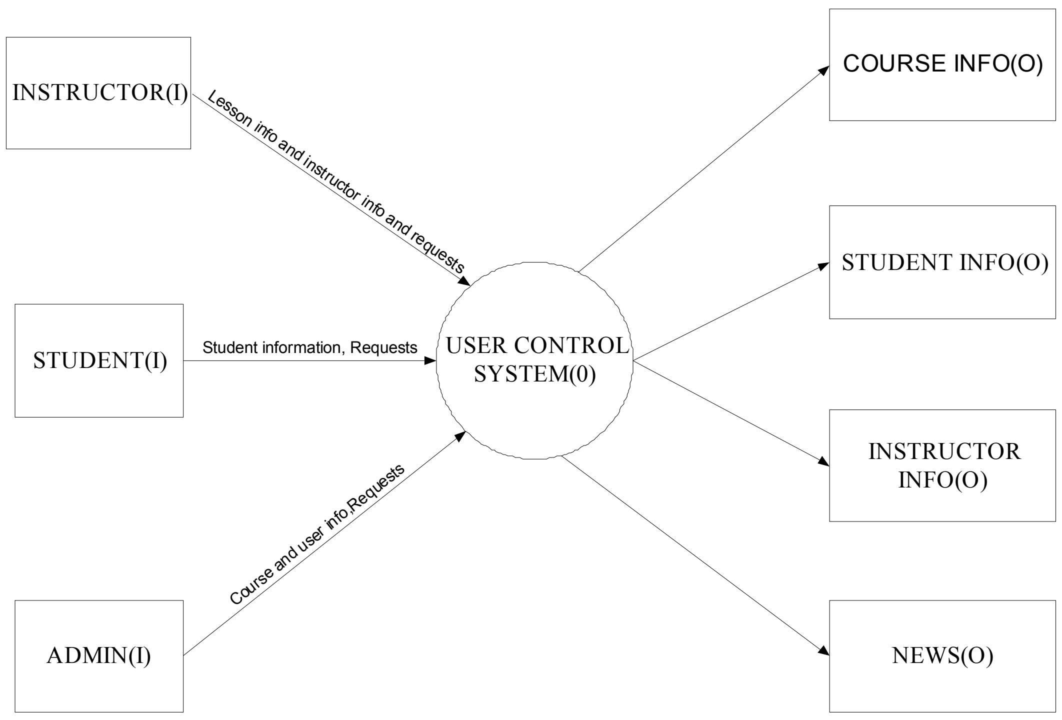
In our system we have two major components. The first component is Virtual Classroom and the other is User Control System. For this reason we examine our DFDs and CFDs into two part. This division also makes the system more clear. In the following section firstly we examine the User Control System and then we continue to examine Virtual Classroom System. Lets begin to examine the first component of the system that is User Control System.

2.2.1 User Control System DFDs & CFDs:

This component of our system only controls the user activities. In addition, this component also controls the database activities of our system.

2.2.1.1 User Control System DFD & CFD Level 0:

As you can see in the following graph this component of the system has tree inputs and four outputs. And all of the process made by the User Control System. Our main process (User Control System) receives some data or request from the inputs (Instructor, Student, Admin) and then manipulates the requested process and then gives the outputs (Course Info, Student Info, Instructor Info, News).

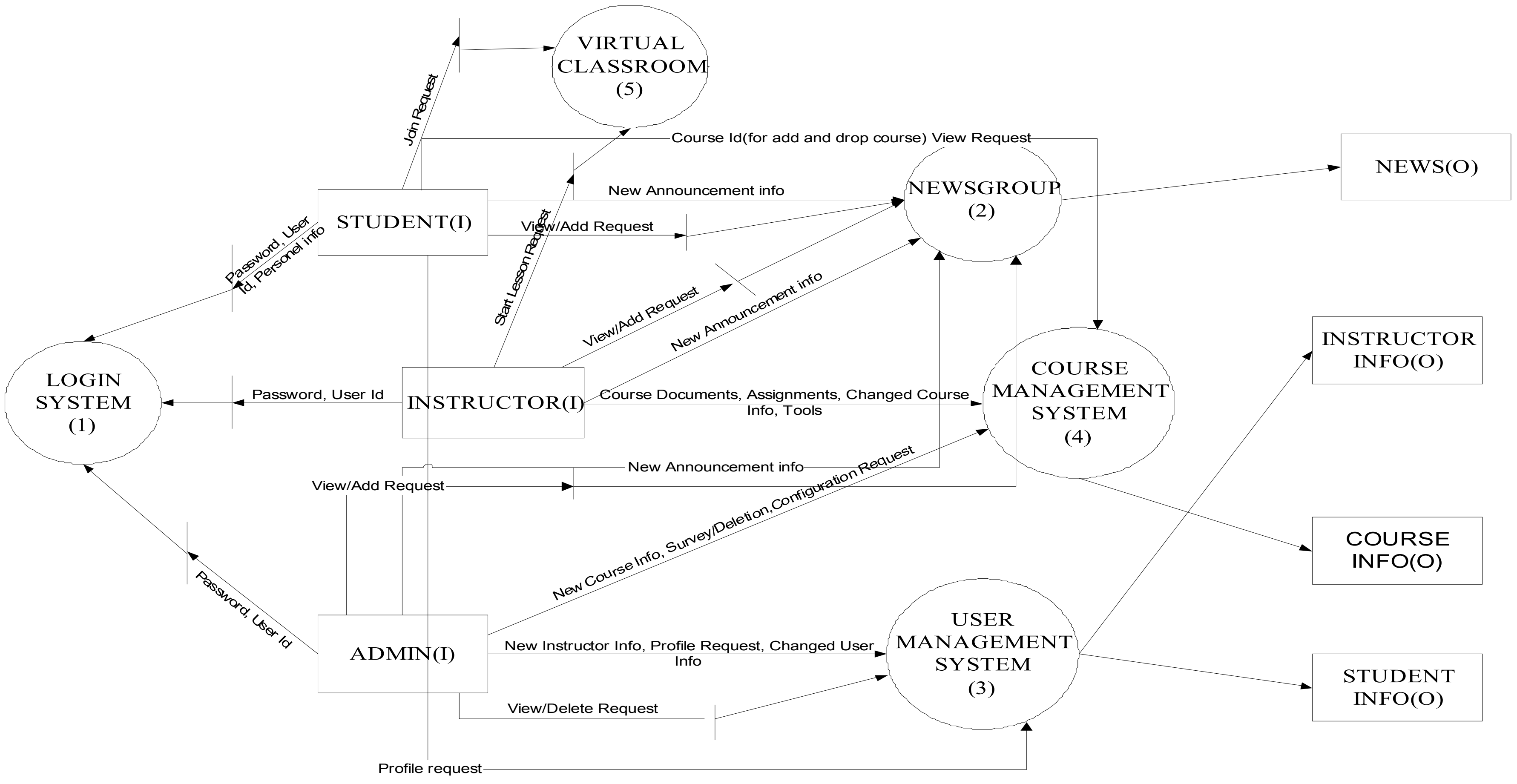


USER CONTROL SYSTEM DFD-0

I : INPUT
O: OUTPUT

2.2.1.2 User Control System DFD & CFD Level 1:

In the User Control System we have five subsystems that are Login, Newsgroup, User Management, Course Management and also one system that provides transition between User Control System and Virtual Classroom System. Login System checks the username and password of the users. Newsgroup system manages some process such as adding, posting of viewing news. User Management System controls user process and also shows the users information. Course Management system manages the process such as adding new course, adding assignment to course and etc. Virtual Classroom process is a transition between two components.

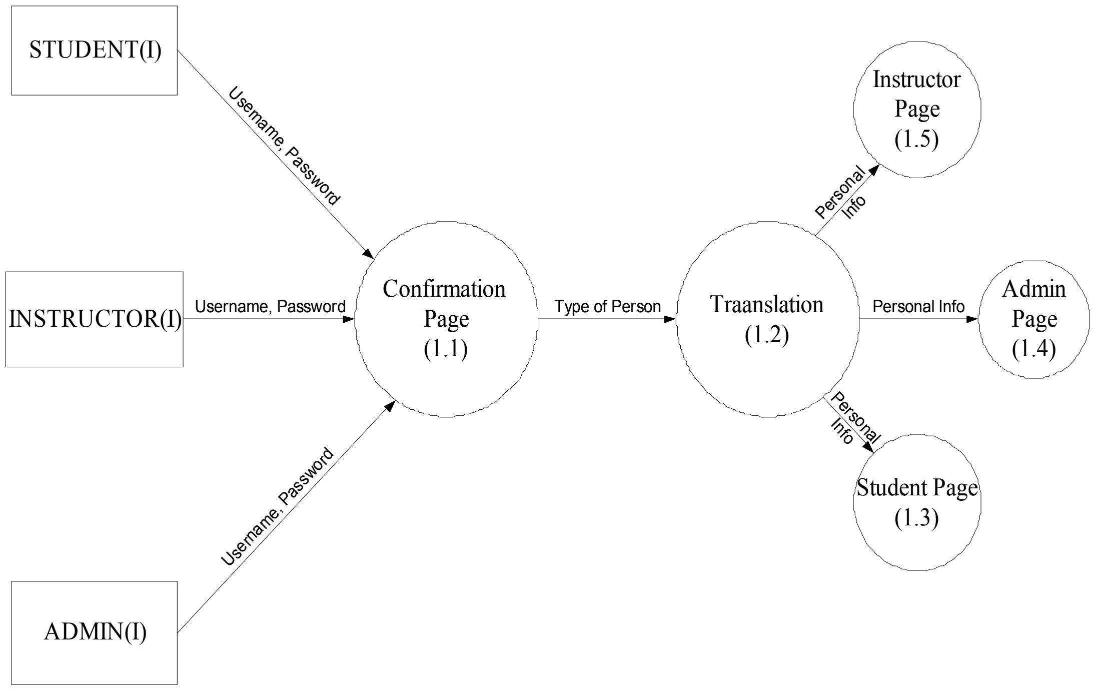


USER MANAGEMENT SYSTEM
DFD - 1

I : INPUT
O: OUTPUT

2.2.1.3 Login System DFD & CFD Level 2:

This system firstly provides the user entrance. After the entrance, this system identifies the user whether he/she is student, instructor, and admin. Then this system translates the user to appropriate page.

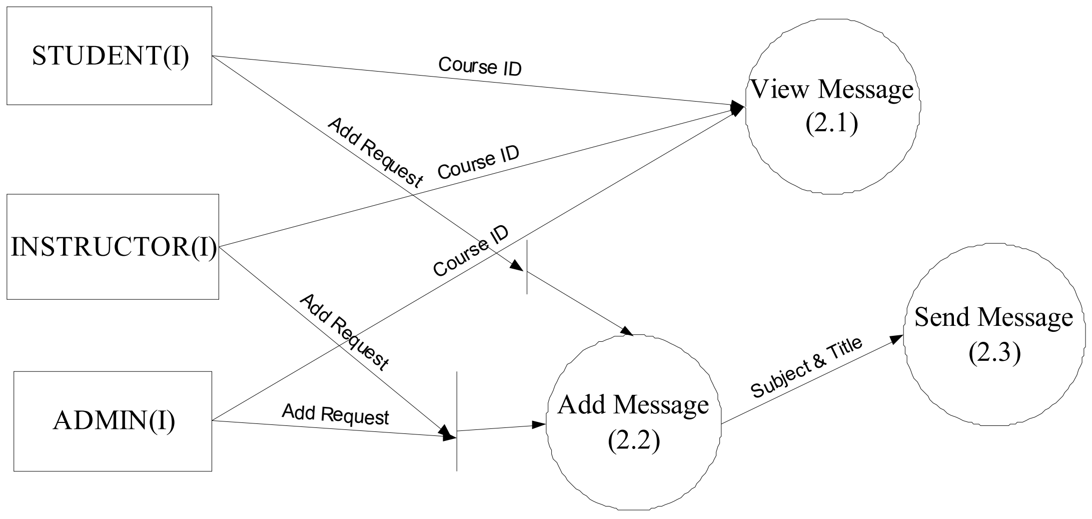


DFD - 2(LOGIN SYSTEM)

I : INPUT
O: OUTPUT

2.2.1.4 Newsgroup System DFD & CFD Level 2:

This system controls the news' processes. All of the users can view the news of courses, but they add only their registered course news.

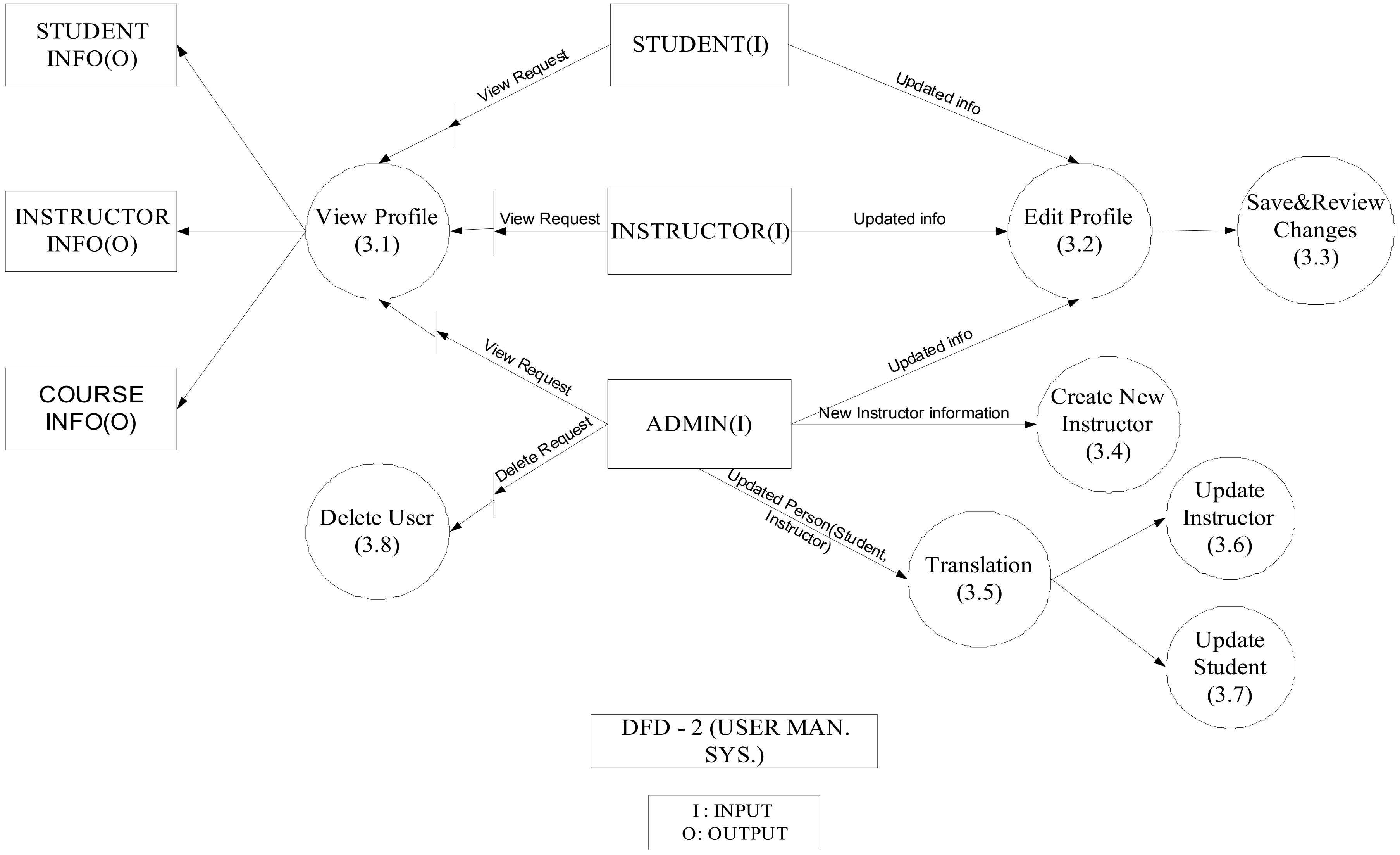


DFD - 2 (NewsGroup)

I : INPUT
O: OUTPUT

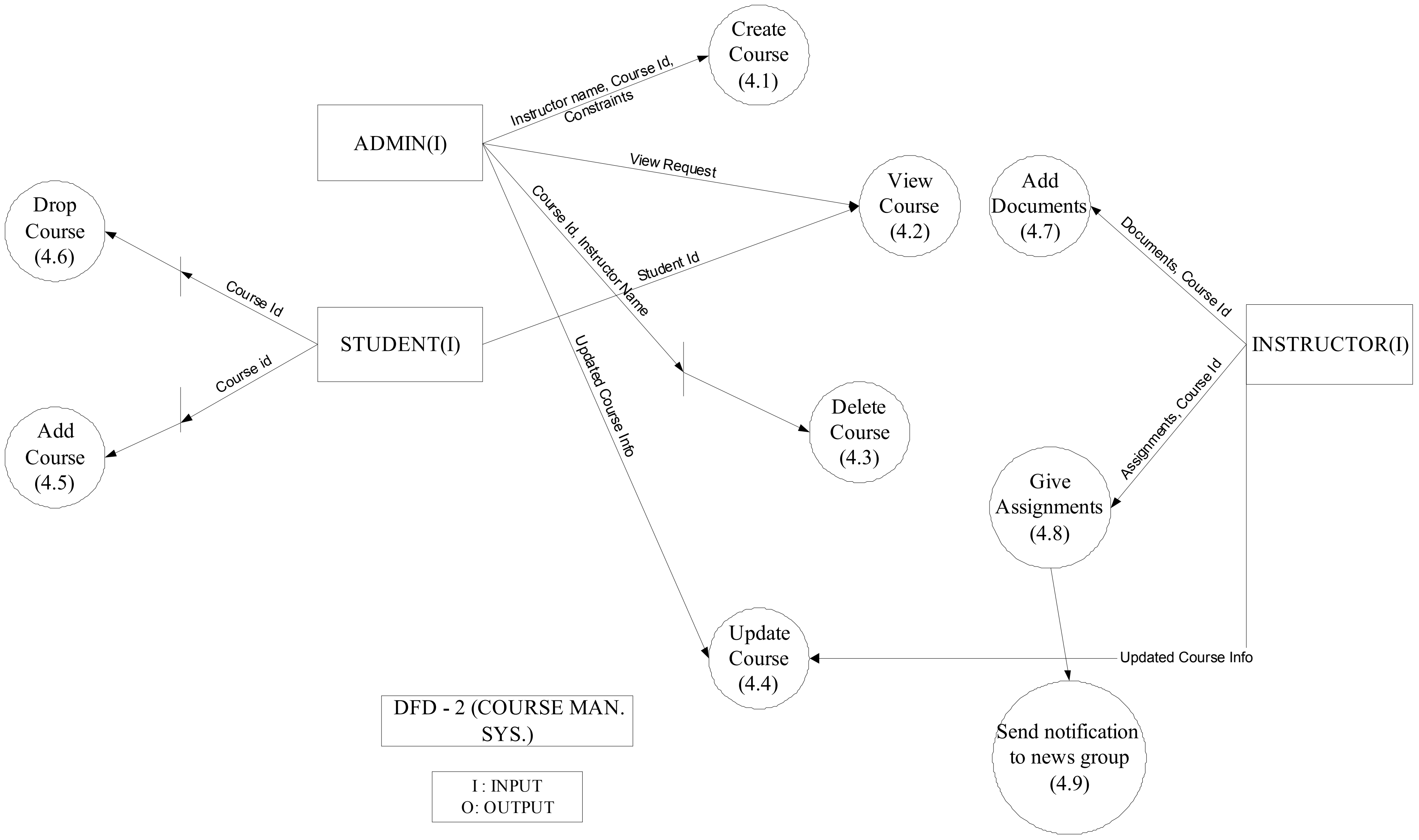
2.2.1.5 User Management System DFD & CFD Level 2:

In this system we have 5 main processes and also 3 sub processes. The basic processes of this system are view profile, delete user, edit profile, create new instructor, and updating user info. All of the users can view their profile from this system. And also all of them edit their own profile, but admin can also edit the student and instructor information. Moreover this system provides the creation of the new instructor. All of the instructor creation is made by admin. And the deletion process of the users from the system is made by admin. Admin is the most important user for this system, the continuity of the system depend on the admin.


STUDENT INFO(O)
STUDENT(I)
View Request
Updated info
INSTRUCTOR INFO(O)
View Profile (3.1)
View Request
INSTRUCTOR(I)
Updated info
Edit Profile (3.2)
Save&Review Changes (3.3)
View Request
Updated info
COURSE INFO(O)
ADMIN(I)
New Instructor information
Create New Instructor (3.4)
Delete Request
Delete User (3.8)
Updated Person(Student, Instructor)
Translation (3.5)
Update Instructor (3.6)
Update Student (3.7)
DFD - 2 (USER MAN. SYS.)
I : INPUT
O: OUTPUT

2.2.1.6 Course Management System DFD & CFD Level 2:

In this system we have 9 processes. If we first examine the relation between this system and the admin, we can see that admin can use 4 processes of the system that are create course, view course, delete course, and update course. All of the courses of the system are created or deleted by the admin. And besides this operation admin can also view or update the course information. The second actor of this system is instructor. Instructor can add assignment or documents and update the course information by using this system. And lastly, student uses this system for adding or dropping course.



DFD - 2 (COURSE MAN. SYS.)

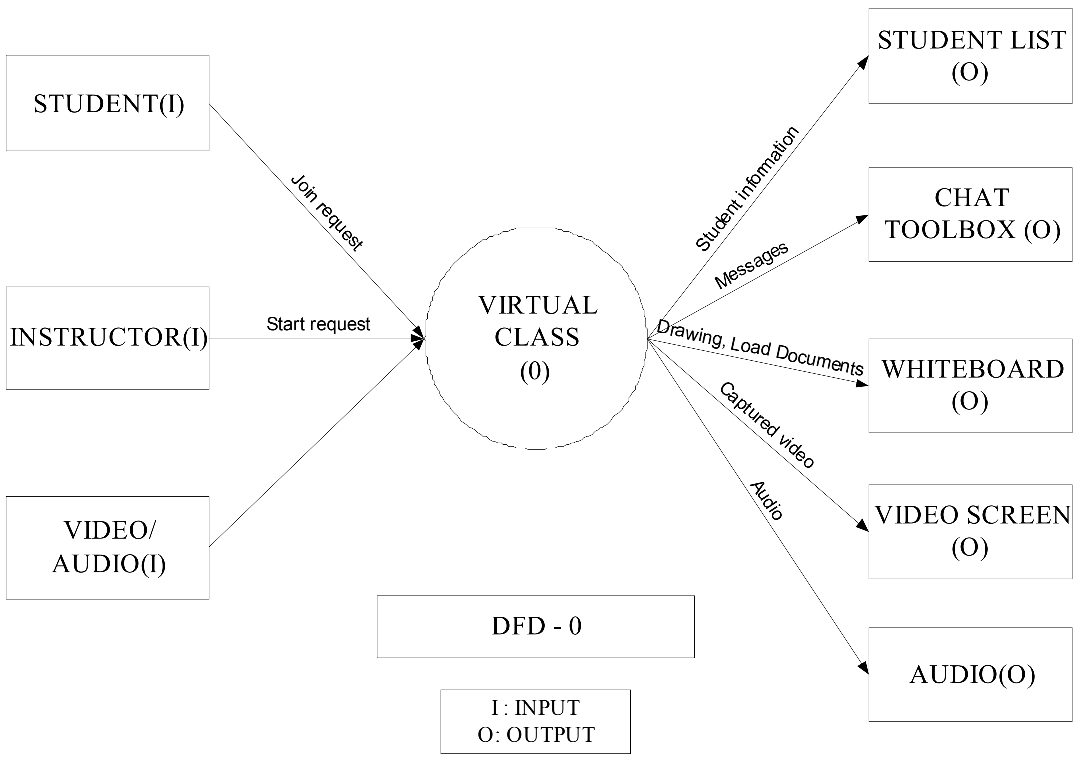
I : INPUT
O: OUTPUT

2.2.2 Virtual Classroom System DFDs & CFDs:

When the students and also instructor entered this system, all of the activities are controlled by this system. All of the tools and their interaction with the users are managed and all of the responses are given by this system.

2.2.2.1 Virtual Classroom System DFD & CFD Level 0:

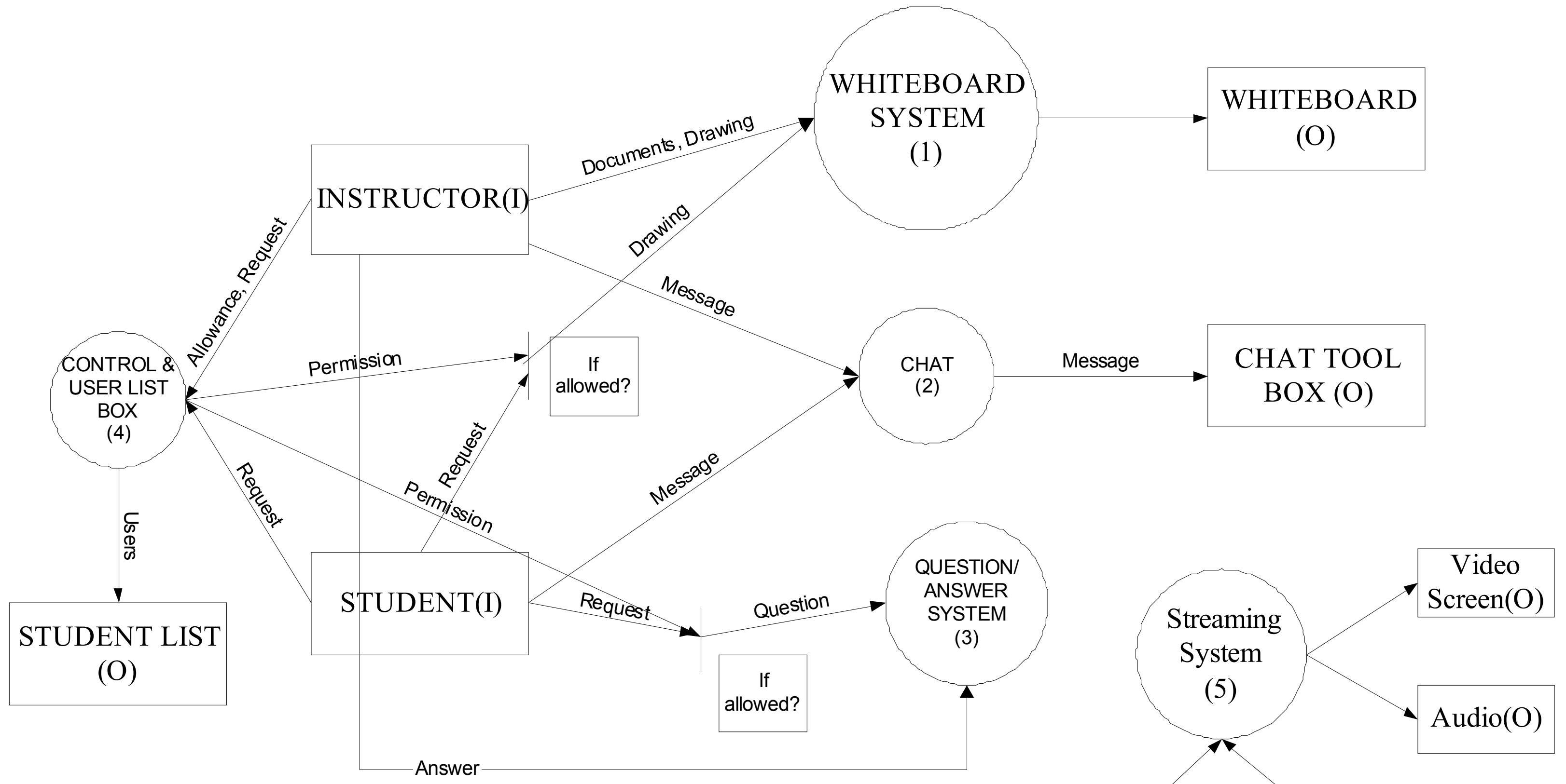
As you can see in the following graph representation of our virtual classroom component, we have 3 inputs and 5 outputs. Besides that, we have one main process of our system, which control all of the tools of the system. All process works under this main process of our system. Our inputs of this component are student, instructor and video/audio streaming device. And outputs are student list, chat toolbox, whiteboard, video screen, and audio. The important processes of this component are Whiteboard, Chat tool, and video/audio streaming system. All of them receive some data from users than all of them is distributed all of the members by this system.



2.2.2.2 Virtual Classroom System DFD &CFD Level 1:

In this system we have 5 main processes, which are whiteboard system, chat, Question/Answer system, Control & User list box, and Streaming system. In the whiteboard only instructor can use this tool but if instructor gives permission to the specific student then the student can also use the whiteboard for asking question. However all of the users can use the chat tool. Question/Answer system provides the communication between instructor and the students. When I student have a question then ask question by using this system. Control & User list box process show the user info and also students' permissions and these permissions are activated/deactivated by the instructor. Streaming system is one of the important processes of this component.

The captured video or audio, which is provided by web cam or any camera and microphone, is streamed to students by this system.

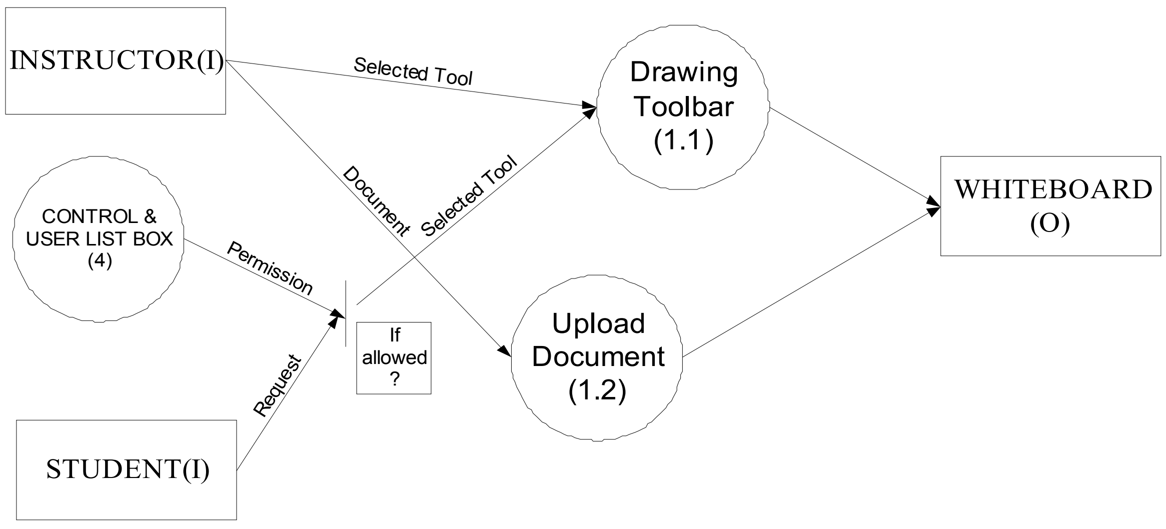


DFD - 1

I : INPUT
O: OUTPUT

2.2.2.3 Whiteboard DFD & CFD Level 2:

As you can see the following graphical representation of the whiteboard system, only the instructor can use the whiteboard. All of the drawings and writing is saved in specific place then it is sent all of the users as an output. If the instructor gives permission to the user then user can use white board system. In the corner of the whiteboard there is a drawing toolbar, which makes the drawing easier. Moreover, instructor can also upload their documents, such as power point file, image or text document, on whiteboard.

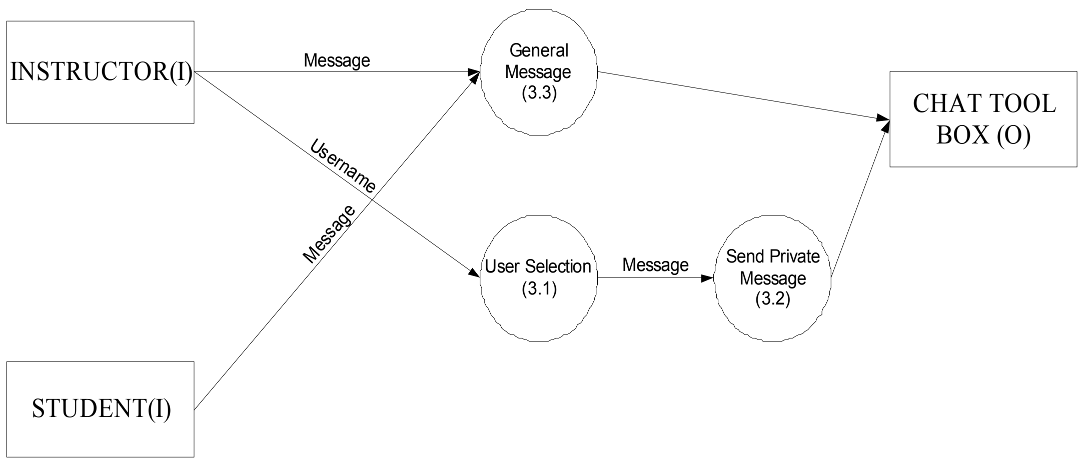


DFD - 2 (WHITEBOARD)

I : INPUT
O: OUTPUT

2.2.2.4 Chat Tool DFD & CFD Level 2:

We have two main processes of this chat tool. These are general message and private message. Only the instructor uses the private message. But general message can be used by all of the users.

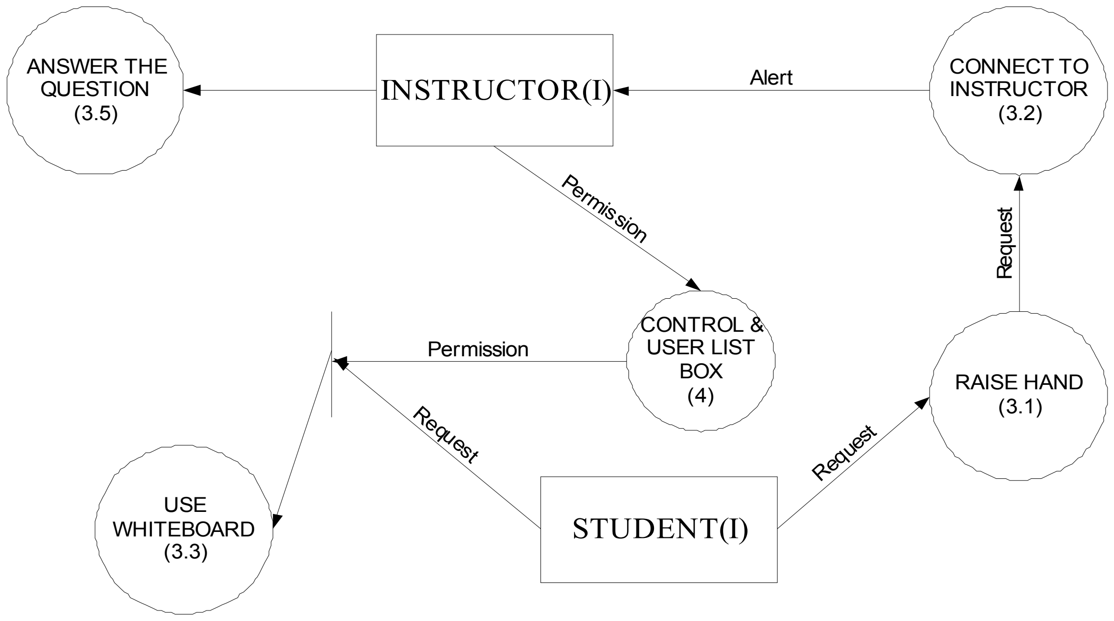


DFD - 2 (CHAT TOOL)

I : INPUT
O: OUTPUT

2.2.2.5 Question/Answer Box DFD & CFD Level 2:

This system provides the interaction between the students and the instructor. When a student wants to ask a question, the system alerts the instructor. Then when the instructor gives the permission to student then this student can use the white board for asking the question. Moreover, instructor can also get the control of whiteboard any time then answer the question.

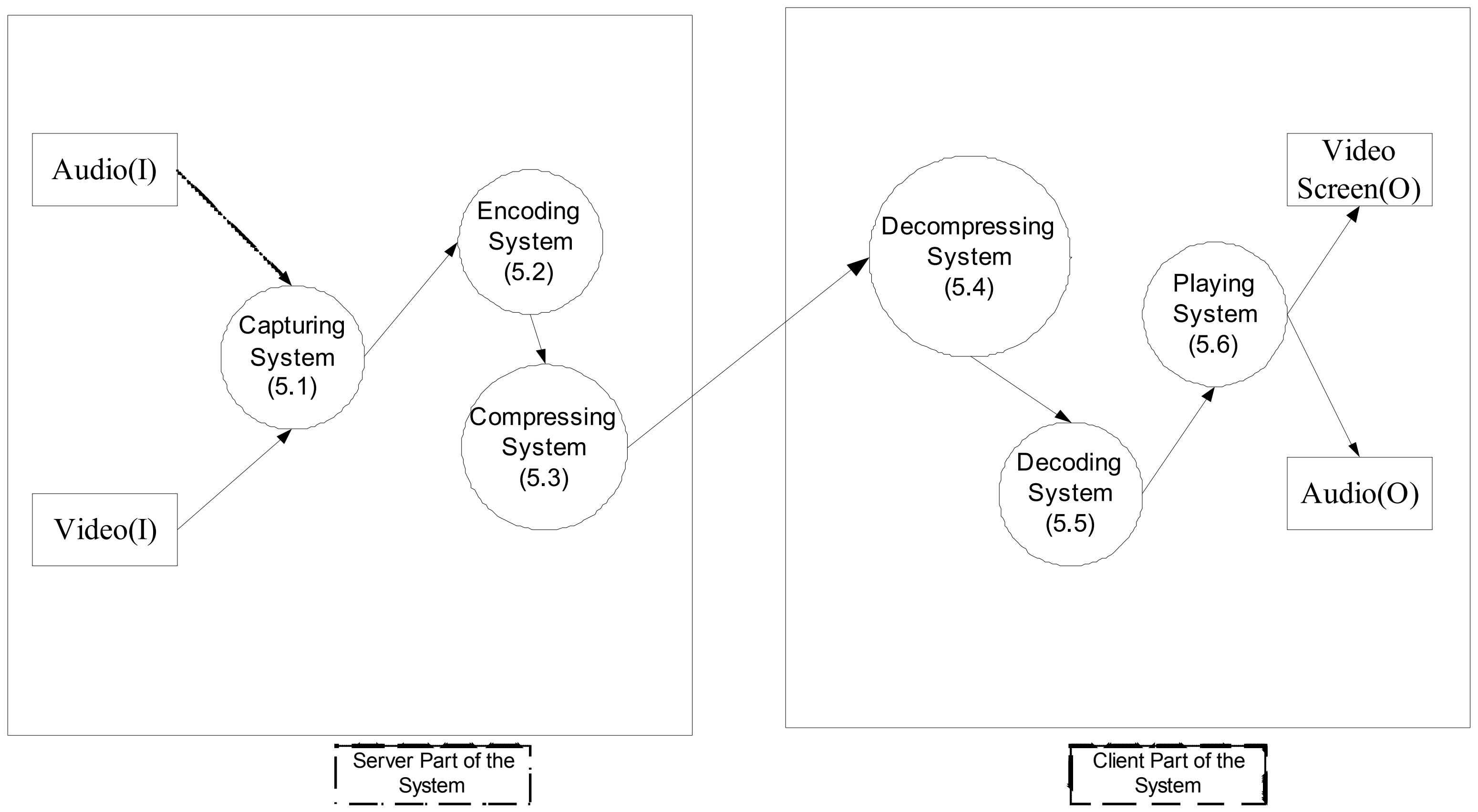


DFD - 2 (Q/A BOX)

I : INPUT
O: OUTPUT

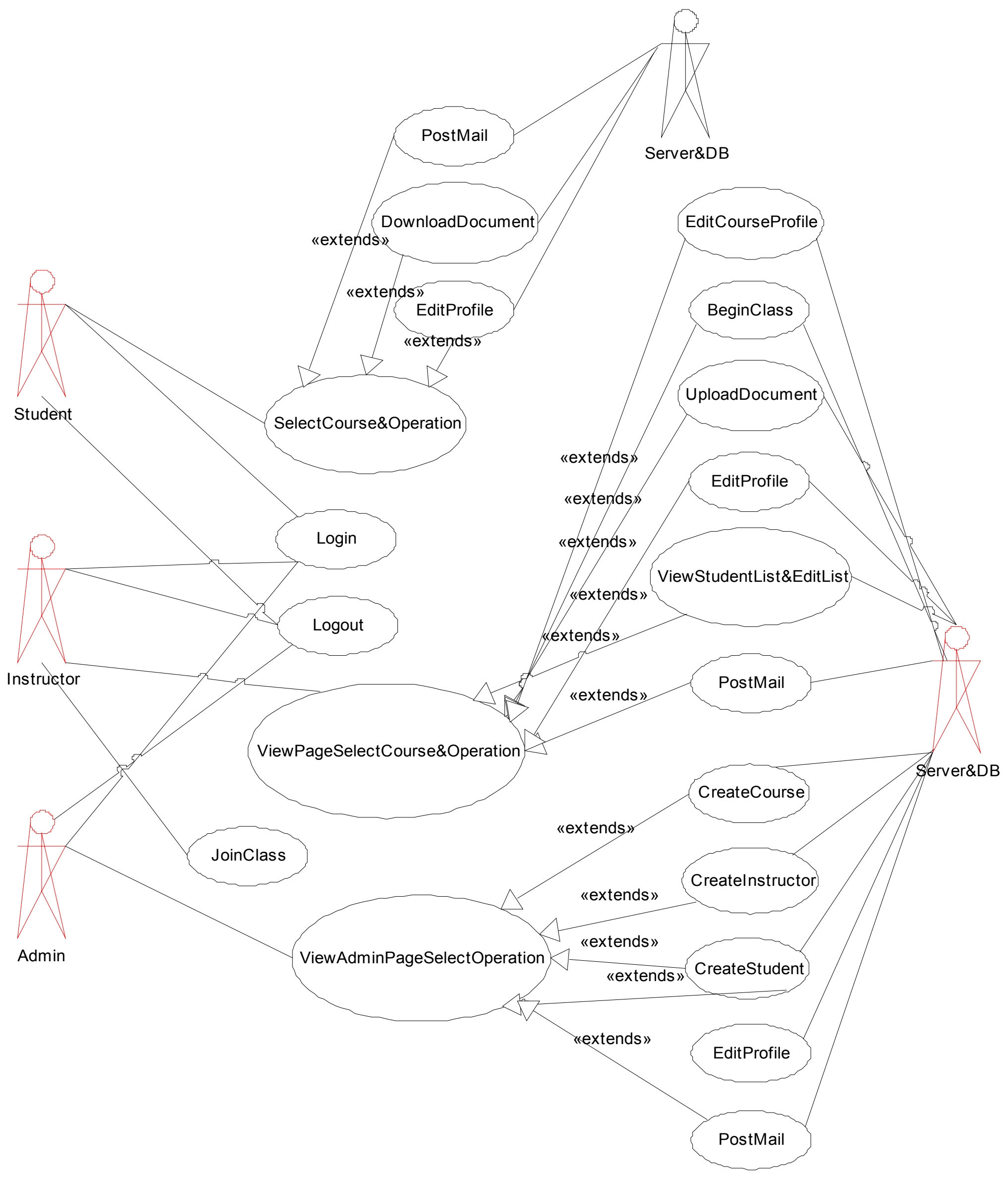
2.2.2.6 Streaming System DFD & CFD Level 2:

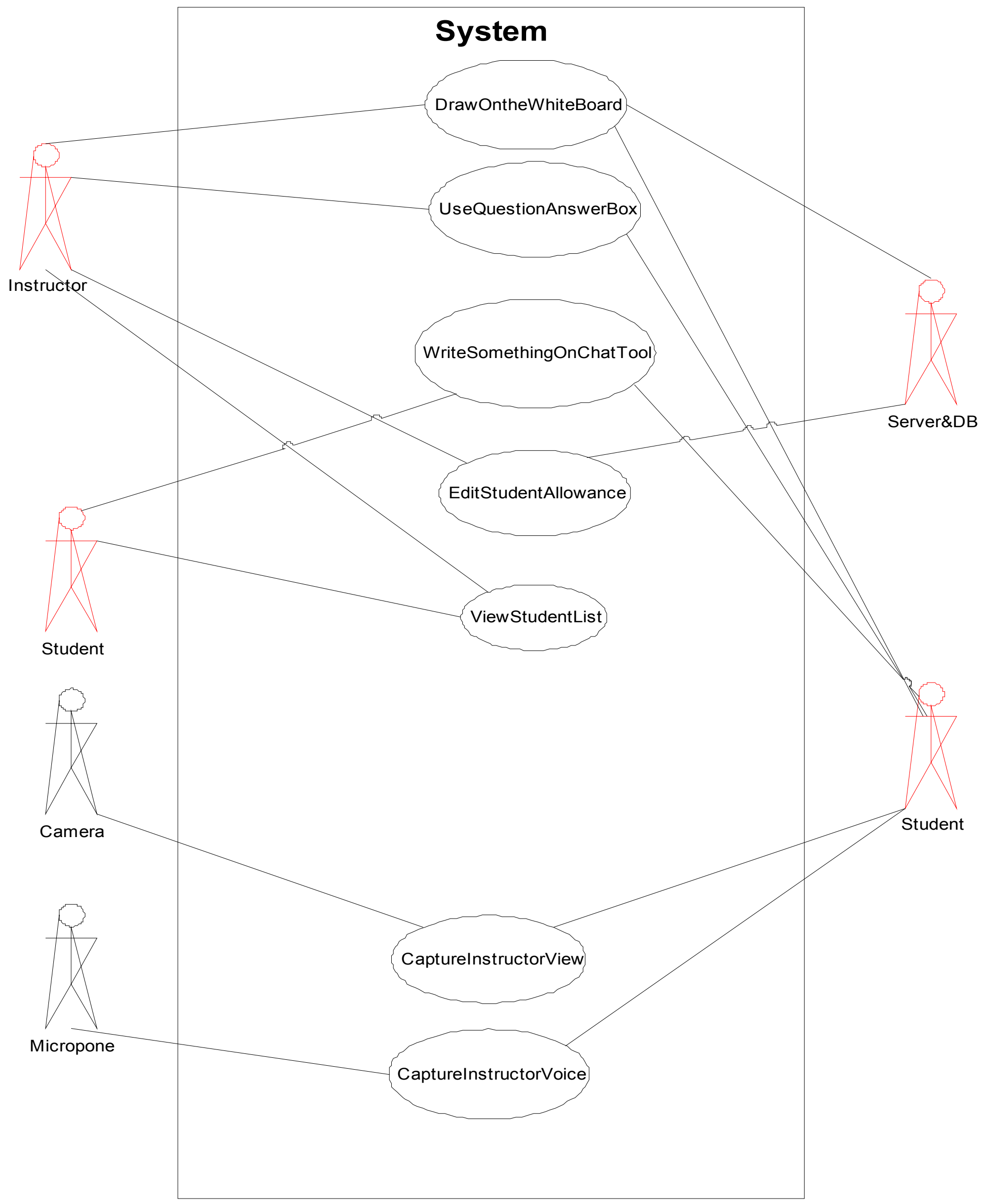
The main processes of this system are capturing, playing, streaming, compressing, encoding and decoding. This system firstly captures the video and audio from the web cam and microphone and then encodes and then compresses these encoded files. And all of the compressed files are streamed to the other users. When the other users get these files then the client part of the process firstly decompresses and then decodes these files by media player.


Audio(I)
Video(I)
Capturing System (5.1)
Encoding System (5.2)
Compressing System (5.3)
Decompressing System (5.4)
Decoding System (5.5)
Playing System (5.6)
Video Screen(O)
Audio(O)
Server Part of the System
Client Part of the System

3. ARCHITECTURAL DESIGN

3.1 USE CASE DIAGRAMS




System
DrawOntheWhiteBoard
UseQuestionAnswerBox
WriteSomethingOnChatTool
EditStudentAllowance
ViewStudentList
CaptureInstructorView
CaptureInstructorVoice
Instructor
Student
Camera
Micropone
Server&DB
Student

Use Case: System Login
Actor: Student, Instructor, Admin
Purpose: Verify UserName and Password to enter the system

Actor Actions	System Response
1. The user selects the user type and writes username and password	
	2. The database is searched in order to verify user info.
	3. If verification is OK than go to the related page else notify user

Use Case: Join Class
Actor: Student
Purpose: If the selected course is currently online and student is not banned from the course then student sends join request to the instructor

Actor Actions	System Response
1. Student clicks on the button	
	2. system notifies the instructor
	3. if allowed system takes student to the virtual classroom

Use Case: Student Course&Operation Choice
Actor: Student
Purpose: Navigate student to realize his/her desired action

Actor Actions	System Response
	1. The system displays the registered courses of the student
2. The student selects a course	
	3. System takes the student to the selected courses operation selection interface
	4. System displays available options to the student
5. Student clicks on the operation link	
	6. System takes student to the related operation’s page. Go to the selected use case

Use Case: Edit Profile
Actor: Student, Instructor, Admin
Purpose: System allows users to make some changes on their profiles

Actor Actions	System Response
	1. System displays user info
2. user makes the changes	
3. clicks the “save” button	

	4. system checks the changes and if verified saves else notify user

Use Case: View Course Info
Actor: Student
Purpose: Students can view their registered courses info

Actor Actions	**System Response**
1. Student selects the link	
	2. System displays the course info (name/schedule/instructor...)

Use Case: Post Mail
Actor: Student, Instructor, Admin
Purpose: Users may send messages to the course's and general newsgroup

Actor Actions	**System Response**
1. User selects the link	
	2. System brings an interface of sending mail
3. User writes mail and clicks "send" button	
	4. System sends the mail to the specified mailbox.

Use Case: Download Course Material
Actor: Student
Purpose: If there exist available documents in the database student may download

Actor Actions	**System Response**
1. Student selects the link	
	2. System checks the database if there exists available materials then lists them
3. Student downloads the desired material	

Use Case: Instructor Course&Operation Choice
Actor: Instructor
Purpose: Navigate instructor to make his/her desired action

Actor Actions	**System Response**
	1 . System shows the assigned courses of the instructor
2. Instructor selects any course from list	
	3. System takes instructor to the selected course's operation selection page
	4. System displays the available operations

5. Instructor selects any operation by clicking on the link	
	6. System takes the instructor to the selected operation’s page. Go to the selected use case

Use Case: Edit Course Profile
Actor: Instructor
Purpose: Instructor can make changes on the course like schedule change

Actor Actions	**System Response**
1. Instructor selects the link	
	2 . System displays the course info table
3. Instructor makes the changes and clicks on the “save button”	
	4. If the changes are valid than updates the DB else notify the instructor

Use Case: View and Edit Student List
Actor: Instructor
Purpose: Instructor can ban some students from class

Actor Actions	**System Response**
1. Instructor selects the link	
	2. System displays the list of students enrolled that course with their allowance table
3. Instructor makes the desired change	
	4. System stores the changes to the DB

Use Case: Upload Document
Actor: Instructor
Purpose: Instructor may put some supplementary material and homework to the DB

Actor Actions	**System Response**
1. Instructor selects the link	
	2.System shows a dialog box
3. Instructor specifies the files and uploads	
	4. System automatically sends a notification mail to the related courses newsgroup.

Use Case: Begin Virtual Class
Actor: Instructor
Purpose: The instructor starts the course

Actor Actions	System Response
1. Instructor clicks on the “start” button	
	2. System takes the instructor to the virtual classroom

Use Case: Create Course/Instructor/Student
Actor: Admin
Purpose: Admin will be responsible of creating entries in the DB

Actor Actions	System Response
1. Admin selects the link that is related to his/her action	
	2. System shows the creating interface
	3. System stores the data to the DB

Use Case: Draw on the WhiteBoard
Actor: Instructor, Student
Purpose: The instructor will use the whiteboard during the lecture in order to make points more clear. If instructor gives permission to the student then student will also use whiteboard.

Actor Actions	System Response
1. Instructor selects drawing tool and selects other options like size, color.	
2 . Instructor draws something on the board	
	3. The events(e.g. mouse move, drawn points coordinates) that instructor has done recorded by the system and they are send to all users in a certain time interval.
	4 . When the packets receive to the students their whiteboards will reflect the changes to the student.
5. If a student raises hand the instructor sends a signal to the server and that student then can use whiteboard and the same events take place.	

Use Case: Use Question/Answer Box
Actor: Instructor, Student
Purpose: Instructor and student may speak privately by this tool

Actor Actions	System Response

1. User(instructor/student) types the question/answer and sends.	
	2. System looks for the IP of the selected user and sends the message
3 . User sees the coming message	

Use Case: Write Something On Chat Tool
Actor: Student
Purpose: This tool will provide a real time discussion environment in the virtual classroom

Actor Actions	System Response
1. Student writes something on the chat console and submits "send" button	
	2. Server takes the message and looks IP of all online users.
	3. System sends the message to all online users.
	4. If any new user joins the class system sends a notification to all online users about newcomer.

Use Case: Edit Student Allowance
Actor: Instructor
Purpose: The system will give option instructor to ban some students from class or give permission them to use whiteboard.

Actor Actions	System Response
1. If a student raise hand instructor will edit the students allowance table and give permission that student to use whiteboard	
	2. The system will receive the signal from instructor and specified student's drawing toolbar will be "enabled"
3. If any student disturb others instructor will send a signal to take that student out of class	
	4. When system takes signal it will throw out the student from classroom.
	5. The students username will be removed from studentlist

Use Case: View Student List
Actor: Instructor, Student

Purpose: The instructor and students will see all online users

Actor Actions	**System Response**
1. The user will see all online users	

Use Case: Capture Instructor View
Actor: Camera
Purpose: A camera will record the instructor and broadcast it to all class

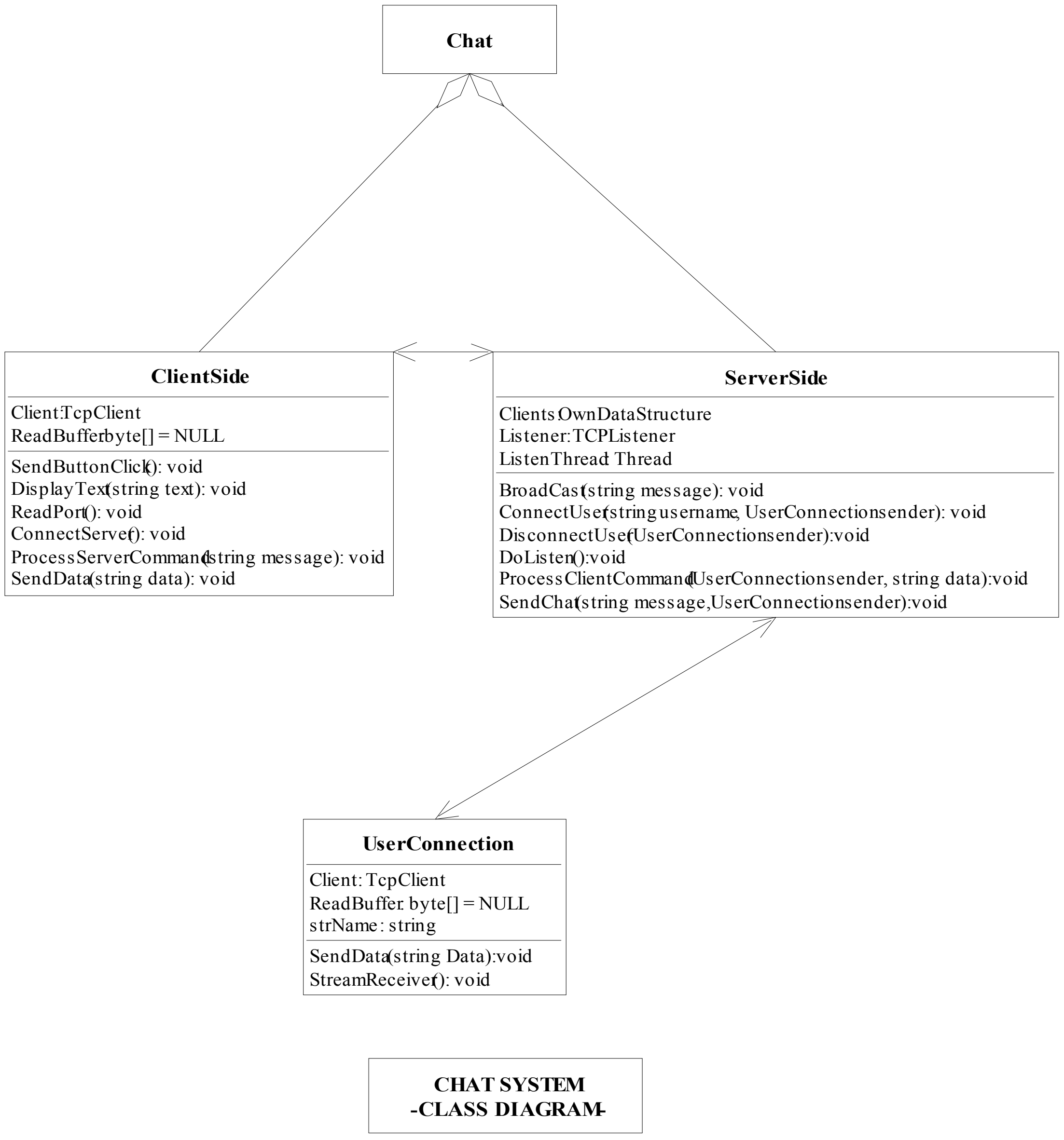
Actor Actions	**System Response**
1. The camera will capture the view of the instructor	
	2. The captured images will be compressed
	3. The packets will send to all users
	4. When users get the packets the system will decompress it
	5 . Students will watch the instructor in real time

Use Case: Capture Instructor Voice
Actor: Microphone
Purpose: A microphone will record the instructor's voice and broadcast it to all class.

Actor Actions	**System Response**
1. The microphone will capture the voice of the instructor	
	2. The captured data will be compressed
	3. The packets will send to all users
	4. When users get the packets the system will decompress it
	5 . Students will listen the instructor in real time

3.2 CLASS DIAGRAMS

3.2.1 Chat Module:


Chat
ClientSide
Client:TcpClient
ReadBuffer:byte[] = NULL
SendButtonClick(): void
DisplayText(string text): void
ReadPort(): void
ConnectServer(): void
ProcessServerCommand(string message): void
SendData(string data): void
ServerSide
Clients:OwnDataStructure
Listener:TCPListener
ListenThread: Thread
BroadCast(string message): void
ConnectUser(string username, UserConnection sender): void
DisconnectUser(UserConnection sender):void
DoListen():void
ProcessClientCommand(UserConnection sender, string data):void
SendChat(string message,UserConnection sender):void
UserConnection
Client: TcpClient
ReadBuffer: byte[] = NULL
strName: string
SendData(string Data):void
StreamReceiver(): void
CHAT SYSTEM
-CLASS DIAGRAM-

We designed the “chat tool” with three different class structures. The first class is the “ClientSide” that will execute in clients and the second and third classes “ServerSide” and “UserConnection” will execute in the server side.

Class Name: ClientSide

Attributes:

1) **Client:TcpClient**
2) **ReadBuffer: Byte[]**

Methods:

1) **ReadPort ():** This method will read the incoming data to the ReadBuffer array.
2) **ProcessServerCommand (string message):** This method will split the incoming message and will determine the type of the message that is a chat message or any information message like "x joined the class".
3) **DisplayText ():**This method will show the incoming or sent message in the clients chat console.
4) **SendData ():** This will send the written message to the server and server will handle the message.
5) **SendButtonClick ():** This is the button event takes place when user clicks the "send" button.
6) **ConnectServer ():** This is the method called when a new user connects to the classroom.

Class Name: ServerSide

Attributes:

1) **ListenThread: Thread**
2) **Clients: Our data structure**
3) **Listener: TCPListener**

Methods:

1) **Broadcast (string Message):** Sends the message to all attached clients
2) **DoListen ():** This method will listen the specified port for incoming messages from clients.
3) **ConnectUser(string username, UserConnection sender) :** This method will inform all the clients if a new user joins the classroom.
4) **DisconnectUser (string username, UserConnection sender):** This method will inform all the clients if a user exits the classroom.
5) **ProcessClientCommand (string username, UserConnection sender):** This method will split the incoming message of a client and will act according to that command

6) **SendChat (string message, UserConnection sender):** This method will send the message to the clients with sender information.

Class Name: UserConnection

In our ServerSide class a thread will work named ListenThread and if any connection is found a new UserConnection object will be created.

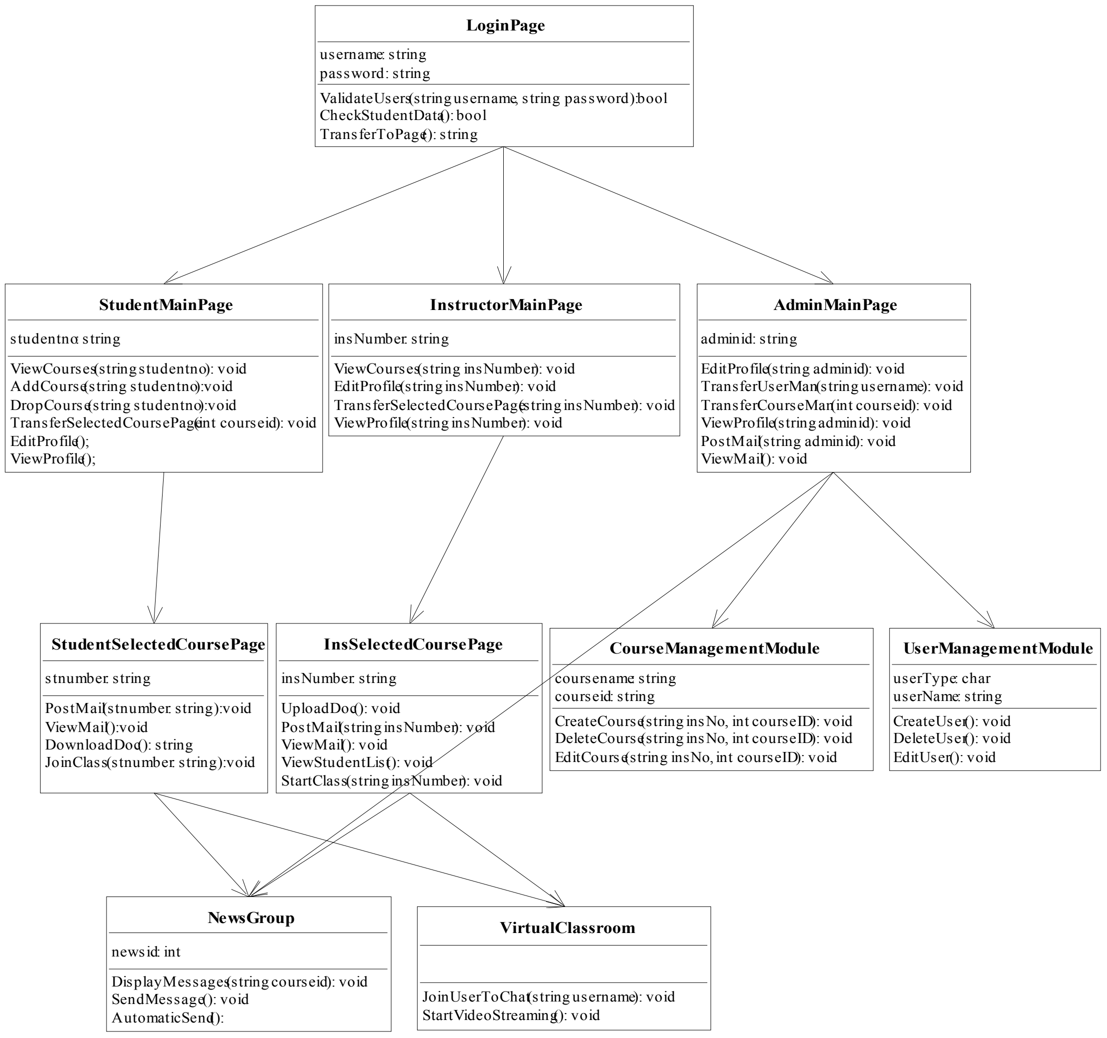
Attributes:

1) **Client: TCPClient**
2) **ReadBuffer: Byte[]**
3) **StrName: String**

Methods:

1) **StreamReceiver ():** This method will begin a asynchronous read from the stream
2) **SendData (string message):** This method will send the message to the user.

3.2.2 User Management System (UMS) Module:


LoginPage
username: string
password: string
ValidateUsers(string username, string password):bool
CheckStudentData(): bool
TransferToPage(): string
StudentMainPage
studentno: string
ViewCourses(string studentno): void
AddCourse(string studentno):void
DropCourse(string studentno):void
TransferSelectedCoursePage(int courseid): void
EditProfile();
ViewProfile();
InstructorMainPage
insNumber: string
ViewCourses(string insNumber): void
EditProfile(string insNumber): void
TransferSelectedCoursePage(string insNumber): void
ViewProfile(string insNumber): void
AdminMainPage
adminid: string
EditProfile(string adminid): void
TransferUserMan(string username): void
TransferCourseMan(int courseid): void
ViewProfile(string adminid): void
PostMail(string adminid): void
ViewMail(): void
StudentSelectedCoursePage
stnumber: string
PostMail(stnumber: string):void
ViewMail():void
DownloadDoc(): string
JoinClass(stnumber: string):void
InsSelectedCoursePage
insNumber: string
UploadDoc(): void
PostMail(string insNumber): void
ViewMail(): void
ViewStudentList(): void
StartClass(string insNumber): void
CourseManagementModule
coursename: string
courseid: string
CreateCourse(string insNo, int courseID): void
DeleteCourse(string insNo, int courseID): void
EditCourse(string insNo, int courseID): void
UserManagementModule
userType: char
userName: string
CreateUser(): void
DeleteUser(): void
EditUser(): void
NewsGroup
newsid: int
DisplayMessages(string courseid): void
SendMessage(): void
AutomaticSend():
VirtualClassroom
JoinUserToChat(string username): void
StartVideoStreaming(): void

USER MANAGEMENT SYSTEM
-CLASS DIAGRAM-

This component of the system starts when the user enters the system and finishes when the instructor or students enters the virtual classroom system. The main purpose of this component is control all of the users activity. As you can the above class diagram we have 9 classes and one class, which provides the transition between this system and virtual classroom system.

Class Name: Login Page

This class checks the username and password of the users and checks whether the student information (in database) is completed and also makes the transition between the other classes. If the user info is not completed then this system enforces the student to complete their information.

Attributes:

1) **username: string**
2) **password: string**

Methods:

1) **ValidateInput():** This method checks the username and password, and also the user type of the person are recognized and this user is student then call the CheckStudentData() function.
2) **CheckStudentData():** This method is called by the ValidateInput() function and check whether the student information is completed.
3) **TransferPage():** This method is used for transferring the user some classes according to the user type.

Class Name: StudentMainPage

The main goal of **StudentMainPage** class is organize the student activities such as add/drop course and edit/view profile. And also transfer the user the other classes.

Attributes:

1) **studentno: String**

Methods:

1) **ViewCourses(string studentno):** When the user enters this page, this function checks whether entered student takes course then display the course list that is taken by this student.
2) **AddCourse(string studentno):** When the student want to take a course then this function is activated.
3) **DropCourse(string studentno):** When the student wants to drop a course then this function is activated.
4) **TransferSelectedCoursePage(int courseid):** There are links all of the courses, which is taken by student, and when the student press this link

button then this function is activated. Then user is transferred to the StudentSelectedCoursePage class.

5) **EditProfile():** When the student wants to update information, and then press the EditProfile button then this function is activated.

6) **ViewProfile():** When the student want to see information, then press the ViewProfile button then this function is activated.

Class Name: InstructorMainPage

The main purpose of InstructorMainPage class is organizing the instructor activities such as View course, View/Edit profile and also transfer the instructor the other classes.

Attributes:

1) insNumber: string

Methods:

1) **ViewCourses(string studentno):** When the user enters this page, this function displays the course list, which is given by this instructor.
2) **TransferSelectedCoursePage(int courseid):** There are links all of the courses, which is given by instructor, and when the instructor presses this link button then this function, is activated. Then user is transferred to the InstructorSelectedCoursePage class.
3) **EditProfile():** When the instructor wants to update information, and then press the EditProfile button then this function is activated.
4) **ViewProfile():** When the instructor want to see information, then press the ViewProfile button then this function is activated.

Class Name: AdminMainPage

The main purpose of AdminrMainPage class is organizing the admin activities such as View/Edit profile, View/post mail and also transfer the admin the other classes.

Attributes:

1) insNumber: string

Methods:

1) **ViewMail():** This function transfers the admin to the NewsGroup class. Then admin can see all of the news about all classes and also can see the general news.
2) **PostMail():** This function transfers the admin to the NewsGroup class. Then admin can post to all classes newsgroup and can also post to the general newsgroup.
3) **TransferCourseMan(int courseid):** This function transfers the admin to the CourseManagement Module class.
4) **TransferUserMan(int courseid):** This function transfers the admin to the UserManagement Module class.
5) **EditProfile():** When the admin want to update information, and then press the EditProfile button then this function is activated.
6) **ViewProfile():** When the admin want to see information, and then press the ViewProfile button then this function is activated.

Class Name: StudentSelectedCoursePage()

This class provides student makes some operations such as downloading the course material or post/view mail of this course or joining the virtual classroom.

Attributes:

1) stnumber: String

Methods:

1) **ViewMail():** This function transfers the student to the NewsGroup class. Then student can see all of the news about only this class and also can see the general news.
2) **PostMail(string stnumber):** This function transfers the student to the NewsGroup class. Then student can post a mail to this class newsgroup or to a general newsgroup.
3) **DownloadDoc():** This function is activated when the user press the download document link.
4) **JoinClass():** If this course is online, then JoinClass button is active, if the user press this button then this function is activated.

Class Name: InsSelectedCoursePage()

This class provides instructor makes some operations such as uploading the course material or post/view mail of this course or views the student list or starts the virtual classroom.

Attributes:

1) **insNumber: String**

Methods:

2) **ViewMail():** This function transfers the instructor to the NewsGroup class. Then instructor can see all of the news about only this class and also can see the general news.
3) **PostMail(string stnumber):** This function transfers the instructor to the NewsGroup class. Then instructor can post a mail to this class newsgroup or to a general newsgroup.
4) **UploadDoc():** This function is activated when the instructor press the upload document link. If the uploaded document is assignment then this system send some info to the newsgroup to activate AutomaticSend() function which send mail to this course page, which is the notification about the assignment.
5) **StartClass():** If this course is online, then StartClass button is active, if the instructor press this button then this function is activated.
6) **ViewStudentList():** If the instructor requests to see the list of the student in this course then presses the related button then this function is activated.

Class Name: CourseManegementModule()

Admin can create/delete/edit the courses by using this module.

Attributes:

1) **coursename: string**
2) **courseid: string**

Methods:

1) **CreateCourse(string coursename, string courseid):** By using this function admin can create new course then this function update the database.
2) **EditCourse(string coursename, string courseid):** By using this function admin can edit new course then this function update the database.

3) **DeleteCourse(string coursename, string courseid):** By using this function admin can delete new course then this function update the database.

Class Name: UserManegementModule()

Admin can create/delete/edit the users by using this module.

Attributes:

1) **usertype: char**
2) **username: string**

Methods:

1) **CreateUser(string coursename, string courseid):** By using this function admin can create new user then this function update the database.
2) **EditUser(string coursename, string courseid):** By using this function admin can edit new user then this function update the database.
3) **DeleteUser(string coursename, string courseid):** By using this function admin can delete new user then this function update the database.

Class Name: NewsGroup()

This class organizes the messages(mails) of system.

Attributes:

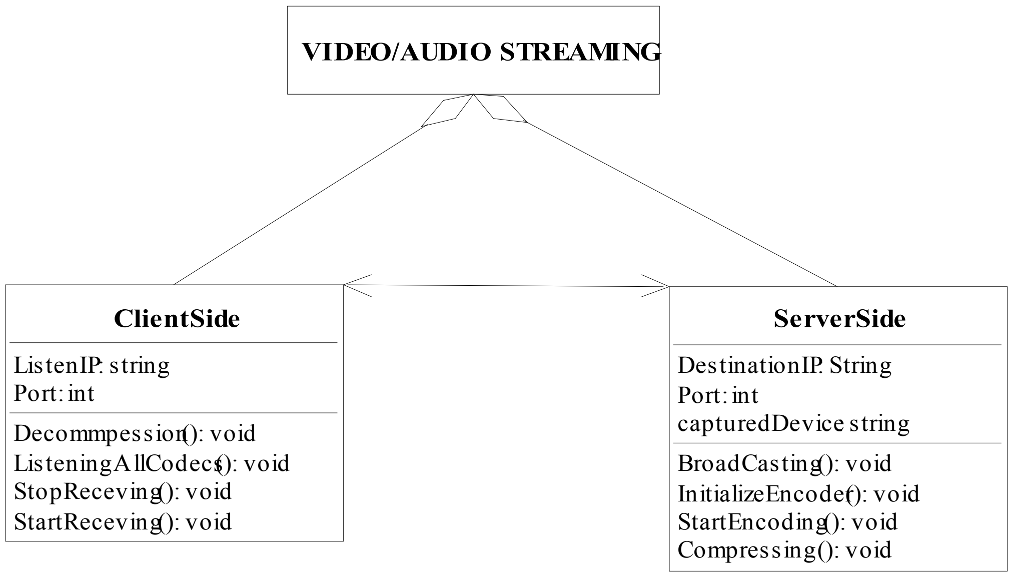
1) **newsid: int**

Methods:

1) **DisplayMessages(string courseid):** This function displays the requested messages according to the courseid.
2) **SendMessage(string stnumber):** This function sends the requested messages according to the courseid.
3) **AutomaticSend():** When the instructor downloads an assignment to the system then this function is activated automatically.

And the other class is “VirtualClassroom” class, which is transition between this system and the virtual classroom system. When this class is alerted, then the requested user automatically join the chat if he/she is allowed, and if he/she is a teacher then streaming module also activated by the JoinUserToChat() and StartVideoStreaming() functions.

3.2.3 Video/Audio Streaming Module:



VIDEO/AUDIO STREAMING
-CLASS DIAGRAM-

The first thing this system does is capturing the movie to a file on our server computer. This process is called digitizing or capturing. Then this system converts our analog audio and video to a digital form that can be saved as a file.

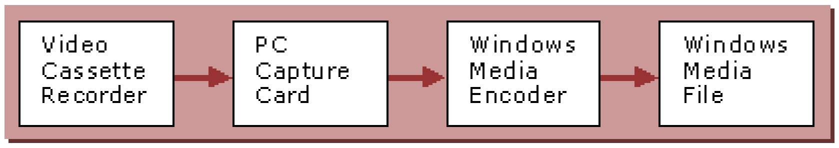


Figure 1. The process of capturing and converting audio and video

For streaming this captured data we use the Windows Media Server. Besides that our server computer running Microsoft Internet Information Services (IIS) on Microsoft Windows 2003 Server can be configured as a Web server for distributing Web pages, images, and other files to thousands of clients on the Internet or an intranet.

The followings are the basic operations for video/audio streaming:

- Capture and Convert
- Compressing
- Encoding
- Decoding
- Decompressing

Capturing, converting, compressing, encoding are the server part of operations and the others the client part of operations. The followings are important concept of this system:

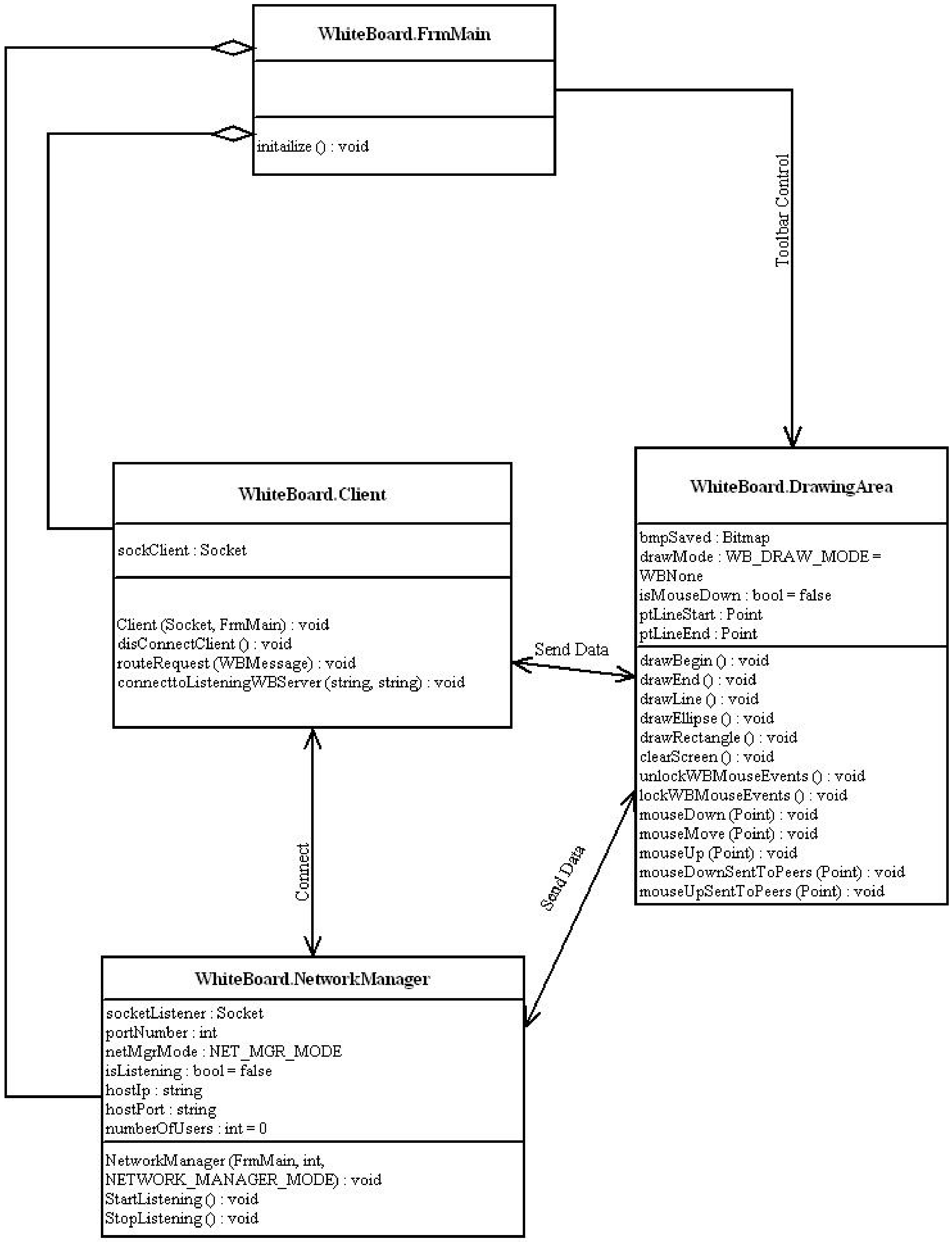
- **Windows Media Encoder.** The basic encoding program. Capture audio and video, encode files and live streams, and switch input sources on the fly.
- **Simple embedded Player.** Add an embedded Windows Media Player control and plug-in to a Web page, and configure the Player to initiate a stream as soon as the page is opened. When a user opens the Web page, the embedded Player opens and plays the stream.
- **Live broadcasting.** You can stream a live broadcast over a network with Windows Media Services. Use Windows Media Encoder to encode the live stream and your Windows Media server to deliver it.

And for this system the most important point is SDK (Software Development Kit), which is used for creating own design. For making this we use the Interfaces of the Windows Media Encoder and Windows Media Services. And we are planning to chose **Windows Media Player 10 SDK.**

We can do the following operations by using SDK:

- Create custom skins and visualizations, and embed the Player into programs and Web pages.
- Enable software applications to read, write, edit and transfer files with the Windows Media Format.
- Add Windows Media playback to portable digital music players, Internet appliances, and other embedded systems.
- Automate the operation of Windows Media Encoder.
- Enable distribution of protected content and licenses using DRM technology.
- Configure, manage, and administer Windows Media Services.

3.2.4 Whiteboard Module:



WHITEBOARD SYSTEM
- CLASS DIAGRAM-

This component starts when the user enters the virtual classroom system and finishes when the instructor or students exits the virtual classroom system. As you can see from the class diagram above, this component contains 4 classes.

Class Name: FrmMain

This class contains the variables and methods used in the creation of the form (Buttons, labels, etc.)

Methods:

4) **initialize ():** This method initializes whiteboard.

Class Name: Network Manager

This class provides necessary methods for networking management. It controls the server-side operations and communicates with the clients. It also stores the number of clients connected to the server.

Attributes:

7) **socketListener: Socket**
8) **portNumber: int**
9) **netMgrMode: NET_MGR_MODE**(*)
10) **isListening: bool**
11) **hostIP: string**
12) **hostPort: string**
13) **numberOfUsers: int**

Methods:

1) **NetworkManager (FrmMain mForm, int portN, NET_MGR_MODE wbMode):** This is the constructor of the NetworkManager class.
2) **StartListening ():** This method starts listening to the port created by the NetworkManager.
3) **StopListening ():** This method forces the server to stop listening to a port. Therefore the network traffic managed by the server is broken and all the clients connected to the server are forced to disconnect.

Class Name: Client

This class contains methods used for the client-side operations of the whiteboard.

Attributes:

1) **sockClient: Socket**

Methods:

1) **Client (Socket socketClient, FrmMain ClientForm):** This is the constructor of the Client class. socketClient is the socket and ClientForm is the form created for the use of the client.
2) **ConnecttoListeningServer (string HostIP, string HostPort):** This method connects the client to the server listening to port HostPort and having the IP address HostIP.
3) **disConnectClient ():** This method disconnects the client that is connected to the server.
4) **routeRequest (WBMessage(**)):** This method maintains the singleton property of the whiteboard that is when a user is drawing on the whiteboard, no other users can access to the drawing methods of the whiteboard. This method calls the method lockWBMouseEvents () of DrawingArea class to ensure that the whiteboards of all the other users is locked when a user is drawing on the whiteboard.

Class Name: DrawingArea

This class creates a bitmap image to store the drawings. This image is overridden each a user draws an image.

Attributes:

1) **bmpSaved: Bitmap**
2) **drawMode: WB_DRAW_MODE(***)**
3) **isMouseDown: bool**
4) **ptLineStart: Point**
5) **ptLineEnd: Point**

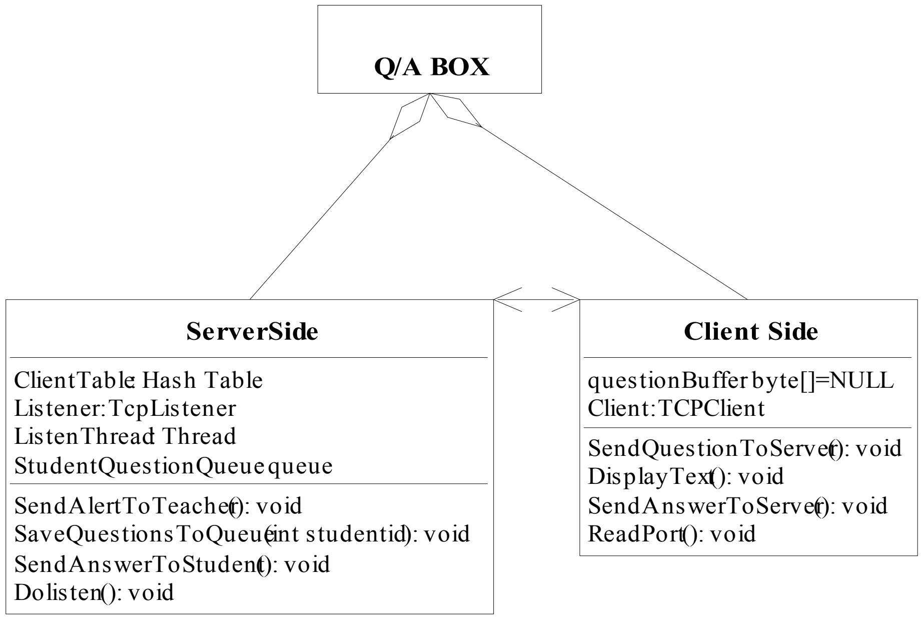
Methods:

1) **drawBegin ():** This method sends the information that a user started to draw something on the whiteboard, to the network manager.
2) **drawEnd ():** This method sends the information that a user finished drawing something on the whiteboard, to the network manager.
3) **drawLine ():** This method is called when the user pressed DrawLine button on the toolbar of the whiteboard. It tells the system that the user is or will be drawing a line.
4) **drawEllipse ():**This method is called when the user pressed DrawEllipse button on the toolbar of the whiteboard. It tells the system that the user is or will be drawing an ellipse.
5) **DrawRectangle ():** This method is called when the user pressed DrawRectangle button on the toolbar of the whiteboard. It tells the system that the user is or will be drawing a rectangle.
6) **clearScreen ():**This method is called when the user pressed clearScreen button on the toolbar of the whiteboard. It clears the screen of the whiteboard.
7) **unlockWBMouseEvents ():** This method is called to unlock the whiteboard so that the user can access the drawing functions of the whiteboard.
8) **lockWBMouseEvents ():** This method is called to lock the whiteboard so that the user cannot access the drawing functions of the whiteboard. This method together with the lockWBMouseEvents method maintains the singleton property of the whiteboard.
9) **mouseDown (Point pt):** This method stores the coordinates of the point when the left mouse button is pressed.
10) **mouseMove ():** This method is called to draw on the whiteboard when the left mouse button is pressed.
11) **mouseUp ():**This method stores the coordinates of the point when the left mouse button is released, that is it stores the end point of the drawing.
12) **mouseDownSentToPeers (Point pt):** This method sends the coordinates of the point when the left mouse button is pressed to the connected peers so that the start point of the drawing is identified by the other users.

13) **mouseUpSentToPeers (Point pt):** This method sends the coordinates of the point when the left mouse button is released to the connected peers so that the end point of the drawing is identified by the other users.

(*) **NET_MGR_MODE** is the status of the peers.(enum NET_MGR_MODE{serverMode, clientMode};)
(**) **WBMessage** is the message type that identifies the drawing status of the whiteboard
(enum WHITEBOARD_MESSAGE{WBBegin, WBLine, WBRectangle, WBEllipse, WBClearScreen, WBEnd};)
(***) **WB_DRAW_MODE** is the drawing status of the toolbar of the whiteboard.
(enum WHITEBOARD_DRAW_MODE { ModeLine, ModeRect, ModeEllipse, WBNone};)

3.2.5 Q/A Box Module:


Q/A BOX
ServerSide
ClientTable: Hash Table
Listener:TcpListener
ListenThread: Thread
StudentQuestionQueue queue
SendAlertToTeacher(): void
SaveQuestionsToQueue(int studentid): void
SendAnswerToStudent(): void
Dolisten(): void
Client Side
questionBuffer byte[]=NULL
Client:TCPClient
SendQuestionToServer(): void
DisplayText(): void
SendAnswerToServer(): void
ReadPort(): void

Q/A BOX SYSTEM -CLASS DIAGRAM-

This component provides the interaction between the instructor and students. The students can ask questions to the instructor and the instructor can answer them simultaneously.

Class Name: ServerSide

This class provides the connection between the server and the users. The basic operations of this class are explained below.

Attributes:

2) **ClientTable: HashTable**
3) **Listener: TcpListener**
4) **ListenThread: Thread**
5) **StudentQuestionQueue: queue**

Methods:

5) **SendAlertToTeacher():** When the students sends a question to the server, the instructor is notified by this method.
6) **SaveQuestionsToQueue(int studentid):** The question received from the student is saved to this student's queue by this method.
7) **SendAnswerToStudent():**When the instructor sends an answer to the server, this method sends this answer to the student who asked this question.
8) **Dolisten():** This method listens to the specified port to see whether a new message came.

Class Name: ClientSide

This class provides the client-side operations such as sending questions or answers to the server.

Attributes:

1) **questionBuffer: byte**
2) **Client: TCPClient**

Methods:

1) **SendQuestionToServer():** This method sends the question and also studentid to the server part.
2) **DisplayText():** This method will show the sent questions in the q/a box console.
3) **SendAnswerToServer():**This method sends the answer and also studentid to the server part.
4) **ReadPort():** This method reads the specified port.

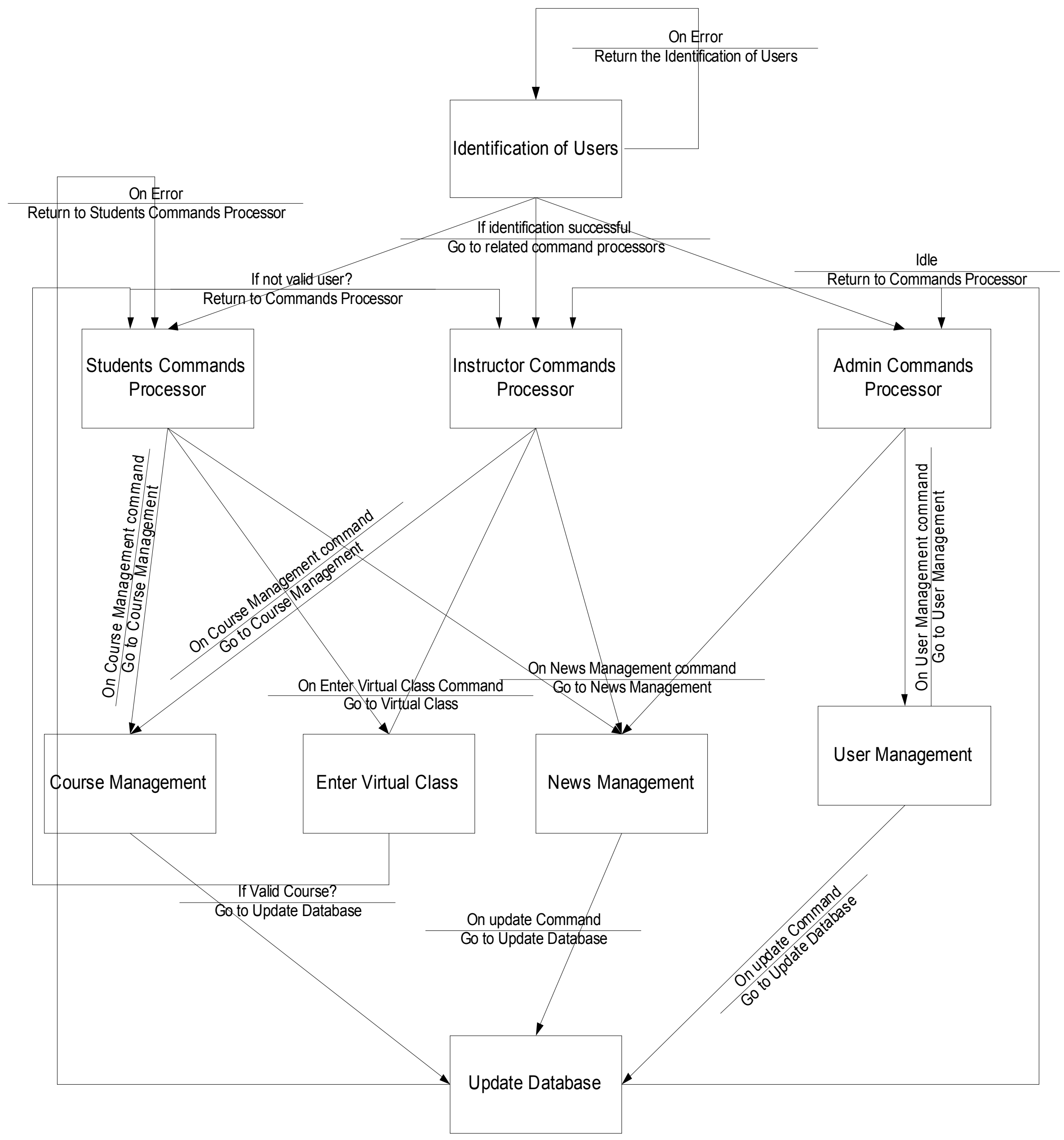
3.3 STATE TRANSITION DIAGRAM

In State Transition Diagram, we showed that if an interrupt occurs the states change into rejection state to display error message and there exists return to every state from rejection state. We examine state transition diagram into two parts as we did before.

In the first part we examine the User Control System, after that we examine the second component of the system, which is Virtual Classroom System

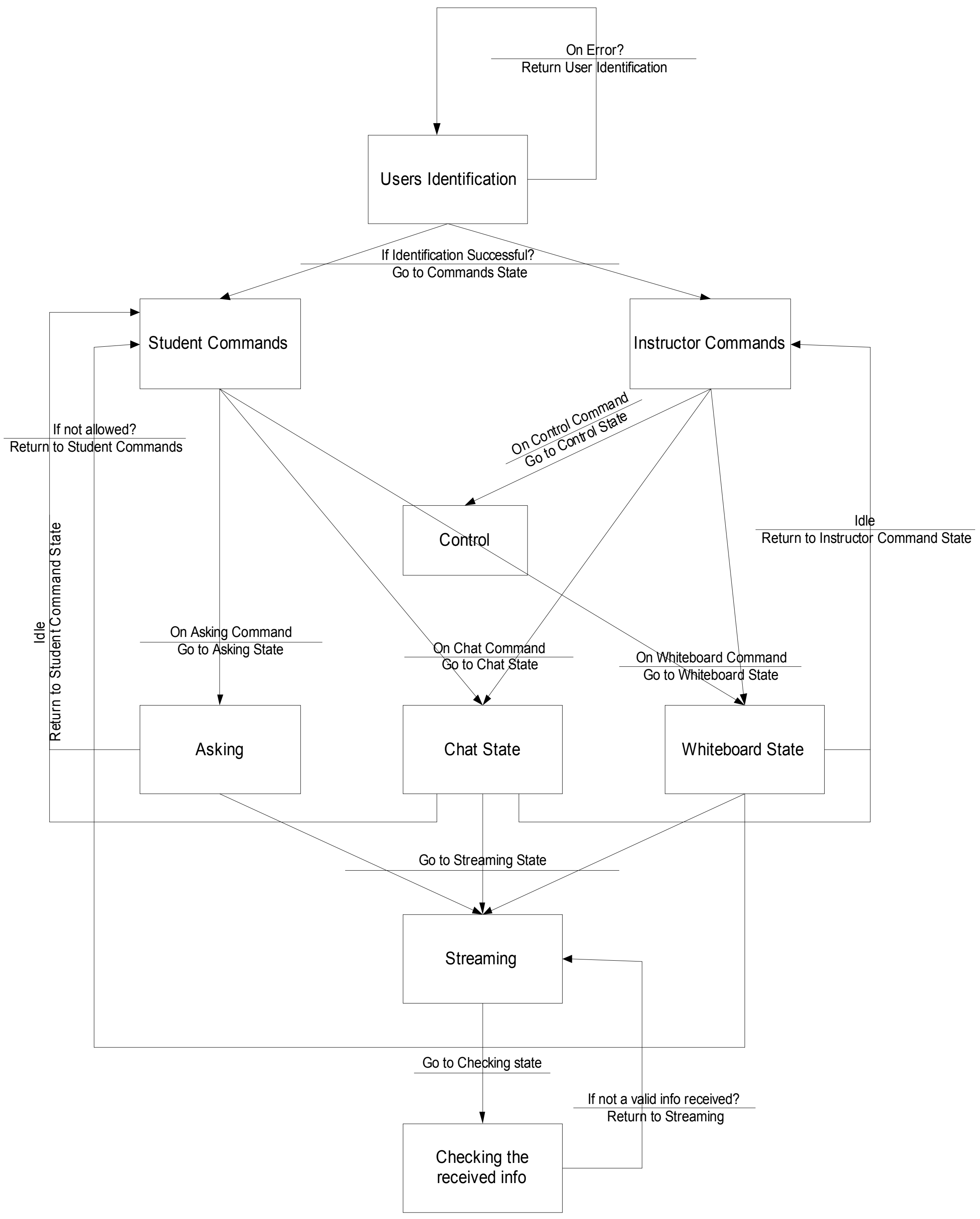
3.3.1 User Management System STD:

As you can see the following diagram, we have 9 states one is the connector between this system and the virtual classroom system. First state of the system is “identification of the users” which provides the users entrance to the system. In this state the system recognizes the users type whether he/she is instructor or student or admin, then the system decides the next state according to the user type. The other state makes the general process of the system and if required then update the system.



3.3.2 Virtual Classroom STD:

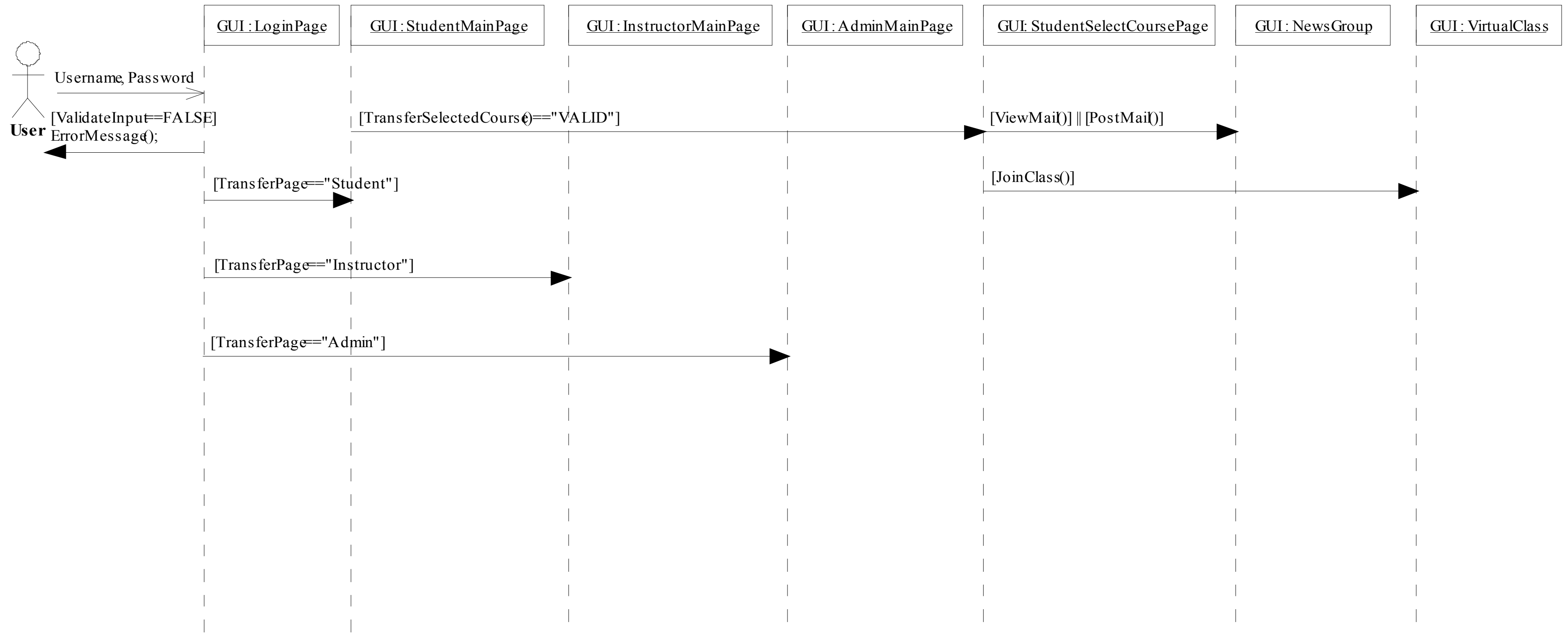
In this state transition diagram, we also have 9 states. The first state of this diagram is also “identification to user”. If the user access is granted, then this system is recognize the user and some of the tool are disabled if this user is student. But if this person is instructor all of the tools of the system are enabled. Then system is go to “student commands” or “instructor state” according to the type of the person. As you can see the following diagram all of the information goes to the streaming state. And then besides the video/audio info, in this state chat, question and also drawing info also sends to the users into the some pockets of info in some interval. If the received info is damaged, then system returns to the streaming state.

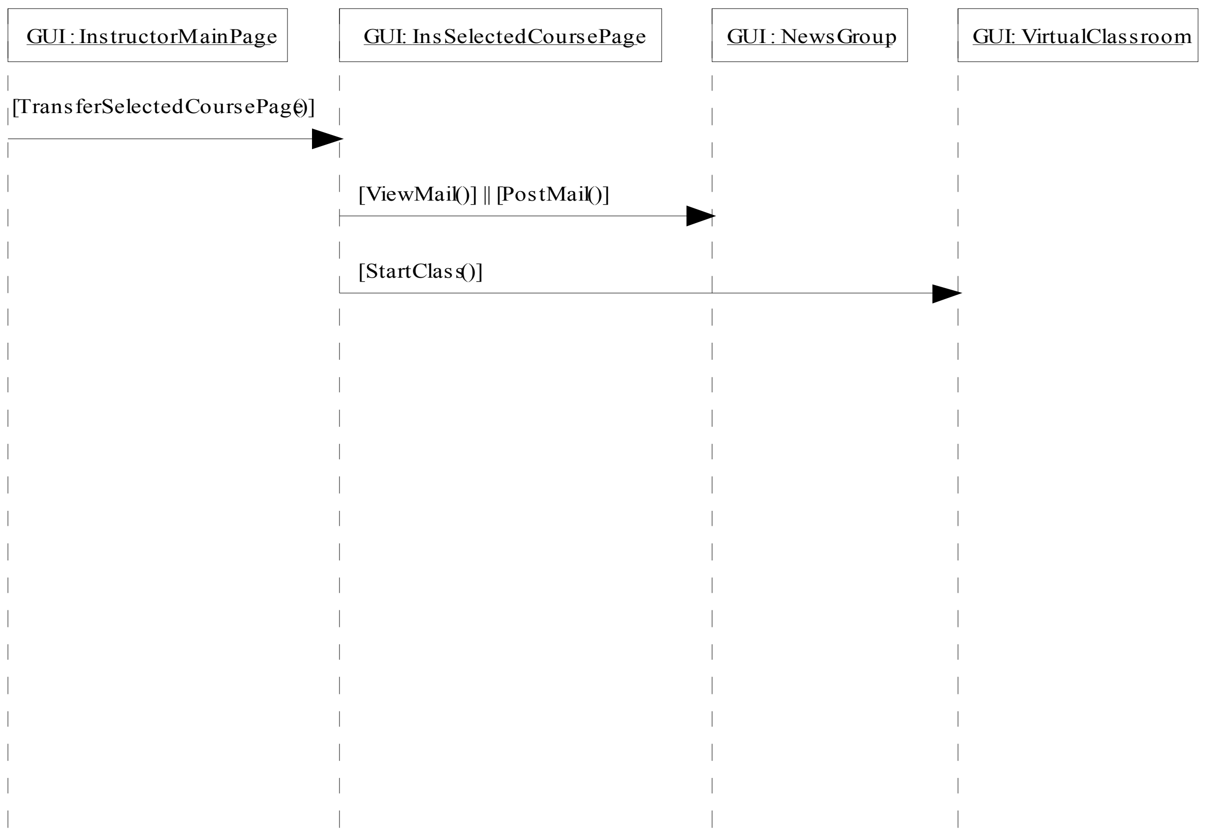


On Error?
Return User Identification
Users Identification
If Identification Successful?
Go to Commands State
Student Commands
Instructor Commands
On Control Command
Go to Control State
If not allowed?
Return to Student Commands
Control
Idle
Return to Instructor Command State
Idle
Return to Student Command State
On Asking Command
Go to Asking State
On Chat Command
Go to Chat State
On Whiteboard Command
Go to Whiteboard State
Asking
Chat State
Whiteboard State
Go to Streaming State
Streaming
Go to Checking state
If not a valid info received?
Return to Streaming
Checking the received info

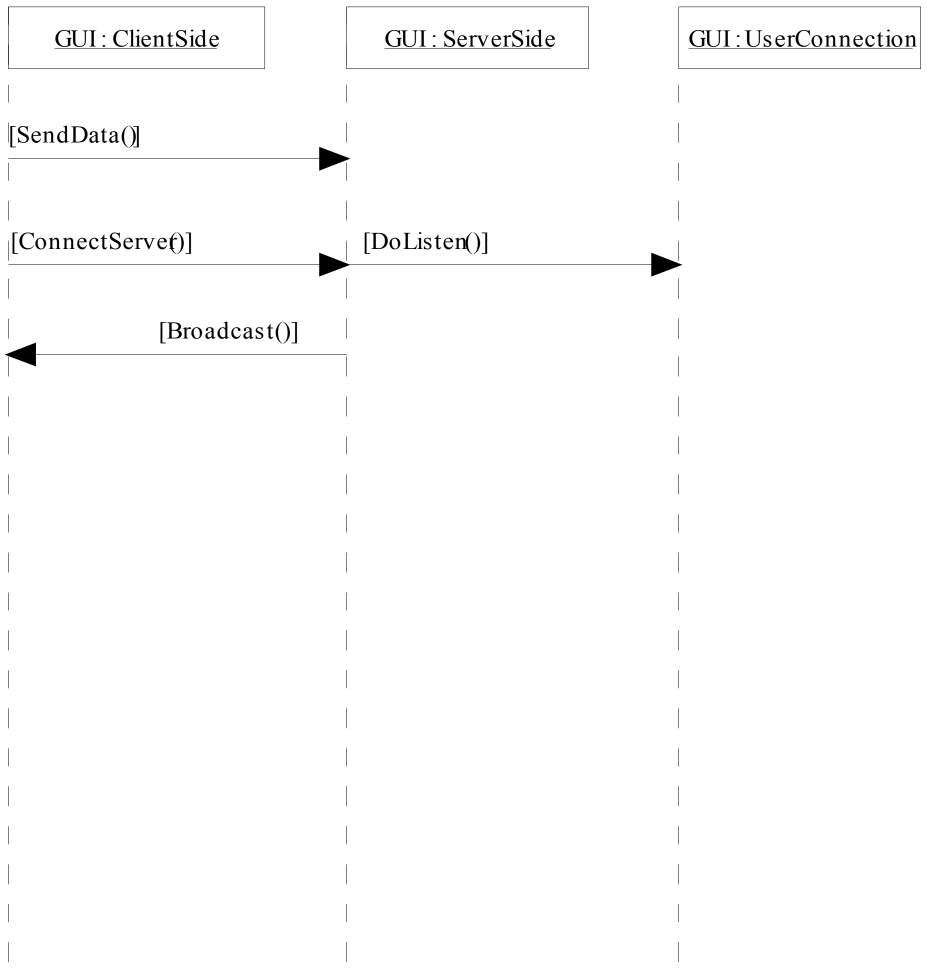
3.4 SEQUENCE DIAGRAMS

3.4.1 User Management System

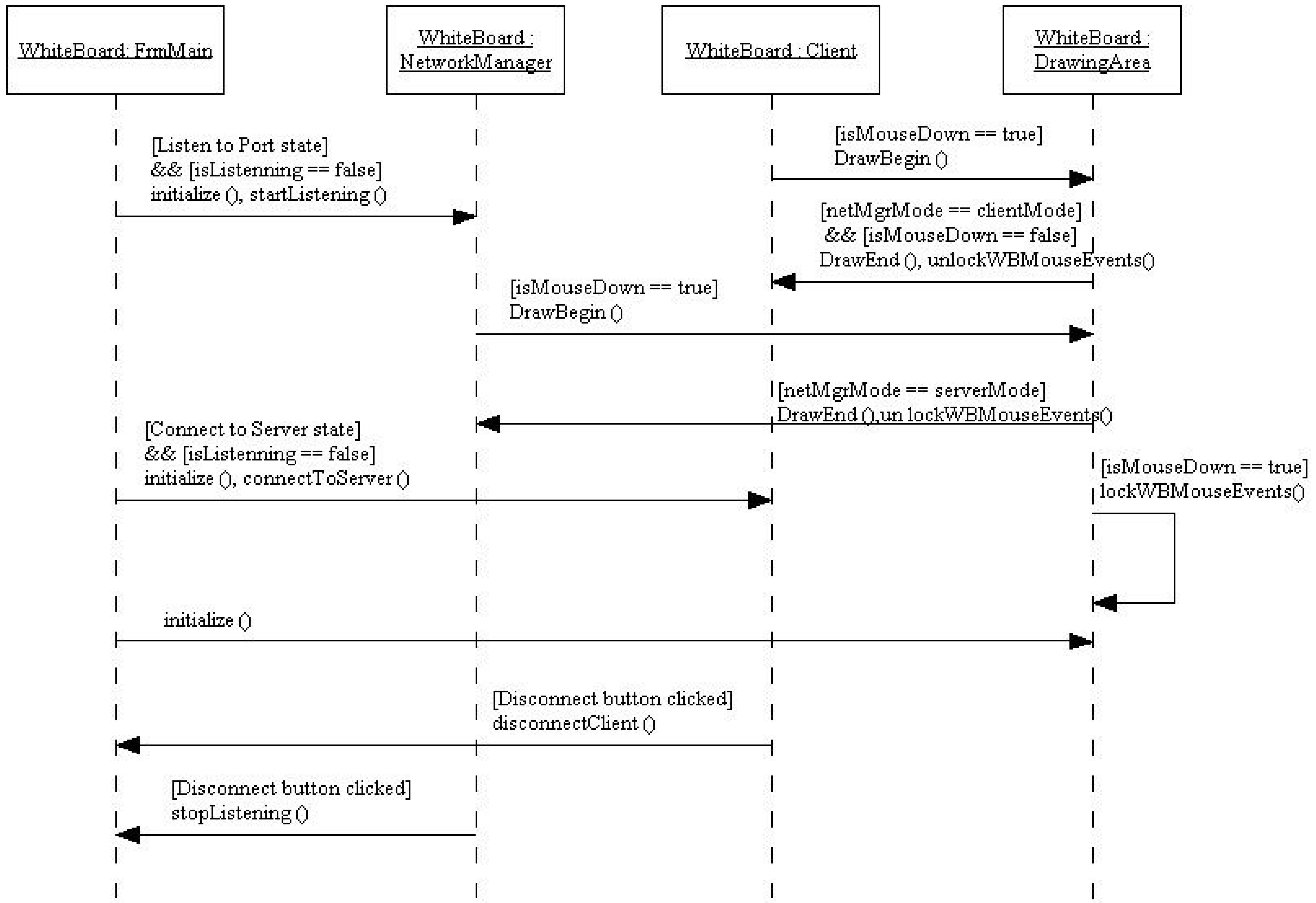



GUI : InstructorMainPage
GUI: InsSelectedCoursePage
GUI : NewsGroup
GUI: VirtualClassroom
[TransferSelectedCoursePage()]
[ViewMail()] || [PostMail()]
[StartClass()]

3.4.2 Chat System:

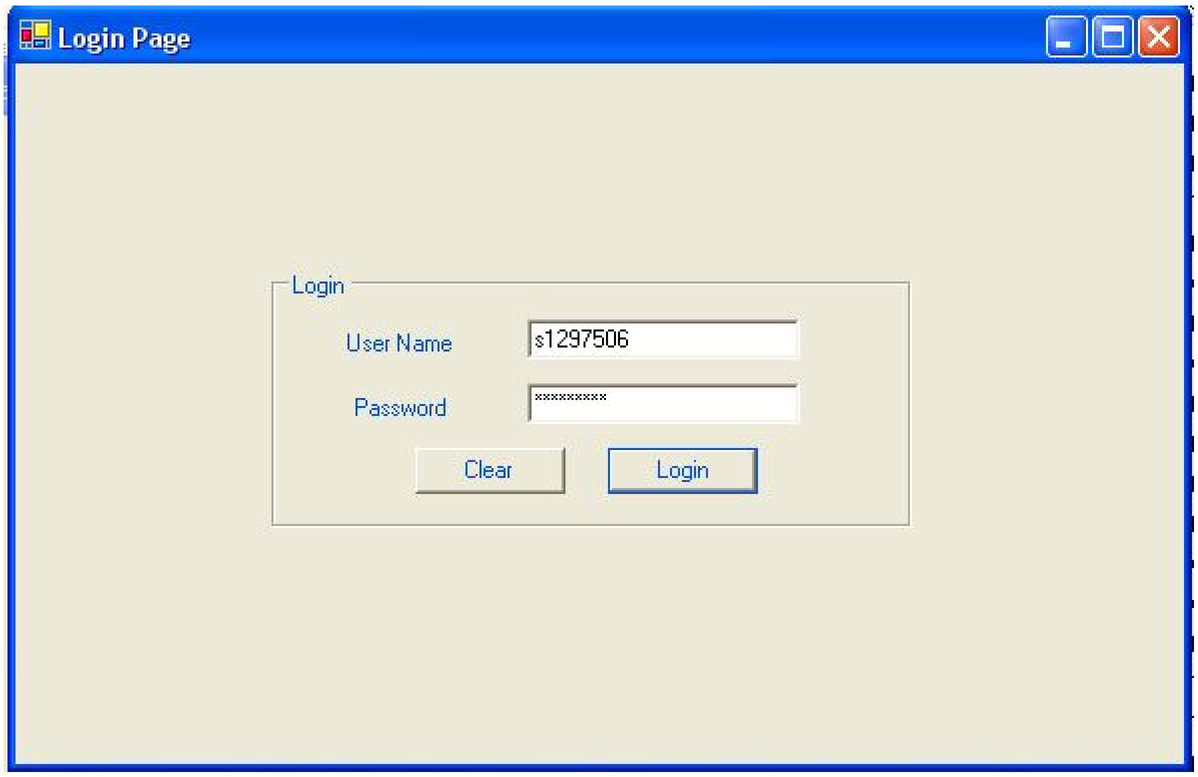


3.4.3 Whiteboard System:


WhiteBoard: FrmMain
WhiteBoard : NetworkManager
WhiteBoard : Client
WhiteBoard : DrawingArea
[isMouseDown == true] DrawBegin ()
[Listen to Port state] && [isListenning == false] initialize (), startListening ()
[netMgrMode == clientMode] && [isMouseDown == false] DrawEnd (), unlockWBMouseEvents()
[isMouseDown == true] DrawBegin ()
[netMgrMode == serverMode] DrawEnd (),un lockWBMouseEvents()
[Connect to Server state] && [isListenning == false] initialize (), connectToServer ()
[isMouseDown == true] lockWBMouseEvents()
initialize ()
[Disconnect button clicked] disconnectClient ()
[Disconnect button clicked] stopListening ()

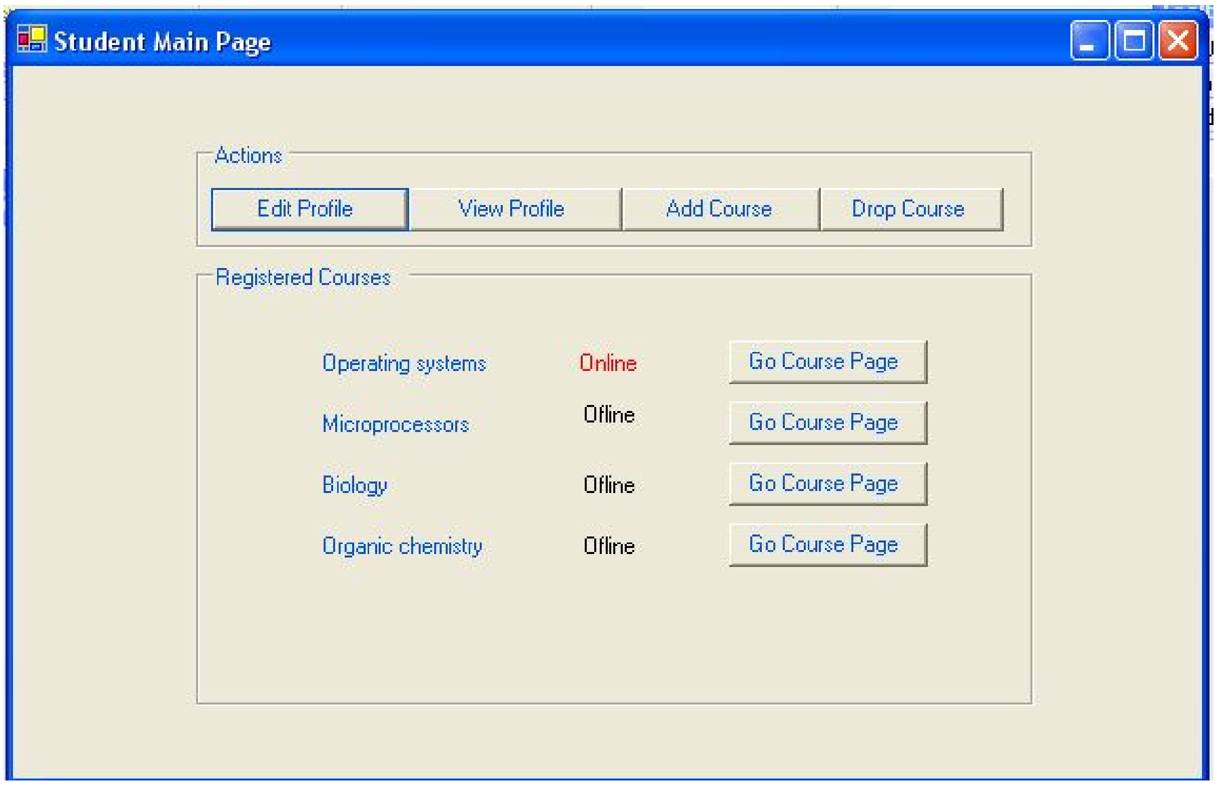
4. GRAPHICAL USER INTERFACES (GUI)

4.1 Login Page



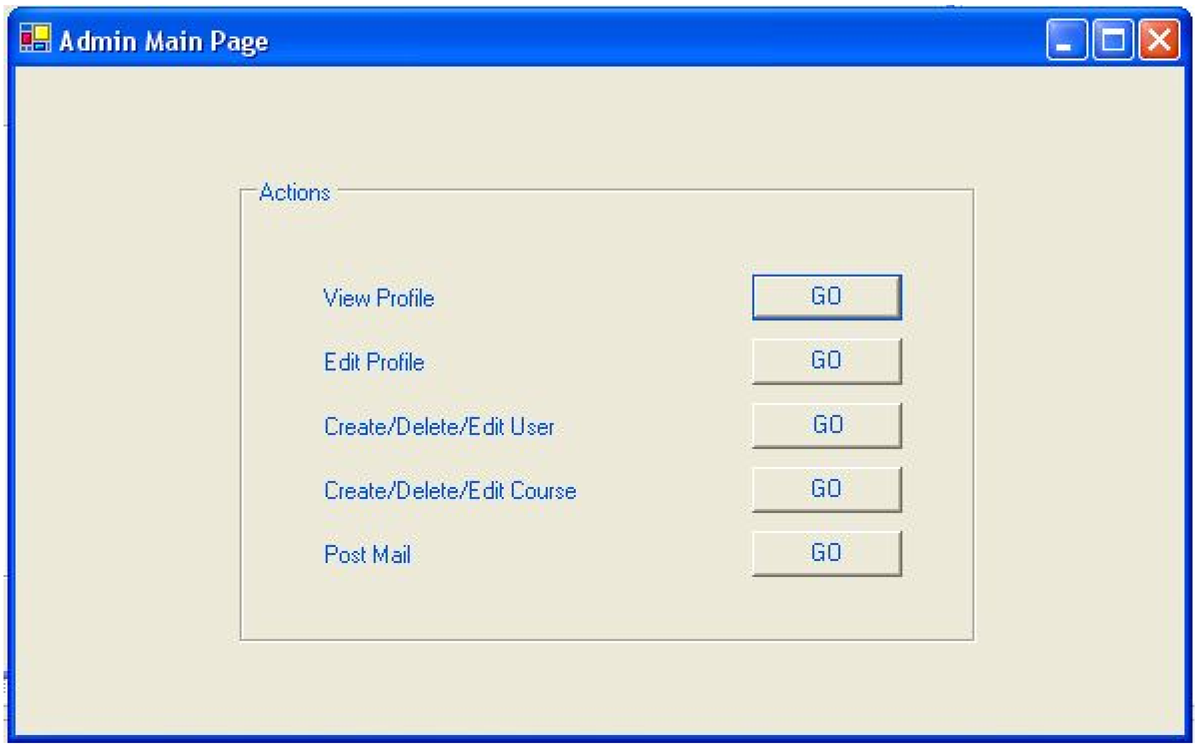
User Name: s1297506
Password: *********

4.2 Student Main Page

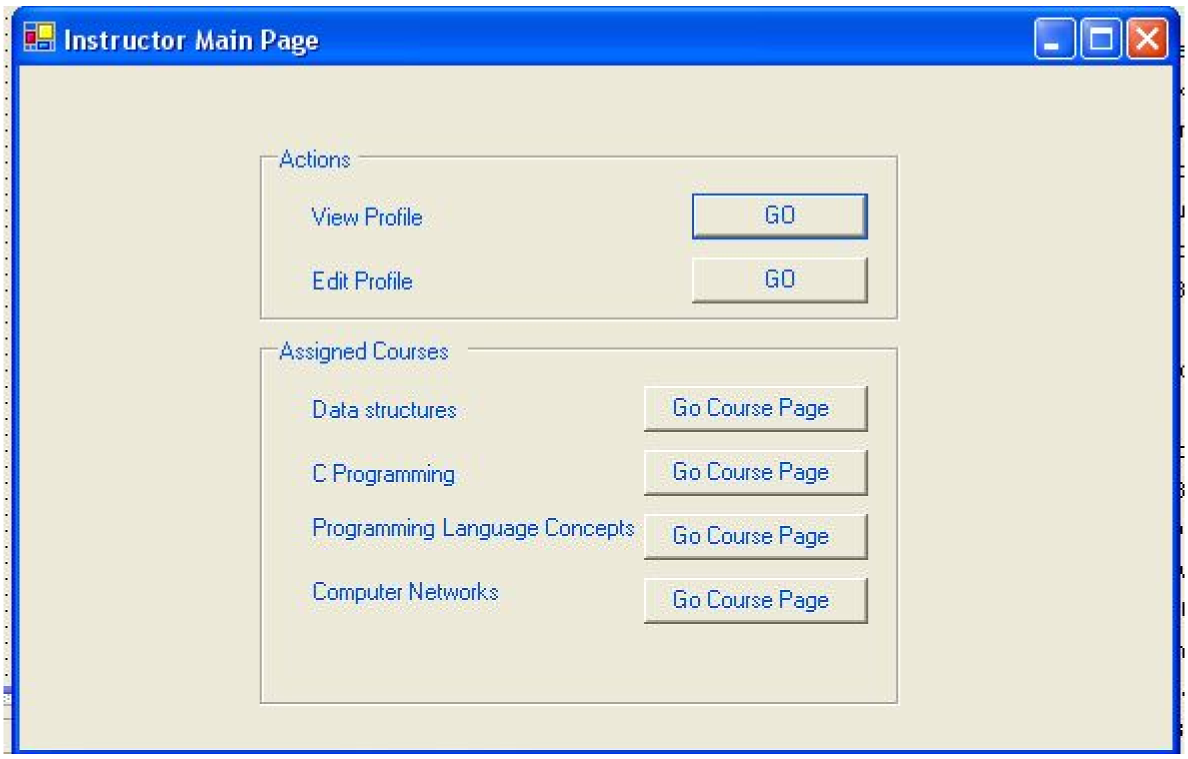


Operating systems	Online	Go Course Page
Microprocessors	Ofline	Go Course Page
Biology	Ofline	Go Course Page
Organic chemistry	Ofline	Go Course Page

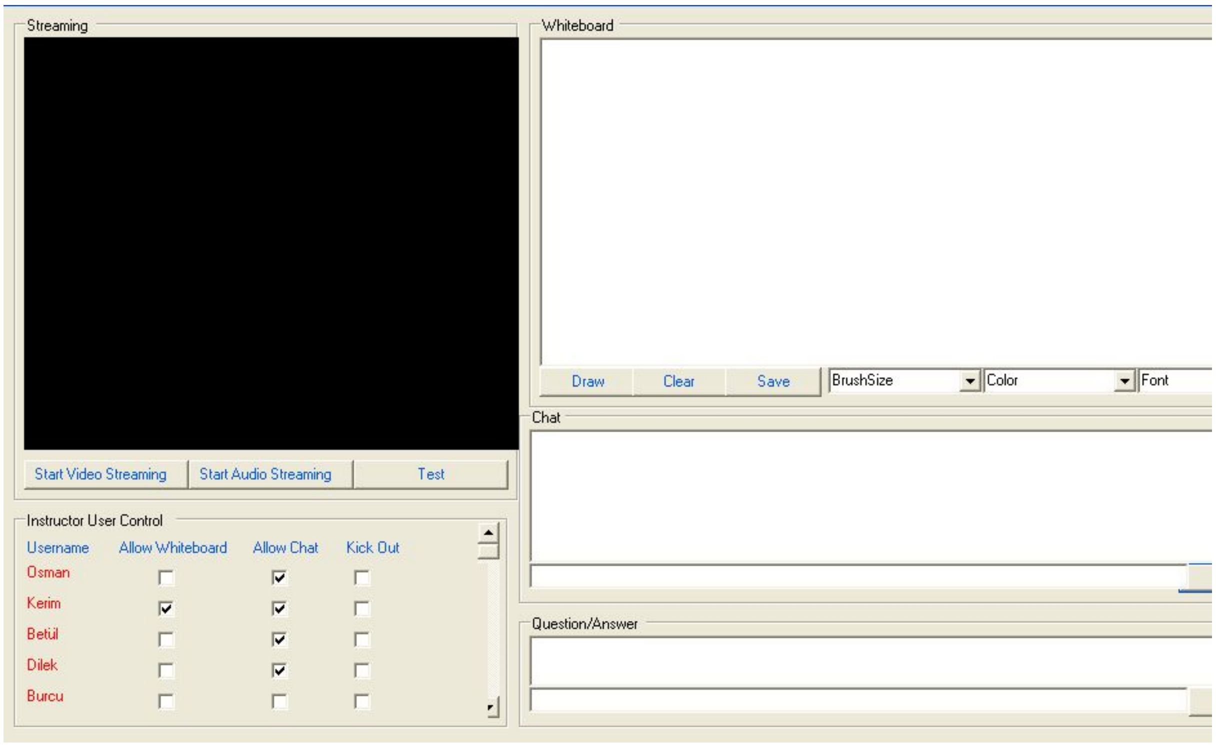
4.3 Admin Main Page



4.4 Instructor Main Page



4.5 Virtual Classroom



Username	Allow Whiteboard	Allow Chat	Kick Out
Osman	☐	☑	☐
Kerim	☑	☑	☐
Betül	☐	☑	☐
Dilek	☐	☑	☐
Burcu	☐	☐	☐

5. CONCLUSION

This initial design report is prepared to establish a connection between our design and implementation. The information given here as diagrams, flow charts and other design products are produced in order to guide us through our way in the implementation of our Virtual Classroom. Despite being an initial design, this document is a milestone that will help us make our prototype and real design report. We believe that this report will contribute to our project in a quite useful way.

APPENDIX

Time chart is at the following page.