**PROJECT REPORT**

(SEMESTER TRAINING)

**“Videoconferencing Application Development**

**Along With Interactive Whiteboard”**

Submitted By

**Simrandeep Singh**

**Roll No. :- 100903105**

Under the Guidance of

**Mr. Sharad Jain**

**(IT-Engineer)**

**(AT&T Communication Services India Pvt. Ltd.)**

To



**Department Of Computer Science and Engineering**

**THAPAR UNIVERITY, PATIALA.**

(Deemed University)

For Period

**Jan – May/Jun 2012**

**DECLARATION**

I hereby declare that the project work entitled “Videoconferencing Application Development

Along With Interactive Whiteboard” is an authentic work of my own work carried out at AT&T

Communication Services India Pvt. Ltd. , Gurgaon (Haryana) as per requirements of semester

project term for the award of degree of B.E. (Computer Science & Engineering), Thapar University,

Patiala, under the guidance of **Mr. Sharad Jain** during period 9th Jan to 1st Jun, 2012.

Date: / / .

Simrandeep Singh

(100903105)

Mr. Sharad Jain

(IT-Engineer)

AT&T Communication Services India Pvt. Ltd.

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| S No. | Particulars | Page No. |
| 1. | Acknowledgement | 4 |
| 2. | Introduction to Organization | 5 |
| 3. | Summary of the Project | 8 |
| 4. | Introduction to Programming and Developing Environment | 10 |
| 5. | Project | 28 |
| 6. | Data Flow Diagrams | 43 |
| 7. | Details of the work | 49 |
| 8. | Testing – Different test cases | 51 |
| 9. | Result, Conclusion and Future Scope | 56 |
| 10. | References | 58 |

**ACKNOWLEDGEMENT**

It is my pleasure to acknowledge the help, that I had received from different individuals and AT&T Communication Services India Pvt. Ltd. During the Project Semester.

I take this opportunity to express my gratitude and thanks to **Mr. Sharad Jain**, Mentor, for his guidance, constructive comments, valuable suggestions and inspirations. During the entire Project Semester, I have received Endless help from him.

It is my pleasure to express my wholehearted sense of gratitude to **Mr. Peter Baker**, IT- Department Head, for his invaluable and untiring guidance, moral support, supervision throughout the project semester and for giving an opportunity to work at the company.

I would also like to express my thanks to my teachers **Mr. Vinay Arora** and **Dr. Anil Kumar Verma**, IAP Co-ordinators, and **Ms. Rupinderdeep Kaur**, Visiting Teacher, for helping me throughout the training period and providing assistance whenever and wherever need arisen.

Finally, I wish to say thanks to all people of the company for their kind co- operation.

1. **Introduction to Organization**



**1.1 About at&t**

AT&T is an American multinational telecommunications corporation headquartered in Whitacre Tower, Dallas, Texas, United States.  It is the second largest provider of mobile telephony and largest fixed telephony provider in the United States, and is also a provider of broadband subscription television services. As of 2010, AT&T is the 7th largest company in the United States by total revenue, as well as the 4th largest non-oil company in the US. In 2011, Forbes listed AT&T as the 14th largest company in the world by market value and the 9th largest non-oil company in the world by market value. It is the 20th largest mobile telecom operator in the world with over 100.7 million mobile customers.

The company began its existence as Southwestern Bell Corporation, one of seven Regional Bell Operating Companies created in 1983 as part of the break-up of the original AT&T due to the United States v. AT&T antitrust lawsuit. In 2005, it purchased its former parent company, AT&T Corp. and took on its branding, with the merged entity naming itself AT&T Inc*.* and using the iconic AT&T logo and stock-trading symbol.

The current AT&T reconstitutes much of the former Bell System and includes ten of the original 22 Bell Operating Companies, along with one it partially owned, and the original long distance division. The company is headquartered in downtown Dallas, Texas.

AT&T Communications, originally AT&T Communications Inc., is a holding company for the 23 subsidiaries that provide interexchange carrier and long distance telephone services owned by AT&T.

**1.2 Services Offered**

AT&T offers significant advanced networking services and application solutions, including data communication, voice communication (VOIP), unified communications, security, hosting and application services to businesses across the globe.

Main fields in which AT&T offers services are :-

* Access

AT&T Access portfolio provides a wealth of choices to match your business environment, from Rings to WANs. AT&T Ethernet solutions give you simple, scalable, economical options for meeting your increasing bandwidth demands with metro, wide-area and global Ethernet services.

* Data

AT&T Data Services portfolio ranges from basic connectivity to global networking, including self-management or complete management options.

* Enterprise Mobility

AT&T Enterprise Mobility deals in deploying mobile solutions. It provides secure, remote access and fixed-mobile solutions.

* Hosting and Application Services

AT&T Hosting and Application Service portfolio provides Web Hosting, Synaptic Hosting Service, Remote Infrastructure Management, Enterprise Hosting Services and Application Management Services.

* Integrated Solutions

AT&T Integrated Solutions portfolio deals with assessment, design, integration of networks, Staff augmentation, Wireless application solutions, VOIP integration etc.

* Security

AT&T Security portfolio deals with e-mail and web security, firewall and client security, Threat Management.

* Unified Communication

AT&T Unified Communication portfolio deals with Voice, Web and Video Conferencing Services. Also it offers e-mail and voice messaging solutions.

* Voice

AT&T Voice portfolio deals with Voice over IP(VoIP) Services and IP Telephony Services.

**1.3 About AT&T in India**

AT&T has a significant presence in the Asia Pacific and India regions, with offices in 14

markets. The company offers advanced networking services and application solutions, including

data, voice, unified communications, security, hosting and application services to businesses across

Asia Pacific region for over 30 years.

For more than a century, AT&T has been consistently providing innovative, reliable, high-

quality products and services and excellent customer care. Today, its mission is to connect people

with their world, everywhere they live and work, and do it better than anyone else. AT&T is

fulfilling this vision by creating new solutions for businesses and by driving innovation in the

communications industry.

AT&T is recognized as one of the leading worldwide providers of IP-based communications

services to businesses. It also have the nation's fastest 3G network and the largest international

coverage of any U.S. wireless carrier, offering the most phones that work in the most countries; the

largest Wi-Fi network in the United States; and the largest number of high speed Internet access

subscribers in the United States.

AT&T is extending the power of our network by building in not only sheer speed, but also the

intelligence to enable and support a full array of solutions. New technologies will enable more

efficient delivery of more content and applications.

AT&T is also adding intelligence to empower people to do more with their connectivity.

We're delivering synaptic hosting, which allows businesses to tap intohosting, applications or

computing power on demand. These innovations combine to create an applications-centric network

that will power the future of communications.

**2. Summary of Project**

I worked in the IT-Networking Department at AT&T Communication Services India Pvt. Ltd.

The IT team’s primary objective is to support the internal servers and to deal with the internal

networking and communication objectives.

The project assigned was related to data communications which deals with communication

using data transfer of type Video, Voice, Files, And Other Information like graphs.

Videoconferencing Application Development with Interactive Whiteboard is an application

that allows the registered users of the application to have Videoconference with the use of JM Studio

and Java Media Framework API. Registered User can conference with 6 users simultaneously.

Quality of Service depends on the bandwidth provided to the user.

Registered Users can communicate graphical information with each other using Whiteboard.

Also this application provides the users with the facility to send files from one computer to another

and also with peer to peer text chat messenger.

It is developed using mysql as database and java technologies as coding environment.

Different Aspects related to project are:

1. DataBase - mysql 5.5

Database designing of the project. So that only authenticated users can login. And it serves as

the central server for all users.

2. Video Conferencing

A window based Java Application which helps to connect the employees on the same internal

network. It deals with the transfer of Voice and Video and Synchronization of both.

2.1 User Maintenance

2.2 Audio and Video Sending

2.3 Audio and Video Receiving

2.4 Merging Audio and Video

2.5 Cloning of Media Player

3. File Transfer

It deals with the file transfer from one PC to another PC. The approach is client- server

approach.

4. Whiteboard

Whiteboard is a window based GUI using Java. Basically to be integrated with the Video

Conferencing module. The idea was to transfer graphical information to the user that was difficult

using Video Conferencing.

4.1 Basic user interface of the Whiteboard

4.2 Developing the model of the Whiteboard that describes how it will work

4.3 Different Networking Connection required to handle the Whiteboard Session.

5. P2P Text Transfer

A window based GUI which deals with one to one P2P text transfer. The Idea was to design

the GUI for one to one person. Because in whiteboard session the information or text was for all the

users of the session.

**3. Introduction to Programming and Developing Environment**

# 3.1 Introduction to MYSQL

MySQL, the most popular Open Source SQL database management system, is developed, distributed, and supported by Oracle Corporation. The MySQL (R) software delivers a very fast, multi-threaded, multi-user, and robust SQL (Structured Query Language) database server. MySQL Server is intended for mission-critical, heavy-load production systems as well as for embedding into mass-deployed software.

The MySQL software is Dual Licensed. Users can choose to use the MySQL software as an Open Source/Free Software product under the terms of the GNU General Public License or can purchase a standard commercial license.

The MySQL web site (http://www.mysql.com/) provides the latest information about the MySQL software.

* MySQL is a relational database management system.

A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server. Since computers are very good at handling large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications.

* MySQL software is Open Source.

A relational database stores data in separate tables rather than putting all the data in one big storeroom. This adds speed and flexibility. The SQL part of “MySQL” stands for “Structured Query Language.” SQL is the most common standardized language used to access databases and is defined by the ANSI/ISO SQL Standard. The SQL standard has been evolving since 1986 and several versions exist.

* The MySQL Database Server is very fast, reliable, and easy to use.

MySQL Server was originally developed to handle large databases much faster than existing solutions and has been successfully used in highly demanding production environments for several years. Although under constant development, MySQL Server today offers a rich and useful set of functions. Its connectivity, speed, and security make MySQL Server highly suited for accessing databases on the Internet

* MySQL Server works in client/server or embedded systems.

The MySQL Database Software is a client/server system that consists of a multi-threaded SQL server that supports different backends, several different client programs and libraries, administrative tools, and a wide range of application programming interfaces (APIs).

MySQL Server also provides an embedded multi-threaded library that you can link into your application to get a smaller, faster, easier-to-manage standalone product.

* A large amount of contributed MySQL software is available.

### [The Main Features of MySQL](file:///F:\thapar\computer%20fundamentals\java\java_se\Projects\emailPLUS\Projects%202010\mysql\Docs\manual_toc.html#Features)

* Internals and Portability
* Column Types
* Commands and Functions
* Security
* Scalability and Limits
* Connectivity
* Localization

**3.2 Introduction to JAVA**

**­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­**

Java is an object-oriented programming language developed by James Gosling at Sun Microsystems (now Oracle) and released in 1995 as a core component of Sun Microsystems Java Platform with a built-in application programming interface (API) that can handle graphics and user interfaces and that can be used to create applications or applets. Because of its rich set of API's, similar to Macintosh and Windows, and its platform independence, Java can also be thought of as a platform in itself. Java also has standard libraries for doing mathematics.

Java applications are typically compiled to bytecode (class file) that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is a general-purpose, concurrent, class-based, object-oriented language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another.

Much of the syntax of Java is the same as C and C++. One major difference is that Java does not have pointers. However, the biggest difference is that you must write object oriented code in Java. Procedural pieces of code can only be embedded in objects. In the following we assume that the reader has some familiarity with a programming language. In particular, some familiarity with the syntax of C/C++ is useful.

In Java we distinguish between applications, which are programs that perform the same functions as those written in other programming languages, and applets, which are programs that can be embedded in a Web page and accessed over the Internet. Our initial focus will be on writing applications. When a program is compiled, a byte code is produced that can be read and executed by any platform that can run Java.

C++

C

Java

**Java Platform**

One characteristics of Java is portability, which means that computer programs written in the

java language must run similarly on any supported hardware/operating-system platform. This is

achieved by compiled the Java language code to an intermediate representation called java bytecode,

instead of directly to platform-specific machine code. Java bytecode instructions are analogous to

machine code, but are intended to be interpreted by a virtual machine written specifically for the host

hardware. End users commonly use a Java Runtime Environment (JRE) installed on their own

machine for standalone java applications, or in a Web browser for Java applets.

Standardized libraries provide a generic way to access host-specific features such as graphics,

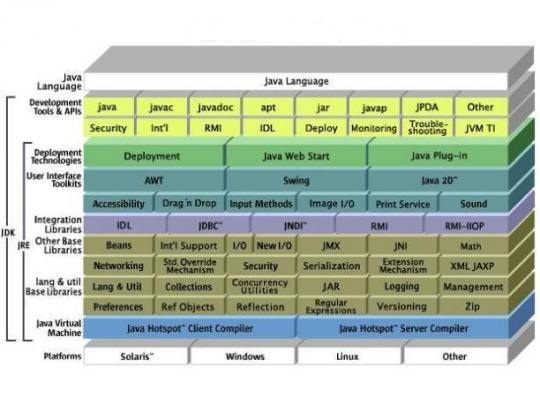
threading,and networking.

A major benefit of using bytecode is porting. However, the overhead of interpretation means

that interpreted programs almost always run more slowly than programs compiled to native

executables would. Just-In-Time compilers were introduced from an early stage that compiles

bytecodes to machine code during runtime.



**Java Platform**

A java platform is the hardware or software environment in which a program runs. The java platform has two components:

* Java Virtual Machine (JVM)
* Java Application Programming Interface (Java API)

JVM is a standardized hypothetical computer, which is emulated inside your computer by a program.

Java Object Code

Java Compiler

Java Source Code

The **Java API** is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI). The java API is grouped into libraries of related classes and interfaces these libraries are known as packages.



**Java Virtual Machine**

A **Java virtual machine** (**JVM**) is a virtual machine that can execute Java bytecode. It is the

code execution component of the Java software platform. The model used by a JVM accepts a form

of computer intermediate language commonly referred as Java Bytecode. Sun Microsystems stated

that there are over 5.5 billion JVM-enabled devices.

A Java virtual machine is software that is implemented on virtual and non-virtual hardware

and on standard operating systems. A JVM provides an environment in which Java bytecode can be

executed, enabling such features as automated exception handling, which provides *root-cause*

debugging information for every software error (exception), independent of the source code. A JVM

is distributed along with a set of standard class libraries that implement the Java application

programming interface (API). Appropriate APIs bundled together with JVM form the Java Runtime

Environment (JRE).

JVMs are available for many hardware and software platforms. The use of the same bytecode

for all JVMs on all platforms allows Java to be described as a *write once, run anywhere*

programming language, versus *write once, compile anywhere*, which describes cross-platform

compiled languages. Thus, the JVM is a crucial component of the Java platform.

Java bytecode is an intermediate language which is typically compiled from Java, but it can also be

compiled from other programming languages. For example, Ada source code can be compiled to

Java bytecode and executed on a JVM.

Oracle, the owner of Java, produces a JVM, but JVMs using the *Java* trademark may be

developed by other companies as long as they adhere to the JVM specification published by Oracle

and to related contractual obligations.

A JVM is distributed along with a set of standard class libraries that implement the Java application programming interface (API). Appropriate API’s bundled together form the Java Runtime Environment (JRE).

**Java Runtime Environment**

Oracle's Java execution environment is termed the Java Runtime Environment, or JRE.

Programs intended to run on a JVM must be compiled into a standardized portable binary format, which typically comes in the form of .class files. A program may consist of many classes in different files. For easier distribution of large programs, multiple class files may be packaged together in a .jar file (short for Java archive).

The Java application launcher, java, offers a standard way of executing Java code.

The JVM runtime executes .class or .jar files, emulating the JVM instruction set by interpreting it, or using a just-in-time compiler (JIT) such as Oracle's HotSpot. JIT compiling, not interpreting, is used in most JVMs today to achieve greater speed. There are also ahead-of-time compilers that enable developers to precompile class files into native code for particular platforms.

Like most virtual machines, the Java virtual machine has a stack-based architecture akin to a microcontroller/microprocessor. However, the JVM also has low-level support for Java-like classes and methods, which amounts to a highly idiosyncratic memory model and capability-based architecture.

**Bytecode Verifier**

A basic philosophy of Java is that it is inherently safe from the standpoint that no user program can crash the host machine or otherwise interfere inappropriately with other operations on the host machine, and that it is possible to protect certain methods and data structures belonging to trusted code from access or corruption by untrusted code executing within the same JVM. Furthermore, common programmer errors that often lead to data corruption or unpredictable behavior such as accessing off the end of an array or using an uninitialized pointer are not allowed to occur. Several features of Java combine to provide this safety, including the class model, the garbage- collected heap, and the verifier.

The JVM verifies all bytecode before it is executed. This verification consists primarily of three types of checks:

* Branches are always to valid locations
* Data is always initialized and references are always type-safe
* Access to private or package private data and methods is rigidly controlled.

The first two of these checks take place primarily during the verification step that occurs when a class is loaded and made eligible for use. The third is primarily performed dynamically, when data items or methods of a class are first accessed by another class.

The verifier permits only some bytecode sequences in valid programs, e.g. a jump (branch) instruction can only target an instruction within the same method. Furthermore, the verifier ensures that any given instruction operates on a fixed stack location, allowing the JIT compiler to transform stack accesses into fixed register accesses. Because of this, that the JVM is a stack architecture does not imply a speed penalty for emulation on register-based architectures when using a JIT compiler. In the face of the code-verified JVM architecture, it makes no difference to a JIT compiler whether it gets named imaginary registers or imaginary stack positions that must be allocated to the target architecture's registers. In fact, code verification makes the JVM different from a classic stack architecture which efficient emulation with a JIT compiler is more complicated and typically carried out by a slower interpreter.

Code verification also ensures that arbitrary bit patterns cannot get used as an address. Memory protection is achieved without the need for a memory management unit (MMU). Thus, JVM is an efficient way to get memory protection on simple architectures that lack an MMU. This is analogous to managed code in Microsoft's .NET Common Language Runtime, and conceptually similar to capability architectures such as the Plessey 250, and IBM System/38.

The original specification for the bytecode verifier used natural language that was incomplete or incorrect in some respects*.* A number of attempts have been made to specify the JVM as a formal system. By doing this, the security of current JVM implementations can more thoroughly be analyzed, and potential security exploits prevented. It will also be possible to optimize the JVM by skipping unnecessary safety checks, if the application being run is proved to be safe.

**Automatic Memory Management**

Java uses an automatic garbage collector to manage memory in the object lifecycle. The programmer determines when objects are created, and the Java runtime is responsible for recovering the memory once objects are no longer in use. Once no references to an object remain, the unreachable memory becomes eligible to be freed automatically by the garbage collector. Something similar to a memory leak may still occur if a programmer's code holds a reference to an object that is no longer needed, typically when objects that are no longer needed are stored in containers that are still in use. If methods for a nonexistent object are called, a "null pointer exception" is thrown.

One of the ideas behind Java's automatic memory management model is that programmers can be spard the burden of having to perform manual memory management. In some languages, memory for the creation of objects is implicitly allocated on the stack, or explicitly allocated and deallocated from the heap. In the latter case the responsibility of managing memory resides with the programmer. If the program does not deallocate an object, a memory leak occurs. If the program attempts to access or deallocate memory that has already been deallocated, the result is undefined and difficult to predict, and the program is likely to become unstable and/or crash. This can be partially remedied by the use of smart pointers, but these add overhead and complexity. Note that garbage collection does not prevent "logical" memory leaks, i.e. those where the memory is still referenced but never used.

Garbage collection may happen at any time. Ideally, it will occur when a program is idle. It is guaranteed to be triggered if there is insufficient free memory on the heap to allocate a new object; this can cause a program to stall momentarily. Explicit memory management is not possible in Java.

Java does not support C/C++ style pointer arithmetic, where object addresses and unsigned integers (usually long integers) can be used interchangeably. This allows the garbage collector to relocate referenced objects and ensures type safety and security.

As in C++ and some other object-oriented languages, variables of Java's primitive data types are not objects. Values of primitive types are either stored directly in fields (for objects) or on the stack (for methods) rather than on the heap, as commonly true for objects (but see Escape analysis). This was a conscious decision by Java's designers for performance reasons. Because of this, Java was not considered to be a pure object-oriented programming language. However, as of Java 5.0, autoboxing enables programmers to proceed as if primitive types were instances of their wrapper class.

Java contains multiple types of garbage collectors. By default, HotSpot uses the Concurrent Mark Sweep collector, also known as the CMS Garbage Collector. However, there are also several other garbage collectors that can be used to manage the Heap. For 90% of applications in Java, the CMS Garbage Collector is good enough.

**3.3 Introduction to JAVA API – Java Media Framework (JMF)**

JMF is a framework for handling streaming media in Java programs. JMF is an optional package of Java 2 standard platform. JMF provides a unified architecture and messaging protocol for managing the acquisition, processing and delivery of time-based media. JMF enables Java programs to :-

(i) Present ( playback) multimedia contents,

(ii) capture audio through microphone and video through Camera,

(iii) do real-time streaming of media over the Internet,

(iv) process media ( such as changing media format, adding special effects),

(v) store media into a file.

JMF supports many popular media formats such as JPEG, MPEG-1, MPEG-2, QuickTime, AVI, WAV, MP3, GSM, G723, H263, and MIDI. JMF supports popular media access protocols such as file, HTTP, HTTPS, FTP, RTP, and RTSP.

JMF uses a well-defined event reporting mechanism that follows the “Observer” design pattern. JMF uses the “Factory” design pattern that simplifies the creation of JMF objects. The JMF support the reception and transmission of media streams using Real-time Transport Protocol (RTP) and JMF supports management of RTP sessions.

JMF scales across different media data types, protocols and delivery mechanisms. JMF provides a plug-in architecture that allows JMF to be customized and extended. Technology providers can extend JMF to support additional media formats. High performance custom implementation of media players, or codecs possibly using hardware accelerators can be defined and integrated with the JMF.

Multimedia processing and presentation is compute-intensive. Therefore most of the existing media players and processors for desktop computers are implemented using native code for performing computationally intensive tasks like media encoding, decoding, and rendering.

The general criticism on Java-based applications and therefore on JMF is that they lack performance as compared to native codes. The answer to this criticism is as follows.

The main drawback of native implementations of media players is that they are platform dependent. Hence they are not portable across platforms. This directly means applications using platform-dependent media players and processors are unsuitable for web-deployment. JMF provides a platform-neutral framework for handling multimedia.

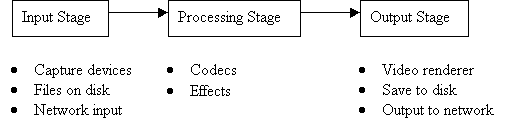
The JMF API provides an abstraction that hides these implementation details from the developer. For example, a particular JMF Player implementation might choose to leverage an operating system's capabilities by using native methods. Indeed Sun’s implementation of JMF has different versions each one tailored for one platform.

**The JMF model**

JMF adopts the same model that is used by the consumer electronics industry in handling the media. According to the JMF model, the life cycle of the media starts from a media source, and ends in a media sink. In between the media is handled by media handlers.

The media source can be a (i) a capture device, or (ii) a media file stored locally or remotely on the network or (iii) a real-time media stream available on the network. The media handlers process the media which may involve demultiplexing or multiplexing or encoding or decoding. The media processing can be implemented partly in hardware but mostly it is done by software. The media sink or destination can be rendering devices, or storage files or media streams.

For audio the capture device is a microphone along with a sound card. For images and video the capture device is a PC add-on digital camera. We typically use CRT monitor for rendering images or video and speakers for rendering audio.



**JMF Architecture**

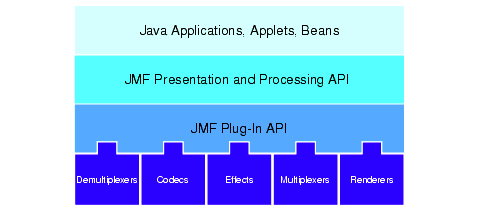
Devices such as tape decks and VCRs provide a familiar model for recording, processing, and presenting time-based media. When you play a movie using a VCR, you provide the media stream to the VCR by inserting a video tape. The VCR reads and interprets the data on the tape and sends

appropriate signals to your television and speakers.

JMF uses this same basic model. A data sourceencapsulates the media stream much like a video tape and a player provides processing and control mechanisms similar to a VCR. Playing and capturing audio and video with JMF requires the appropriate input and output devices such as microphones, cameras, speakers, and monitors. Data sources and players are integral parts of JMF’s high-level API for managing the capture, presentation, and processing of time-based media.

JMF also provides a lower-level API that supports the seamless integration of custom processing components and extensions. This layering provides Java developers with an easy-to-use API for incorporating timebased media into Java programs while maintaining the flexibility and

extensibility required to support advanced media applications and future media technologies.



High-Level JMF Architecture

**JMF Input**

In JMF, an input is generally represented by a MediaLocator object. As stated previously, the MediaLocator looks and acts much like a URL, in that it uniquelyidentifies a resource in a network. In fact, it is possible to create a MediaLocator using a URL; we did this in our two previous example applications. For the purpose of our media conversion example, we could build a MediaLocator to describe the original WAV file. As we will see in the next few sections, a MediaLocator may also be used to represent a media stream being broadcast across a network. In this case, instead of building the MediaLocator to identify a file on the local file system, the MediaLocator would describe the URL of the broadcast -- much like a resource on the Web is identified by its URL.

**JMF processors**

When we are working with JMF, the processor component of the application is represented by an instance of the Processor interface. You should already be somewhat familiar with the Processor, as it is an extension of the Player interface. Because the Processor inherits from the Player interface, it also inherits all of the valid states from the Player. In addition, the Processor adds two

more states: Configuring and Configured. These extra states (and associated events) are used to communicate when the Processor is gathering information from the input stream.

For our final example application, we will create a Processor to convert audio encoded in the MP3 format into a format suitable for broadcast over a network. We will discuss the steps to create a simple Processor in a later panel.

**JMF output**

There are a few ways to represent the output phase of the JMF process model. The simplest (and the one we will use in the final example) is the javax.media.DataSink interface. A DataSink reads media content and renders it to some destination. In the audio-format conversion scenario at the beginning of this section, the MP3 (output) file would be represented by the DataSink. In our

final example, we will use a DataSink to actually do the work of broadcasting audio media over a network. A DataSink is created through the Manager class by specifying a DataSource (the input to the DataSink) and a MediaLocator (the output of the DataSink).

A DataSource instance represents input data, which is used in Players, Processors, and DataSinks. The output of a Processor is also represented as a DataSource object. This is how Processors can be chained together to perform multiple operations on the same media data. This is also how the output from a Processor can be used as input to a Player or to a DataSink (which would render the media to an output destination).



**3.4 Introduction to JMF RTP API**

RTP provides end-to-end network delivery services for the transmission of real-time data.JMF supports a protocol called Real-time Transport Protocol (RTP) for media transmission and reception. RTP is a transport layer protocol and is typically used above the UDP layer. RTP packets has (i) time stamping to indicate the time instant at which the media carried by the packet has to be played, (ii) sequence number that can be used for the ordered delivery of the packets, (iii) identification of the media source and (iv) payload media format identification.

RTP can be used over both unicast and multicast network services. Over a unicastnetwork service, separate copies of the data are sent from the source to each destination. Over a multicastnetwork service, the data is sent from the source only once and the network is responsible for transmitting the data to multiple locations. Multicasting is more efficient for many multimedia applications, such as video conferences. The standard Internet Protocol (IP) supports multicasting.

A RTP session consists of a set of applications exchanging media using the RTP. Each of these applications is called a participant. Every participant uses an object called RTPManager to co- ordinate the RTP session on its behalf. Media streams exchanged in a RTP session are called RTPStreams. The RTPStreams can be of two types, sendStream and ReceiveStream.

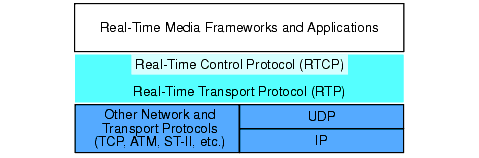
The JMF RTP API allows us to construct RTPManagers using which we can send and receive RTPStreams among the participating JMF applications.

**RTP Architecture**

A RTP sessionis an association among a set of applications communicating with RTP. A session is identified by a network address and a pair of ports. One port is used for the media data and the other is used for control (RTCP) data. A participantis a single machine, host, or user participating in the session. Participation in a session can consist of passive reception of data

(receiver), active transmission of data (sender), or both. Each media type is transmitted in a different session. For example, if both audio and video are used in a conference, one session is used to transmit the audio data and a separate session is used to transmit the video data.

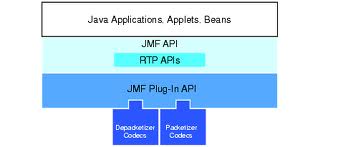
This enables participants to choose which media types they want to receive for example, someone who has a low-bandwidth network connection might only want to receive the audio portion of a conference.



RTP Architecture

The JMF RTP APIs are designed to work seamlessly with the capture, presentation,

and processing capabilities of JMF. Players and processors are used to present and manipulate RTP media streams just like any other media content. You can transmit media streams that have been captured from a local capture device using a capture DataSource or that have been stored to a file using a DataSink. Similarly, JMF can be extended to support additional RTP formats and payloads through the standard plug-in mechanism.



High Level JMF RTP Architecture

RTP applications are often divided into those that need to be able to receive data from the network (RTP Clients) and those that need to be able to transmit data across the network (RTP Servers). Some applications do both for example, conferencing applications capture and transmit data at the same time that they are receiving data from the network.

**3.5 Introduction to Java 2D API**

The Java 2D™ API enhances the graphics, text, and imaging capabilities of the Abstract Windowing Toolkit (AWT), enabling the development of richer user interfaces and new types of Java applications. Along with these richer graphics, font, and image APIs, the Java 2D API supports enhanced color definition and composition, hit detection on arbitrary geometric shapes and text, and a uniform rendering model for printers and display devices.

The Java 2D API also enables the creation of advanced graphics libraries, such as CAD-CAM libraries and graphics or imaging special effects libraries, as well as the creation of image and graphic file read/write filters.

When used in conjunction with the Java Media Framework and other Java Media APIs, the Java 2D APIs can be used to create and display animations and other multimedia presentations. The Java Animation and Java Media Framework APIs rely on the Java 2D API for rendering support.

## Basic Concepts

These objects are a necessary part of every Java 2D drawing operation.

### Shapes

A shape in Java 2D is a boundary which defines an inside and an outside. Pixels inside the shape are affected by the drawing operation, those outside are not. Trying to fill a straight line segment will result in no pixels being affected, as such a shape does not contain any pixels itself. Instead, a thin rectangle must be used so that the shape contains some pixels.

### Paints

A paint generates the colors to be used for each pixel of the fill operation. The simplest paint is **java.awt.Color**, which generates the same color for all pixels. More complicated paints may produce gradients, images, or indeed any combination of colors. Filling a circular shape using the color yellow results in a solid yellow circle, while filling the same circular shape using a paint that generates an image produces a circular cutout of the image.

### Composites

During any drawing operation, there is a source (the pixels being produced by the paint) and a destination (the pixels already onscreen). Normally, the source pixels simply overwrite the destination pixels, but the composite allows this behaviour to be changed.

The composite, given the source and destination pixels, produces the final result that ultimately ends up onscreen. The most common composite is **java.awt.AlphaComposite**, which can treat the pixels being drawn as partially transparent, so that the destination pixels show through to some degree.

### Filling

To fill a shape, the first step is to identify which pixels fall inside the shape. These pixels will be affected by the fill operation. Pixels that are partially inside and partially outside the shape may be affected to a lesser degree if anti-aliasing is enabled.

The paint is then asked to generate a color for each of the pixels to be painted. In the common case of a solid-color fill, each pixel will be set to the same color.

The composite takes the pixels generated by the paint and combines them with the pixels already onscreen to produce the final result.

## Advanced Objects

These objects can be viewed as performing their duties in terms of the simpler objects described above.

### Transform

Every Java 2D operation is subject to a transform, so that shapes may be translated, rotated, sheared, and scaled as they are drawn. The active transform is most often the identity transform, which does nothing.

Filling using a transform can be viewed as simply creating a new, transformed shape and then filling that shape.

### Stroke

In addition to the fill operation, Java 2D provides a draw operation. While fill draws the interior of a shape, draw draws its outline. The outline can be as simple as a thin line, or as complicated as a dashed line with each dash having rounded edges.

The object responsible for generating the outline is the stroke. Given an input shape, the stroke produces a new shape representing its outline. For instance, an infinitely thin line segment (with no interior) might be stroked into a one-pixel-wide rectangle.

A draw operation can therefore be described as creating a new, stroked object and then filling that object.

Technically speaking, the stroke is only required to accept an input shape and produce a new shape. The stroke implementation provided with Java 2D implements the outline rules described above, but a custom-written stroke could produce any shape it wished.

**Java 2D API Packages**

The Java 2D API and its documentation are available for download as a part of JDK 6. Java 2D API classes are organised into the following packages in JDK 6:

* *java.awt* The main package for the Java Abstract Window Toolkit.
* *java.awt.geom* The Java standard library of two dimensional geometric shapes such as lines, ellipses, and quadrilaterals.
* *java.awt.font* The library for manipulating glyphs in Java.
* *java.awt.color* The library dealing with the many different ways that color can be represented.
* *java.awt.image* The library for manipulating graphical images.
* *java.awt.print* The library of tools for writing to paper.

**4. Project - ‘Videoconferencing Application Development**

**Along With Interactive Whiteboard’**

**Introduction**

Videoconferencing Application Development Along With Interactive Whiteboard is a data communication project whose main purpose is to provide maximum interactivity so that registered users can communicate with each other by transfer of Voice, Video, Graph, File, Text data. It is a window based application that allows the registered users of the application to have Videoconference with the use of JM Studio and Java Media Framework API. Registered User can conference with 6 users simultaneously. Quality of Service depends on the bandwidth provided to the user.

Registered Users can communicate graphical information like pie chart, bar graph, line graph etc. with each other using Whiteboard, an application developed using java 2D API. The session can be created private i.e. password protected, so that only required users can join the session. The session can also be created as read-only, so that the person who created the session can make changes and other users are only spectators. Also this application provides the users with the facility to send files from one computer to another and also with peer to peer text chat messenger.

At the back end mysql is used as central database for the authentication of registered users. And it also helps the users by storing the dynamic IP of other users with whom the user wants to connect. Also the user’s personal profile information is stored in the database.

Different aspects/modules related to the project with brief description are as follow:

**1.** **Videoconferencing**

A window based Java Application which helps to connect the employees on the same internal

network. It deals with the transfer of Voice and Video and Synchronization of both. A single user

can communicate with 6 users simultaneously at maximum. The quality of the video and voice also

depend on bandwidth provided to a particular user.

The various modules of the videoconferencing part are as follows

**1.1 User Maintenance**

User Maintenance module deals with the basic information regarding user’s login, user’s registration and the details about his/her personal profile. And stores the user’s information in the central database. Required information for the module like username, password are fetched from central database during login.

First of all user has to sign-up for the application by choosing his/her username, password and an email-id with Gmail account required in case if user forgot his/her login details. And only users registered with the application can login. User maintenance also deals with the user’s personal profile details such as name, DOB, address, mobile no., email-id etc. The application also has the facility for the user to change his password.

In case user forgets his password user can retrieve his password by providing his username. An email will be sent to the user at his/her provided email-id by the administrator of the application. In case if the user has any type of query related to any aspect of the application user can send a query to the admin’s email account by providing his/her password of the email account with which user have registered with the application. Admin’s reply will be sent back to the user’s email account.

**1.2 Audio and Video Sending**

For sending the audio and video to the other user first of all it is required to capture the audio and video devices that will be used for the application. These devices should be registered with Java Media Registry ( JMRegistry ). For example the video device in windows is represented as “vfw://0” and two audio devices in windows is represented by two different media locator string i.e. “dsound://” or “javasound://44100”. Different source media locator can also be created for sending files from the PC or files by giving the URL of the file for eg. “file:clips/clip01mov.mov” for file and “http://www.cs.odu.edu/~cs778/clip01mpg.mpg” for http URL locator. But for real time media capturing for video and audio we use datasource such as “vfw://0” for video and “dsound://” or “javasound://44100” for audio.

For sending the Voice and Video data we require the IP-address of the user to whom we will send the data. And two different port number of the user at which user is listening for audio and video differently. Also the important part is that both the ports should be of even number.

The source video, whether it is a file or live video, needs to be in a format that can be converted to JPEG/RTP or H263/RTP. H263, Cinepak, RGB, YUV and JPEG are good formats. Also the dimensions of the video should be a multiple of 8x8.

A diagram that represents how sending of audio/video data is done is given below.



RTP Transmission

A session manager can also be used to initialize and control a session so that you can stream data across the network. The data to be streamed is acquired from a Processor. For example, to create a send stream to transmit data from a live capture source, you would:

1. Create, initialize, and start a SessionManager for the session.

2. Construct a Processor using the appropriate capture DataSource.

3. Set the output format of the Processor to an RTP-specific format. An appropriate RTP packetizer codec must be available for the data format you want to transmit.

4. Retrieve the output DataSource from the Processor.

5. Call createSendStream on the session manager and pass in the Data-Source.

You control the transmission through the SendStream start and stop methods. When it is first started, the SessionManager behaves as a receiver (sends out RTCP receiver reports). As soon as a SendStream is created, it begins to send out RTCP sender reports and behaves as a sender host as long as one or more send streams exist. If all SendStreams are closed (not just stopped), the session manager reverts to being a passive receiver.

**1.3 Audio and Video Receiving**

Audio and Video both streams are received on different ports. And port number should be even. As port numbers are different, also the player created for audio and video is also different. But you have to merge the both players according to the use. And a total of four ports are used for receiving both the streams, two for each. One port acts as data port and other port act as control port.

To receive the stream we have to provide the IP-address of the user from whom we are receiving the stream and the two data ports at which another user is sending the stream. The two ports at both the sides, transmitter and receiver should be same.

The presentation of an incoming RTP stream is handled by a Player. To receive and present a single stream from an RTP session, you can use a MediaLocator that describes the session to construct a Player. A media locator for an RTP session is of the form:

rtp://address:port[:ssrc]/content-type/[ttl]

The Player is constructed and connected to the first stream in the session. If there are multiple streams in the session that you want to present, you need to use a session manager. You can receive notification from the session manager whenever a stream is added to the session and construct a Player for each new stream. Using a session manager also enables you to directly monitor and control the session.

The diagram explaining how receiving of Audio/Video works shown below:



RTP Reception

JMF Players and Processors provide the presentation, capture, and data conversion mechanisms for RTP streams. A separate player is used for each stream received by the session manager. You construct a Player for an RTP stream through the standard Manager create Player m mechanism. You can either:

* Use a MediaLocator that has the parameters of the RTP session and construct a Player by calling Manager.createPlayer(MediaLocator).
* Construct a Player for a particular ReceiveStream by retrieving the DataSource from the stream and passing it to Manager.createPlayer (DataSource).

If you use a MediaLocator to construct a Player, you can only present the first RTP stream that’s detected in the session. If you want to play back multiple RTP streams in a session, you need to use the SessionManager directly and construct a Player for each ReceiveStream.

**JM Studio**

The JMF Player used for the project is JM Studio. JM Studio is an important tool that allows you to playback, capture, store, transmit or receive media. Before using the application you have to install JM Studio. And register your capture devices for audio and video in JMF Registry. Another application provided with JM Studio.

The JMF Registry is a registry for the Codecs, capture devices that are used by the JMF. Open the capture device tabpane of JMF Registry Editor to find the capture devices that you can use with JMF. Normally you would have audio capture devices. In case you have a JMF-compatible camera in your system that would have also been listed. Otherwise you can detect the capture device and register it with the JM Studio. JM Studio is player developed purely in java. It supports the transmission and reception of media over rtp protocol. JM Studio helps you in very wide way to choose the quality you require for audio and video.

For the video track you can set the following parameters:

(I) Encoding format

(ii) video size and

(iii) frame rate.

For the audio track you can set the following parameters

(I) encoding format

(ii) sample rate

(iii) bits per sample

(iv) number of channels

(v) big/little endian and

(vi) signed /unsigned.

In the application both audio and video are merged in a single JM Studio player. And control of video is provided to the audio player. So that both audio and video can run simultaneously. The control panel of the media player is attached to the application Jframe. The application Jframe is made visible and the player is started. When the user closes the application frame the anonymous WindowAdaptor Object uses the WindowClosing event to stop the media player.

**1.4 Merging Audio and Video**

Merging of audio and video mean that both of the streams should be transferred simultaneously and received by another user simultaneously. This implies that whenever the Players corresponding to the animation video clip is started the Player playing the background music should also be started and similarly both should be stopped simultaneously. Because you cannot receive a merged datasource directly you have to receive two streams individually and then try to merge them on the receiver side. Because of the two different streams two different player are created. Merging audio and video means merging the players and create one player for both audio and video.

One way to achieve the synchronization is as follows.

(a) First set the TimeBase of all the Players to be the same.

(b) Whatever playback control you wish to do, the program should invoke the corresponding Player method ( start, stop, setMediaTime etc ) on all the Players that are to be synchronized. The same is true even if a user controls the playback through a GUI.

Setting the TimeBase of all the Players to be the same is essential because unless the TimeBase of all the synchronizing Players are same, the methods setRate and setMediaTime cannot synchronize the Players. To set the TimeBase of all the Players get the TimeBase from one Player using the getTimeBase() method and set the TimeBase of the rest of the Players using the method setTimeBase().

To solve this problem JMF uses the concept of a control to add extra functionality on to a player. This can allow a single control class to support several different players, while at the same time meaning that the player itself only has to implement a very small public API to provide this added functionality.

The main advantage that controls offer is that they do not affect the state of a player - any operations on the control will not affect the underlying state of the player.

To let the managingPlayer to take control over the managedPlayer we invoke the method addController() as follows,

managingPlayer.addController(managedPlayer).

While invoking this method the managedPlayer should be in the Realized state. If a Player method is invoked on the managingPlayer the corresponding method gets automatically invoked on the managedPlayer. A Player can control any number of Controllers using the above procedure. The managingPlayer posts a completion event only after all its managedPlayers have posted that particular event.

The Player.getControls() method returns an array containing all the controls available for that player. JMF itself defines a couple of controls that may be available, depending on the platform and the type of player. Controls don't really have a standard API to the player, and they are usually designed to take advantage of non-public interfaces in the underlying middleware implementation.

When the application ends all the resources held by the Players should be released by calling the Player.close() method. To let the user indicate when he/she wants to close the application we create a JFrame. When this frame is closed the close() method is invoked on the master Player which will thereby close all the Controlled Player.

**1.5 Cloning of Media Player**

Cloning of Media Player actually means cloning of the datasource. That is the only way multiple players can be produced. When we capture a device it can only be used in one application, means device can be captured if it is not used by any other application. And after capturing stream we can send data to one datasink. But to send data to multiple users, we actually require to clone the datasource from which the stream is captured, so that same stream can be send to the different users. Creating of the clone is only possible with datasource, not with processor. However, processor can be initialized from clonable data source.

A cloneable data source can be used to create clones of either a pull or push DataSource. To create a cloneable DataSource, you call the Manager createCloneableDataSource method and pass in the DataSource you want to clone. Once a DataSource has been passed to createCloneableData- Source, you should only interact with the cloneable DataSource and its clones; the original DataSource should no longer be used directly.

Cloneable data sources implement the SourceCloneable interface, which defines one method, createClone. By calling createClone, you can create any number of clones of the DataSource that was used to construct the cloneable DataSource. The clones can be controlled through the cloneable DataSource used to create them when connect, disconnect, start, or stop is called on the cloneable DataSource, the method calls are propagated to the clones.

The clones don’t necessarily have the same properties as the cloneable data source used to create them or the original DataSource. For example, a cloneable data source created for a capture device might function as a master data source for its clones in this case, unless the cloneable data source is used, the clones won’t produce any data. If you hook up both the cloneable data source and one or more clones, the clones will produce data at the same rate as the master.

Many RTP usage scenarios involve sending a stream over multiple RTP sessions or encoding a stream into multiple formats and sending them over multiple RTP sessions. When a stream encoded in a single format has to be sent over multiple RTP sessions, you need to clone the DataSource output from the Processor from which data is being captured. This is done by creating a cloneable DataSource through the Manager and calling getClone on the cloneable DataSource. A new Processor can be created Transmitting RTP Media Streams from each cloned DataSource, its tracks encoded in the desired format, and the stream sent out over an RTP session.

**2. File Sharing/Transfer**

File Sharing/Transfer module deals with transfer of files with extensions like .doc, .xl, .ppt, .pptx, etc. But the transfer of executable files i.e. .exe files is not allowed. File size limitation is also applied. File greater than 5 MB cannot be uploaded.

File Sharing/Transfer is done on the basis of concept of Client-Server. Each client can upload a file. The file will be uploaded on his PC itself, not at central Server, means each PC acting as server for its uploaded files. File can be downloaded while receiving the video from that particular user. While receiving video streaming from a user, when user clicks on download files, then a screen showing the files uploaded by the user from whom video streaming is received are shown. Then user can select the path of saving that particular file and save it on his/her PC.

File Sharing/Transfer is implemented by using Server class and Socket Server class of java. A socket is on a different port number for different files. When client requests a file it requires the IP address of the user from whom to download the file and port number of the socket on which server for that file is running. The information about port number is fetched from the central database.

The transfer of files is done by using abstract classes Input Stream and Output Stream which are extended by File Input Stream and File Output Stream classes respectively. A File Input Stream obtains input bytes from a file in a file system. What files are available depends on the host environment. File Input Stream is meant for reading streams of raw bytes such as image data. A file output stream is an output stream for writing data to a File or to a File Descriptor. Whether or not a file is available or may be created depends upon the underlying platform. Some platforms, in particular, allow a file to be opened for writing by only one File Output Stream (or other file-writing object) at a time. In such situations the constructors in this class will fail if the file involved is already open. File Output Stream is meant for writing streams of raw bytes such as image data.

The objects of File Input Stream and File Output Stream are respectively used by Buffered Input Stream and Buffered Output Stream respectively for reading from the buffer and writing to the buffer respectively. A Buffered Input Stream adds functionality to another input stream-namely, the ability to buffer the input and to support the mark and reset methods. When the Buffered Input Stream is created, an internal buffer array is created. As bytes from the stream are read or skipped, the internal buffer is refilled as necessary from the contained input stream, many bytes at a time. The mark operation remembers a point in the input stream and the reset operation causes all the bytes read since the most recent mark operation to be reread before new bytes are taken from the contained input stream. The class implements a buffered output stream. By setting up such an output stream, an application can write bytes to the underlying output stream without necessarily causing a call to the underlying system for each byte written.

**3. Whiteboarding**

Whiteboard is a window based GUI using Java, basically to be integrated with the Video Conferencing module. The idea was to transfer graphical information like pie chart, line graph, bar graph, to the user that was difficult using Video Conferencing. It is developed in java using java 2D API, which defines the basic classes and methods to deals with different shapes and text etc. and with different swing utilities.

This graphical data is transferred to the user by the network connection, which is developed by using the Server and Server Socket classes of java. And the transmission of the different shapes is done using abstract classes Input Stream and Object Stream, which are extended by classes Object Input Stream and Object Output Stream respectively.

Also a single user can run multiple instances of whiteboard session with same users or different users as a single session is handled as Thread running on different port number. A whiteboard session can be created as password protected, which allows only the specific users which are required during the session can join. Password protected session asks the user to enter the password before joining into the session which maintains the authenticity of the session. A whiteboard session can also be created as write only. Write only whiteboard session means that only administrator of the whiteboard session can make changes in the session. Other users can join in as participants only. But they are not allowed to make changes, they can only see changes made by the administrator.

The different packages or modules related to whiteboarding are :

**3.1 Whiteboard User Interface**

User Interface of the whiteboard is developed using different swing utilities present in java. This package deals with the placement of different components on the main container and different user interface required apart from the visible ones, like font list, color selection, type of pen stroke etc. The module also deals with designing the interface for password dialog and status bar.

Also control point i.e. different cursor positions are also described in this package. And the images for shortcut buttons are loaded from data package. For the working of different objects like ellipse, rectangle, text classes are defined in different module i.e. whiteboarding.model.

Different components on the main container are whiteboard screen, menu bar for leaving the session or group/ungrouping the objects etc. , shortcut options for the options given in menu bar i.e. leaving the session, bringing the object in front or sending the object behind, grouping the different objects or ungrouping the different objects, deleting the selected object. Another vertical bar is there having options to create ellipse, rectangle, enter text, claiming an area, selecting an object, rotating an object, flipping an object, selecting an object etc. Another bar has options to select the font size, width of the object, selecting the different type of pen stroke and option to fill the objects with color or selecting the color of the font.

The main container also has a split window at the bottom, at right side of which it shows the name of the users who have joined the session. At the left of split window there is text area which shows the time of different type of activities done by different users like joining in, leaving the session and any other activity. At the bottom of the split window there is text field at which user can enter any string to be shown at the text area. Using this way, user can chat in a session. At the bottom of the window, there is one more text filed that help users by indicating how to use that object in the session.

**3.2 Whiteboard Model**

Whiteboarding.model describes the model of the whiteboard session. It uses java 2D classes to describe the working of different objects to be used in the whiteboard session like basic shapes ellipse, rectangle, line, polyline and text etc. It also contains the java class which represent any object as an entity and an entity ID is given to the different object. Distinction of the objects of one user from other is also done by storing the user information and related entities and entity Ids. Also to relate the different shortcuts with different functions is described in Whiteboarding.model package.

Apart from describing various entities it also contains the information that which changes can be done for what object, what are the options available for different objects. Different objects can be treated as a group and the working of using different objects, shapes as a group is defined in the same package.

To make changes in the different entities, it is required to define at which position changes can be made, for that we require anchor point. Anchor points help when are to make any change, because it gives the new position i.e. new coordinates for the entity. Anchor points will appear while selecting an object or group. They help in rotating the object, flipping the object and moving the object or group.

This model also defines the basic changes possible with pen or brush. Both the classes are widely used. Pen helps while to draw any object like ellipse, rectangle or line. And brush helps in selecting the color of the object or the filling color.

Changes related to the chatting state are handled here. Any change related to group chat are first changed in the class and then the changed group state or change chat state is passed to other users.

This package maintains the session state and history related to the session. So that when a new user joins in, session state can be loaded in the same state as it is now and history helps in listing out the different events taken place during the session.

An option to claim an area is provided in this whiteboard container. This means that a user can select an area that will belong to only that user. Any other user cannot make any other object in that area. This is done by relating a hash code with related user. If any other user tries to access that area, a warning is raised for the user. A claimed area is only visible to other users, bur not editable for other users.

Whiteboarding.model is used by Whiteboarding.userinterface to relate the working of different entities and shortcut options listed and will use it as described in Whiteboarding.model.

**3.3 Whiteboard Network Connections**

Whiteboarding.network package defines the way network connection is maintained between the users. And it also keeps a record of the user who is admin of which server, so that when the administrator of a particular server logs out or closes the session, then other members who have joined the session can be informed that the session has been closed by the server owner.

Network connection between the users or the session server is implemented with the help of Server and SocketServer classes of java, implemented as a concept of sessionserver and sessionclient. Sessionclient runs through the initial logon procedure after joining a session, requesting the necessary data from the remote session server to bring its local SessionModel up to date. It relays change in entity and old-history request Messages from the session control to the remote session server and returns reply Messages when received. It acts as an intermediary between the SessionModel and the session control to avoid concurrent access to the former. It receives change in entity messages from the remote session server and passes them on to the SessionModel for integration.

The session server holds a server socket open and accepts connections. It relays Messages from remote session clients to the local session client. It may block entity change messages from remote session clients deemed to be read-only users within this session. It serves the data for other clients logging on to the session, from its local reference SessionModel, well as old-history requests. It validates and then integrates entity change messages sent from remote clients.

The package also keeps the track of types of message which will be sent across the network by the server or by the client. The various messages that can be send across the network by the session browser server are requesting session list, requesting session creation, session list updation, session list removal, actual session list. And those which are sent between session client and session server are requesting start up data, actual start up data, suggestion of changing the group, denial of changing the group, changing the group, requesting the client id by server, sending actual client id, request of closing the session.

The number of sessions can be run by a single user by the use of multithreading concept. Package also helps in starting different connection by a single user by creating threads. The sending of information or objects is done by the input stream and output stream classes, which are actually used by the object input stream and object output stream objects respectively. ObjectInputStream ensures that the types of all objects in the graph created from the stream match the classes present in the Java Virtual Machine. Classes are loaded as required using the standard mechanisms. The method readObject is used to read an object from the stream. An ObjectOutputStream writes primitive data types and graphs of Java objects to an OutputStream. The objects can be read (reconstituted) using an ObjectInputStream. Persistent storage of objects can be accomplished by using a file for the stream. If the stream is a network socket stream, the objects can be reconstituted on another host or in another process. The method writeObject is used to write an object to the stream. Any object, including Strings and arrays, is written with writeObject. Multiple objects or primitives can be written to the stream. The objects must be read back from the corresponding ObjectInputstream with the same types and in the same order as they were written.

**4. P2P Text conversation**

A window based GUI which deals with one to one peer to peer conversation with the help of text data like sending messages. The Idea was to design the graphical user interface for one to one person. Because in whiteboard session you can have text conversation, but all the users of the session will participate, not person to person.

Each user will have a server running within application which will accept to ask the confirmation of acceptance of the connections made by other users. If connection is accepted then it will generate a window for text conversation with the user. For having conversations with multiple users at the same time, concept of multithreading is used. Each thread will be running on different port number. When a connection is accepted by a user, then the port number is incremented by two for the availability of the other user.

The sending of messages between peers is done by java classes, datagram socket and datagram packet which use user datagram protocol for sending the messages between remote users. User datagram protocol is a connectionless protocol. No connection is established first. Packets are sent independently. That is why it is not much reliable protocol. Datagram socket represents a socket

for sending and receiving datagram packets. A datagram socket is the sending or receiving point for a packet delivery service. Each packet sent or received on a datagram socket is individually addressed and routed. Multiple packets sent from one machine to another may be routed differently, and may arrive in any order. Datagram packets are used to implement a connectionless packet delivery service. Each message is routed from one machine to another, based solely on information contained within that packet. Packet delivery is not guaranteed with the use of datagram packets because it uses user datagram protocol.

**5. Database – mysql 5.5**

Mysql 5.5 is the database used at the backend of the application to store the profile information of the user and other detail of the user like the current IP address or the files uploaded by the user etc. The name of the database for the application is videoconferencing. The database for the application will be central so that each user can make contact with the database, so that each user can fetch information and store information in the central database which will also be used by the other users of the application. Because the database is central and will be accessed by all users, the database if designed with granting the access of database from remote location. It has four tables the details of which are provided below.

1. Personaldetail

The table has fields username, password and email-id. The row is inserted into the database when a new user signs up for the application. And updation of the table takes place when the user changes his password or email id. Username is the primary key of the table, which will also act as reference to other foreign keys in the database.

2. Userpersonaldetail

The table has fields username, name, date of birth, address, phone number, mobile number, email id. Email id will refer to the email id field of personal detail of same username. Username is primary key and foreign key in this table which will refer to the primary key of the personal detail table. The insertion and updation of the table is done when user enter his data or updates his details in the personal profile detail.

3. Status

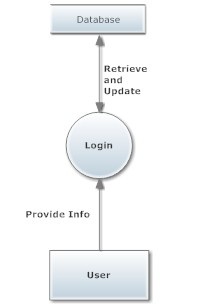
Status is the most important table for application which contains the detail of current status of the user like whether he is logged in or not, his current ip etc. The fields of the table are username, loggedin, chat, ip, rport, tport, fileport. Username is primary and foreign key of the table which refers to the personal detail table. Loggedin filed gives the detail whether the user is logged in or not, chat indicates whether the user is currently in text chat or video conferencing. Ip contains the current ip address of the user. Rport, tport, contain port number for the user that is free means that at which port the next user will contact.

4. Filename

Filename contains the detail about which file is uploaded by which user at which port number. The fields of the table are username, filename, fileport. Username is foreign key which refers to the personal detail table. Filename contains the name of the file uploaded and corresponding fileport contains the port number at which the file is uploaded by the user.

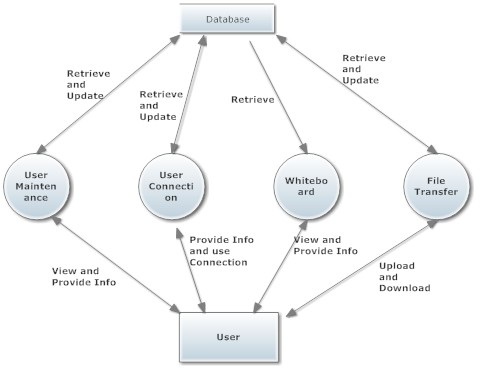
**5. Data Flow Diagrams**

5.1) Login :



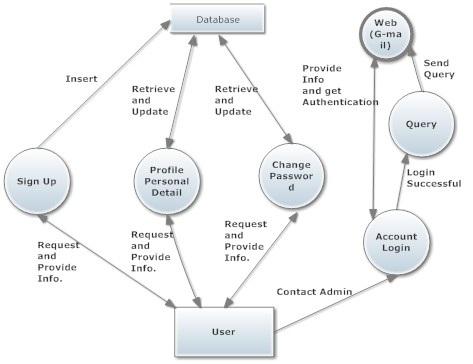
Login

5.2) Videoconferencing along with Interactive Whiteboard



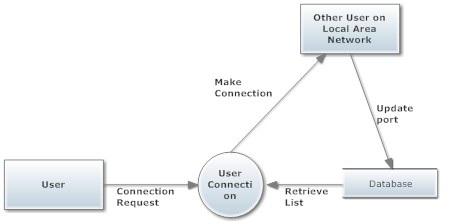
Videoconferencing along with Interactive Whiteboard

5.2.1) User Maintenance



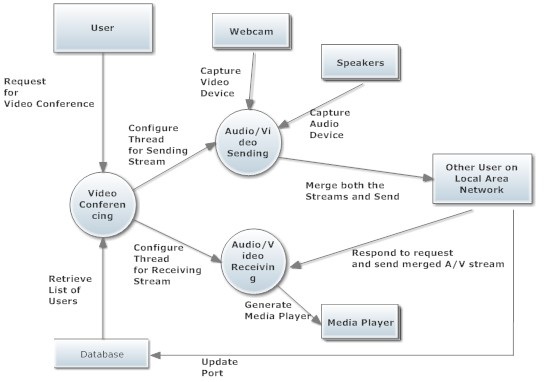
User Maintenance

5.2.2) User Connection



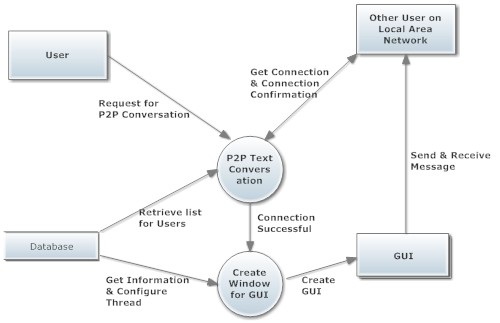
User Connection

5.2.2.1) Videoconferencing



Video Conferencing

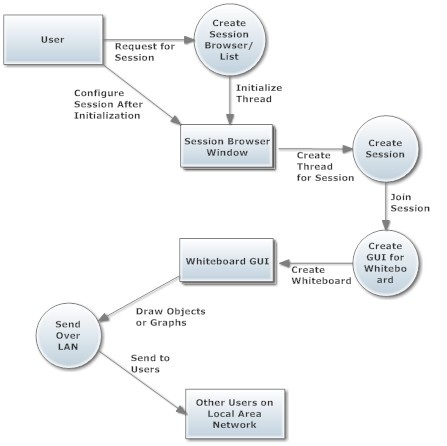
5.2.2.2) Peer to peer text conversation



P2P Text Conversation

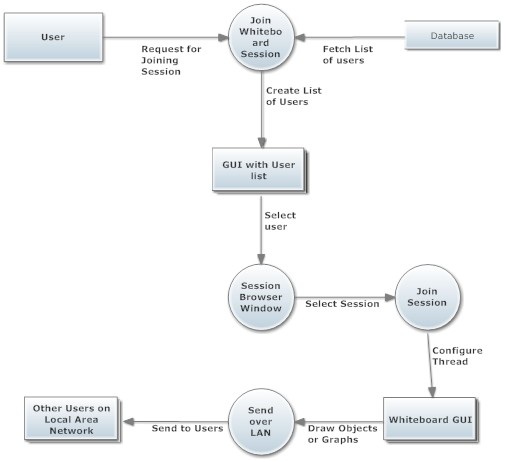
5.2.3) Whiteboard

5.2.3.1) Whiteboard server session creation



Whiteboard Session Server Creation

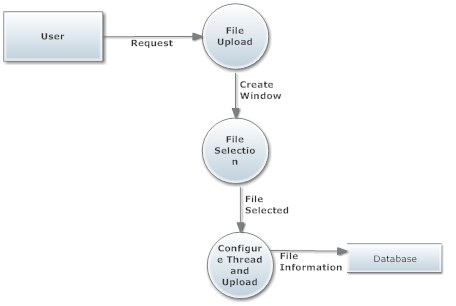
5.2.3.2) Whiteboard Client Session



Whiteboard Client Session

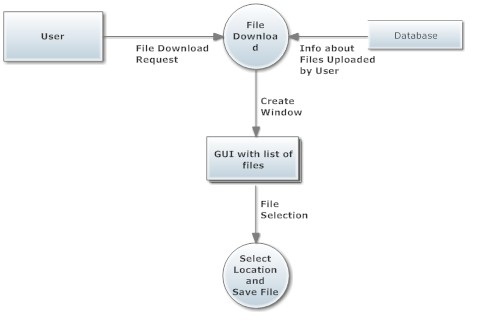
5.2.4) File Transfer

5.2.4.1) Upload



File Upload

5.2.4.2) Download



File Download

**6. Details of the work**

The project’s objective is to

1. Only authenticated users can use this application.

2. Provide the graphical user interface so that users within the LAN can communicate with each other.

3. Send files like .doc, .xl etc. to each other.

4. Users can participate in a conference or session using graphical user interface.

5. Authenticated users can enquire about any aspect of the application.

For the project’s usage requirements are:

1. At least two computer or laptops connected within same local area network and having access to database.

2. For videoconferencing, the computers or laptops must have integrated webcam or external webcam.

3. The devices that will be used i.e. speakers and webcam must be registered with jmstudio.

4. Webcam must be started in instant messaging mode.

The project divided into different small modules contains modules named as:

1. User Maintenance

2. Videoconferencing

2.1 User Maintenance

2.2 Audio and Video Sending

2.3 Audio and Video Receiving

2.4 Merging Audio and Video

2.5 Cloning of Media Player

3. File Transfer

4. Whiteboarding

4.1 Basic user interface of the Whiteboard

4.2 Developing the model of the Whiteboard that describes how it will work

4.3 Different Networking Connection required to handle the Whiteboard Session.

5. P2P text conversation

**7. Testing – state of different test cases taken to test the project**

The most important activity at the implementation stage is the system testing with the objective of validating the system against the designed criteria. During the development cycle, user was involved in all the phases that are analysis, design and coding. After each phase the user was asked whether he was satisfied with the output and the desired rectification was done at the moment. During coding, generally bottom up technique is used. Firstly the lower level modules are coded and then they are integrated together. Thus before implementation, it involves the testing of the system. The testing phase involves testing first of separate parts of the system and then finally of the system as a whole. Each independent module is tested first and then the complete system is tested. This is the most important phase of the system development. The user carries out this testing and test data is also prepared by the user to check for all possible combinations of correct data as well as the wrong data that is trapped by the system. So the testing phase consists of the following steps:

* Unit testing:

In the bottom of coding technique, each module is tested individually. Firstly the module is tested with some test data that covers all the possible paths and then the actual data was fed to check for results.

* Integration testing:

After all the modules are ready and duly tested, these have to be integrated into the application. This integrated application was again tested first with the test data and then with the actual data

* Parallel testing:

The third in the series of tests before handling over the system to the user is the parallel processing of the old and the new system. At this stage, complete and thorough testing is done and supports out the event that goes wrong. This provides the better practical support to the persons using the system for the first time who may be uncertain or even nervous using it.

The testing will be performed considering the following points:

1) Clerical procedure for collection and disposal of results

2) Flow of data within the organization

3) Accuracy of report output

4) Software testing which involves testing of all the programs together. This involves the testing of system software utilities being used and specifically develops application software.

5) Incomplete data formats

6) Halts due to various reasons and the restart procedures.

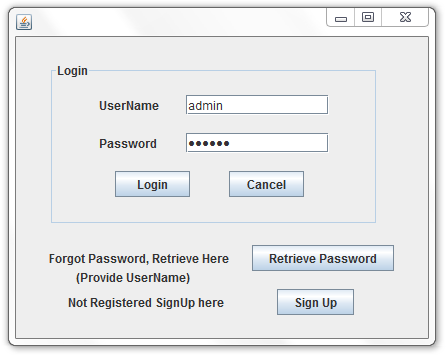
7) Range of items and incorrect formats

8) Invalid combination of data records.

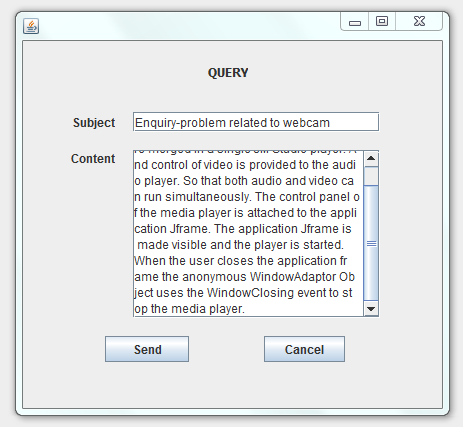
9) Access control mechanism used to prevent unauthorized access to the system.

Some snapshots that show the different views of the project while testing:

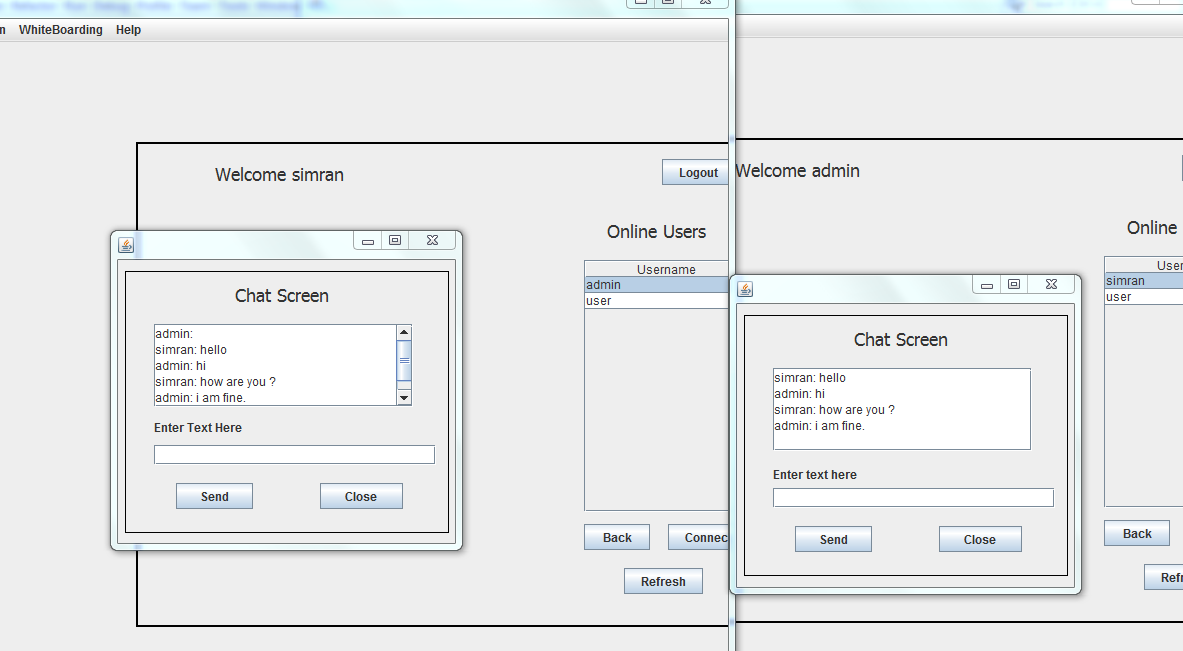
Login



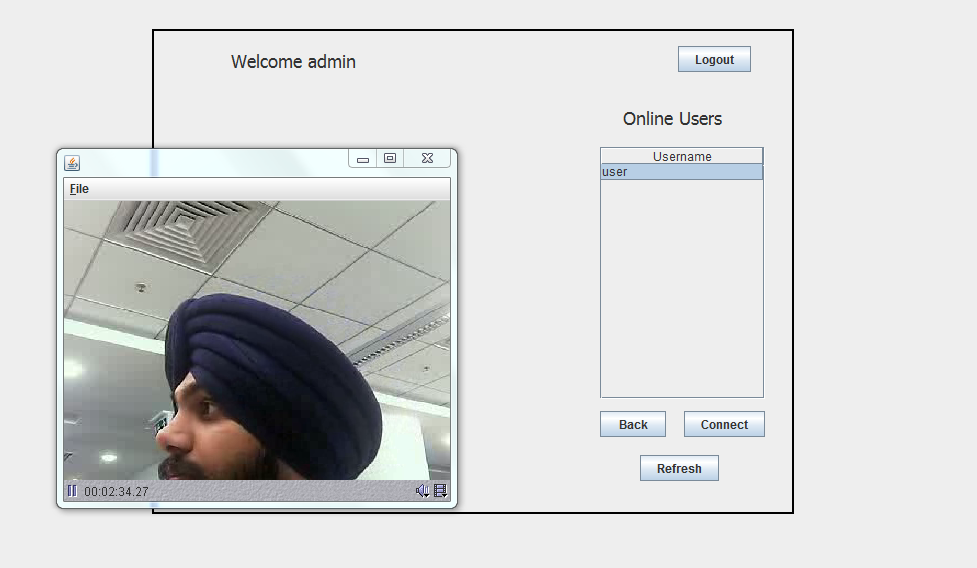
Query



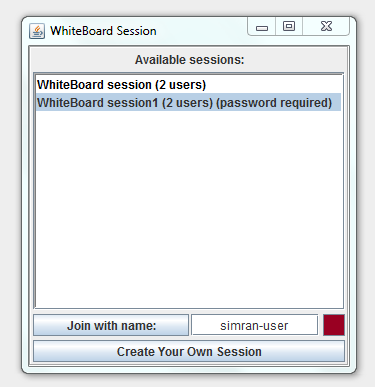
P2P text



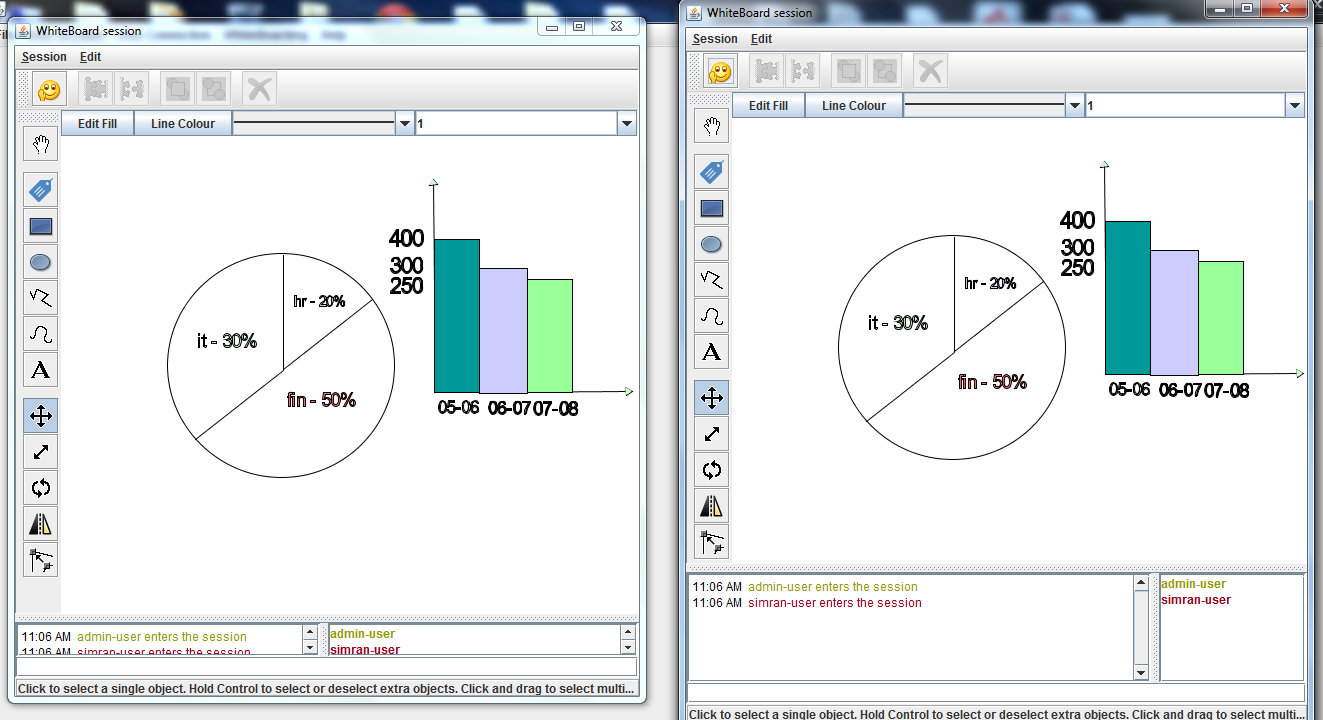
Videoconferencing



Whiteboard browser



Whiteboard GUI



**8. Result, Conclusion and Future scope of work**

**Result**

Result of the above work is a window application in which

* Only registered users can use this application.
* Application is for users within same network. Only users within same local network area can be authenticated users.
* Employees within same local network area can simultaneously communicate with each other using videoconferencing.
* Employees can share graphical information and other visual data or can do group text chat using whiteboard.
* History of the whiteboard session is maintained so that whenever a new user joins in the same information can be presented to him.
* Files like .doc, .xl, .ppt, .pptx etc. can be shared among users of the application.

**Future Scope**

Future scope of the project is

* Data sharing is a very wide field. Many other plug-ins related to data communication can be added to project like whiteboard added as a plug-in the project.
* Files are transferred within users. Whiteboard can be extended so that files can be shared with session and changes made to files are shown immediately in the session.
* The project developed has window interface. Future version can be developed with web technologies so that the product can be hosted on a website.
* The number of users allowed in videoconferencing is six depending on the quality of service and bandwidth provided. In future versions the number can be increased considerably.
* To stand with more new technology of modern world window interface can be changed to Mobile Interface.

**9. References**

1. The Complete Reference J2SE, 7th Edition by Herbert Schildt, Tata McGraw Hill Publication.
2. Oracle java documents (http://docs.oracle.com/javase/6/docs/)
3. Oracle java API docs (http://docs.oracle.com/javase/6/docs/api/)
4. Java Media Framework 2.0 API guide by Oracle
5. Java Media Framework Basics by IBM (www.ibm.com/developerworks/java/tutorials/j-jmf/j-jmf-pdf.pdf)
6. Tutorial : Getting started with the Java Media Framework by IIT Madras (www.ee.iitm.ac.in/~tgvenky/JMFBook/Tutorial.pdf)
7. JMF 2.1.1 Solutions by Sun (http://web.archive.org/web/20080216001555/java.sun.com/products/java-media/jmf/2.1.1/solutions/)
8. Java 2D API guide by Oracle
9. Java 2D API solutions by Sun (http://java.sun.com/products/java-media/2D/index.jsp)
10. Tutorials: Java 2D by deitel
11. Oracle Forums (https://forums.oracle.com/forums/)