Kumaraguru College of Technology
Department of Computer Science and Engineering
Coimbatore – 641006.

September 2004 P – 1239

SUBSCRIBER MANAGEMENT

Project Work done at

SRM Systems and Softwares., Coimbatore

PROJECT REPORT

Submitted in partial fulfillment of the
Requirements for the award of the degree of

M.Sc. Applied Science (Software Engineering)
Bharathiar University, Coimbatore

Submitted by

Nithya.K
Reg. No. : 0137S0042

Internal Guide: Mr. S. Hameed Ibrahim, M.C.A.,
Dept. of Computer Science & Engineering,
Kumaraguru College of Technology,
Coimbatore.

External Guide: Mr.N.Venkatesh, M.C.A.,
SRM Systems & Softwares
R.S. Puram
Coimbatore -641008

KUMARAGURU COLLEGE OF TECHNOLOGY

(Affiliated to Bharathiar University)

COIMBATORE – 641006

BONAFIDE CERTIFICATE

This is to certify that this is the Bonafide Project Record Work done by **Nithya.K,** Reg.No **0137S0042** in partial fulfillment for the award of Degree of **M.Sc. [SOFTWARE ENGINEERING]**, during the academic year 2004 –2005.

[illegible]
Prof & HOD

[illegible] 21/9/04
Guide

This Project entitled "SUBSCRIBER MANAGEMENT " is submitted for the V11 semester of M.Sc. [SOFTWARE ENGINEERING] for Bharathiar University Project Viva-voce examinations held on .....30:9:04...........

[illegible]
Internal Examiner

[illegible]
External Examiner

SRM Systems and Software Ltd.

24, G.N. Chetty Road, T. Nagar, Chennai - 600 017, INDIA.
Ph. : 91-44-28217918, 28269471, 52118632 Fax : 28283359
Website: http://www.srmsoft.com

15-September-2004

CERTIFICATE

This is to Certify that the Project work entitled **"Subscriber management"** was Analyzed, Designed and Developed by **Ms. K.Nithya (Reg. No. 0137S0042)** of **Kumaraguru College of Technology, Chinnavedampatti, Coimbatore,** submitted in partial fulfillment of the requirements of degree of **M. Sc., Software Engineering** has been carried out in our organization from June 2004 to September 2004. This project has been developed using **VC++, VB & Ms-Access.**

We wish success in all her future endeavors.

For SRM Systems and Software Limited

[illegible]

Manager - Projects

DECLARATION

I hereby declare that the project entitled **Subscriber Management,** submitted towards the fulfillment of M.Sc Applied Science (Software Engineering) from Bharathiar University is a record of original work done by me under the supervision of **Mr.N.Venkatesh, M.C.A.,** SRM Systems and Softwares Coimbatore and **Mr.S. Hameed Ibrahim, M.C.A.,** Lecturer, Dept of Computer Science and Engineering, Kumaraguru College of Technology and this project work has not formed the basis for the award of any Degree/ Diploma/Associate-ship/Fellow-ship or any other similar title to any candidate of any University.

Place: coimbatore

Date: 24.9.04

[illegible]

Nithya.K

REG NO:0137S0042

ACKNOWLEGEMENT

I express my sincere gratitude to **Dr. K.K Padmanabhan,** Ph.D., esteemed Principal, Kumaraguru College Of Technology for giving me this opportunity to do the project work and providing facilities in the college to make it possible for me to complete my work without difficulties.

I express my profound gratitude to **Prof. S. Thangasamy, B.E (Hons), Ph.D, Head of the Department**, Computer Science and Engineering, Kumaraguru College of Technology who motivated me with his valuable ideas and support in doing this project.

I express my deep sense of gratitude and indebtedness to **Mr. S. Hameed Ibrahim, M.C.A**, Senior Lecturer, Dept of Computer Science and Engineering, Kumaraguru College of Technology for her invaluable guidance throughout the length of this project.

I am grateful to **Mr.N.Venkatesh, SRM Systems and Software** for giving me an opportunity to do the project in their esteemed organization.

I also thank my **Beloved Parents** who have been a pillar of support for me in completing this project, my friends and the department teaching and non-teaching staff for their support in completing this project.

SYNOPSIS

The project entitled "Subscriber Management" is developed for SRM systems and Software.

"To develop user interface, which interacts with the TV tuner card driver that helps in receiving the live stream of signals from the TV tuner card, allows tuning-in to a different channel and displays video and produce audio. It also helps in capture the signal and allows the user to store the live video clips into the hard disk in compressed format. This is facilitated by developing a GUI for the TV tuner card by interacting with its driver that should capture, display and store channel. This project displays the TV channels on video capture screen in the application. It allows the user to view the available TV channels, capturing live video from a particular channel and allowing it to store on a specified location in the storage media.

The key activities of the system are

- Display available TV channels.
- Store the programs from the channel that is being viewed.
- New users may be added , also modifications may be done to password of the existing users.
- Video window can be viewed with options like start capture and stop capture buttons enabled.
- Server has given the option to view the details of the authorized users for this reference.
- An efficient message passing mechanism is being used between the server and clients

THE SUBSCRIBER MANAGEMENT SYSTEM
CONTENTS

1. INTRODUCTION

1.1 PROJECT OVERVIEW

The Subscriber's Management is a part of the DSM, which is a product of SRM Systems and Software. This is facilitated by developing a GUI for the TV tuner card by interacting with its driver that should capture, display and store channel. This project displays the TV channels on video capture screen in the application. It allows the user to view the available TV channels, capturing live video from a particular channel and allowing it to store on a specified location in the storage media.

LOGIN

Using this form, only authorized users are granted access to the system.
This form ensures that our software remains secure.

SERVER

Only the server can have access to this part of the software.

- New users may be added, also modifications may be done to password of the existing users.
- Video window can be viewed with options like start capture and stop capture buttons enabled.
- Server has given the option to view the details of the authorized users for this reference.
- An efficient message passing mechanism is being used between the server and clients.

CLIENT

Clients are given access to this part of the software

- An efficient message passing mechanism is being used between the server and clients.
- Using this mechanism client can have interaction with the server for any details needed.
- Video window can be viewed with options like start capture and stop capture buttons disabled

1.2 ORGANISATION PROFILE

THE GROUP:

- SRM SYSTEMS AND SOFTWARE was founded in 1983 with an enthusiastic and enterprising attitude.

- SRM SYSTEMS AND SOFTWARE has been one of the leading professional makers of software products in the southern India.

THE STRENGTH

- SRM SYSTEMS AND SOFTWARE is a customer focused organization built on long term relationships through quality services and established value chains.

- SRM SYSTEMS AND SOFTWARE maintains stringent quality assurance and testing processes at all levels in the design and development of software products and services.

- SRM SYSTEMS AND SOFTWARE provides integrated knowledge enhancement capabilities both for the customer and the technologist within the organization.

2. SYSTEM STUDY AND ANALYSIS

2.1 SOFTWARE REQUIREMENT SPECIFICATION

Visual Basic – An Introduction

Visual Basic is developed by Microsoft. It provides a fast and easy way to create applications for Window. Visual Basic provides you with a complete set of tools to simplify rapid application development. "Visual" refers to the method used to create the graphical user interface (GUI).

The "Basic" refers to the BASIC (Beginners All - Purpose Symbolic instruction Code) language, a language used by more programmers. BASIC helps the programmers to learn easily about what a programming language is and how to write error free programs.

Using GUI we can add the existing pre-build objects to the screen instead of writing codes to describe the appearance and location of interface elements or the objects that we use to display the data. Visual Basic has evolved from the original BASIC language and now contains several hundred statements, functions, and keywords, many of which relate directly to the Windows GUI.

Visual Basic includes advanced features such as native code compilation, High speed database access, and an improved development environment.

Beginners can create useful applications by learning just a few of the keywords, yet the power of the language allows professionals to accomplish anything that can be accomplished using any other Windows programming language.

Visual Basic Tools

Data access features which allows us to create databases, front-end applications for most popular database formats like Microsoft SQL Server and other enterprise level databases.An ActiveX technology allows us to use the functionality provided by other applications, such as Microsoft Word (word processor), Microsoft Excel spreadsheet, and other Windows applications. A finished application is a true .exe file that uses a Visual Basic Virtual Machine that we can freely distribute.

Visual Basic Editions

The Visual Basic Learning edition allows programmers to easily create powerful applications for Microsoft Windows and Windows NT. This edition consists of all intrinsic controls along with grid, tab and data-bound controls.

The Professional edition provides computer professionals with a full-featured set of tools for developing solutions for their clients. This edition consists of all the features of the Learning edition along with additional ActiveX controls, integrated Visual Database Tools and Data Environment, Active Data Objects, and the Dynamic HTML Page Designer.

The Enterprise edition allows professionals to create distributed applications by a team of programmers. This edition consists of all the features of the Professional edition along with Back Office tools such as SQL Server, Microsoft Transaction Server, Internet Information Server and more.

Visual Basic – An Event-Driven Application

An event is an action recognized by a form or control. Event-driven applications execute Basic code in response to an event. Each form and control in Visual Basic has a predefined set of events. If one of these events occurs and there is code in the associated event procedure, Visual Basic invokes that code. We have to decide how they will respond to a particular event.

A section of code (an event procedure) corresponds to each event. When we want a control to respond to an event, we write code in the event procedure for that event.

MICROSOFT VISUAL C++

Microsoft Visual C++ provides an Integrated Development Environment (IDE) for developing applications. Win 32 application programming interface (API) functions are used in the program development. The Direct X SDK is a part of Microsoft Software Development Kit (SDK) .The DirectX is a specialization on the multimedia programming. The basic knowledge of COM is needed to use the DirectX API's.Win 32 Dynamic Link Libraries programming environment in VC++ is used to create DLL file. Core functionality of the project is implemented as a DLL file. Microsoft Visual Basic provides an Integrated Development Environment (IDE) for developing applications for Windows Platform. Visual Basic is used to develop the user interface. The whole project is converted into EXE file using Visual Basic. The DLL file created using Visual C++ is declared in Visual Basic which calls the functions as per the user requirement. The coding process and the code itself are well documented to prevent any ambiguity. The variables and constants have been named with due consideration of easiness in code review.

ABOUT THE DIRECT X

Microsoft® DirectX® is a set of low-level application programming interfaces (APIs) for creating games and other high-performance multimedia applications. It includes support for two-dimensional (2-D) and three-dimensional (3-D) graphics, sound effects and music, input devices, and support for networked applications such as multiplayer games

ABOUT THE DIRECTSHOW

Microsoft® DirectShow® is an architecture for streaming media on the Microsoft® Windows® platform. DirectShow provides for high-quality capture and playback of multimedia streams. It supports a wide variety of formats, including Advanced Streaming Format (ASF), Motion Picture Experts Group (MPEG), Audio-Video Interleaved (AVI), MPEG Audio Layer-3 (MP3), and WAV files. It supports capture using Windows Driver Model (WDM) devices or older Video for Windows devices. DirectShow is integrated with other DirectX technologies. It automatically detects and uses video and audio acceleration hardware when available, but also supports systems without acceleration hardware.

DirectShow simplifies media playback, format conversion, and capture tasks. At the same time, it provides access to the underlying stream control architecture for applications that require custom solutions. You can also create your own DirectShow components to support new formats or effects. Examples of the types of applications you can write with DirectShow include DVD players, video editing applications, AVI to ASF converters, MP3 players, and digital video capture applications.

ORACLE 9i– AN INTRODUCTION

Oracle 9i is an ORDBMS, which is used as a backend database to maintain the information Oracle database can be connected to any application through the ODBC Bridge. Oracle 9i is an enhanced version of the previous oracle versions and is very much useful in information management environment. It provides the features to store very large amounts of data, to provide users rapid access methods to retrieve and modify data and to provide security to the data. The Oracle 9i server allows for the sharing of data between applications; the information is stored in one place and used by many systems.

ORACLE 9i SERVER CONFIGURATIONS

Oracle 9i is Host-based in which the users are connected directly to the same computer on which the database resides. Oracle 9i is developed using Client/Server technology, so the user can access the database from their personal computer (Client) via a network and the database from a separate computer (Server).

Oracle 9i provides Distributed Processing, so the user can access a database that resides on more than one computer. The database is spread across more than one machine, and the users are unaware of the physical location of the data they work with. Oracle 9i also provides Web-enabled computing, it has ability to access data from an Internet-based application.

2.2 EXISTING SYSTEM

As far as the existing system is concerned, there is no existing system here in SRM Systems Software for the system titled "Subscriber Management''. The Subscriber Management will display the TV channels available in the TV tuner card by interacting with its driver.

3. PROGRAMING ENVIRONMENT

3.1 HARDWARE CONFIGRATION

Processor:	Pentium
Clock Speed:	166 MHz
Memory:	32MB Ram
Hard Disk:	8 GB
TV Tuner card:	TV Tuner Card with its Driver developed on Windows Driver Model (WDM) architecture. TV cable provided by cable operator.

3.2 DESCRIPTION OF SOFTWARE AND TOOLS USED

Operating System:	Windows 98 /Windows 2000
Language :	Microsoft Visual C++
Front End :	Microsoft Visual Basic 6.0.
Back End :	Oracle 9i
Development Kit:	Direct X8.0 SDK.

4. SYSTEM DESIGN

The process of design involves "conceiving and planning out in the mind" and "making a drawing, pattern or sketch of it". The design is concerned with identifying software components, the general modular structure of the software, the function provided by each module and the internal data stream and store that makes up the interface between modules.

4.1 INPUT DESIGN

Input plays the most important role in completion of system. Input design is the process of converting user-originated inputs into computer- based format. The goal of designing input data is to make data-entry as easy as possible and error-free.

Input serves 4 purposes

- To control work flow
- To reduce redundancies in recording data
- To allow easier checking of data
- To increase clerical accuracy

System design is the path, which leads to the creation of a new system. It provided the understanding and procedural details necessary for implementing the system recommended in the feasibility study. Emphasis is on translating the performance requirements into design specifications. System analysis focuses on the logical, implementation-independent aspects of the system (the requirements), while system design deals with the physical implementation aspects of a system (technical specifications).

4.2 DATABASE DESIGN

The database required for the system is created using ORACLE 9i.The tables are created that is required to hold the information. The information about the channel number, channel name, hours, minutes and seconds are recorded for each channel that is viewed by the user. Using this information viewer information is generated. OLE DB connectivity is used to connect the oracle 9i with the Visual Basic.

TABLE NAME: USER DETAILS

DESCRIPTION: Contains all the details of users allowed by the server.

COLUMN NAME	TYPE	NULL?
User name	Varchar2(20)	Not null
Password	Varchar2(20)	Not null
Con_password	Varchar2(20)	Not null
Category	Varchar2(20)	Not null

TABLE NAME: MESSAGE DETAILS

DESCRIPTION: Contains all the details of message transaction.

COLUMN NAME	TYPE	NULL?
From1	Varchar2(20)	Not null
To1	Varchar2(20)	Not null
Message	Varchar2(50)	Not null
Date1	Varchar2(20)	Not null
Subject	Varchar2(20)	Not null
Status	Varchar2(20)	Not null

TABLE NAME: CHANNEL TIMER

DESCRIPTION: Contains all the details of auto storing facility by the server.

COLUMN NAME	TYPE	NULL?
Channel name	Varchar2(20)	Not null
Starting time	Varchar2(20)	Not null
Ending time	Varchar2(20)	Not null
View time	Varchar2(20)	Not null

4.3 PROCESS DESIGN

The Subscriber's Management application is divided into many program modules. Microsoft Visual C++ is used to program the capture, display and store modules. The functionalities of these modules are implemented as DLL's. Microsoft Visual Basic is used to develop User Interface. Microsoft Access provides the database required for the application. The program modules used are

- User Interface
- Capture
- Display
- Store

USER INTERFACE MODULE

- The GUI for the subscriber management is developed using Visual Basic 6.0
- The user interface has a full-fledged feature for the user to manage the details of subscription.

CAPTURE, DISPLAY AND STORE MODULES

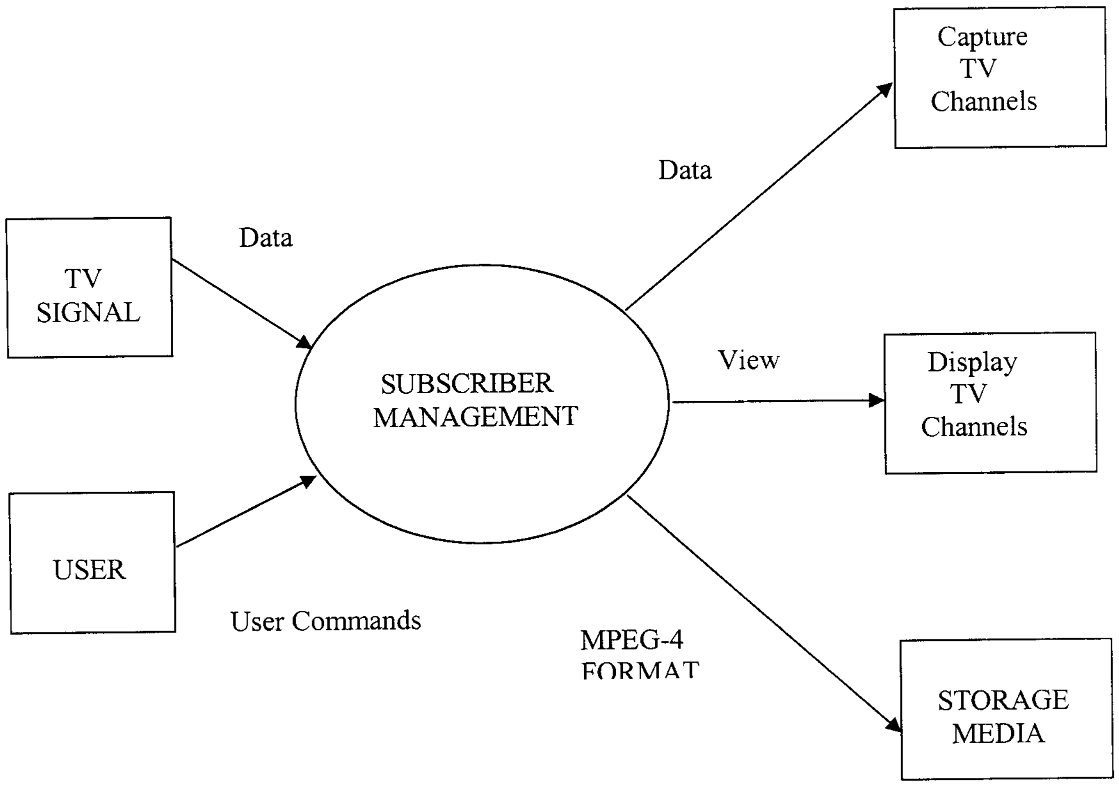
- These modules are programmed using VC++ 6.0.
- These modules are converted into Dynamic Link Libraries (DLL) using Win32 Dynamic Link Library in VC++6.0
- Display and store module needs filter graph to be constructed twice since only one filter graph can use the source at one time.
- Interfaces used in the filter graph should be released once the process is over. This is to allow the source to be used by other process.

4.4FLOW CHART

CONTEXT LEVEL DIAGRAM

The first step involved in creating a model of the system is to establish the fundamental activities being performed by the system. An attempt is made to delineate the boundary of the system by specifying all that is immediately outside it. We show all the inputs and outputs of a system which also represent its interaction with all external entities of the system.

CONTEXT DIAGRAM FOR SUBSCRIBER MANAGEMENT



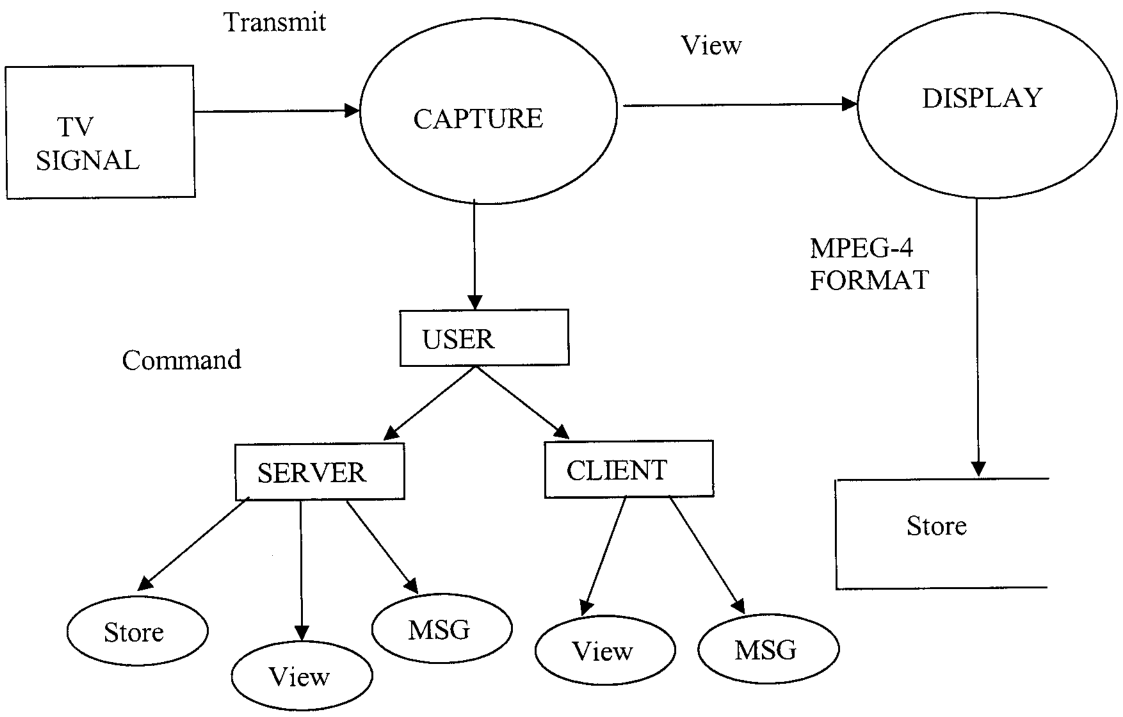
DATA FLOW DIAGRAM

Data flow diagram is used to define the flow of the system and its resources such as information. Data flow diagrams are a way of expressing system requirements in a graphical manner. Data flow diagrams represent one of the most ingenious tools used for structured analysis. A Data flow diagram or DFD as it is shortly called is also known as a bubble chart. It has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. It is the major starting point in the design phase that functionally decomposes the requirement specifications down to the lowest level of detail.

A DFD consists of a series of bubbles joined by lines. The bubble represents data transformation and lines represent flow in the system. In the normal convention, a DFD has four major symbols.

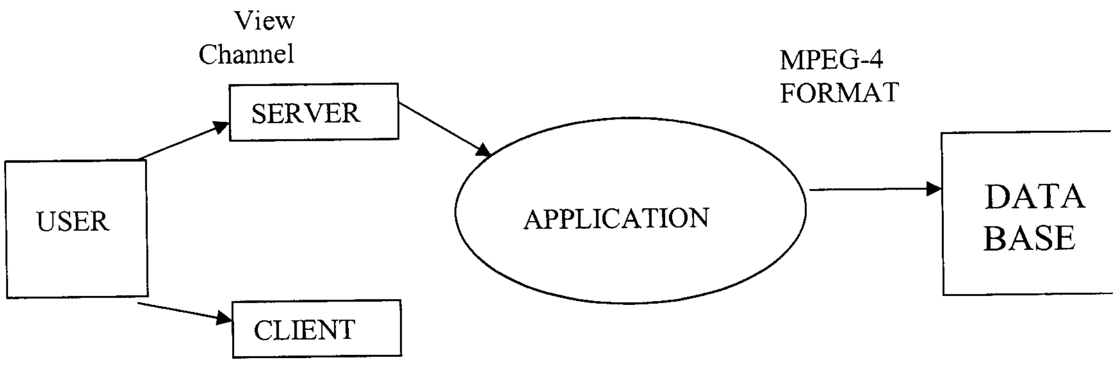
- Square, that defines source or destination of data.
- Arrow, which shows data flow.
- Circle, which represents a process that transforms incoming data into outgoing flow.
- Open rectangle, which shows data store.

LEVEL1: DATA FLOW DIAGRAM



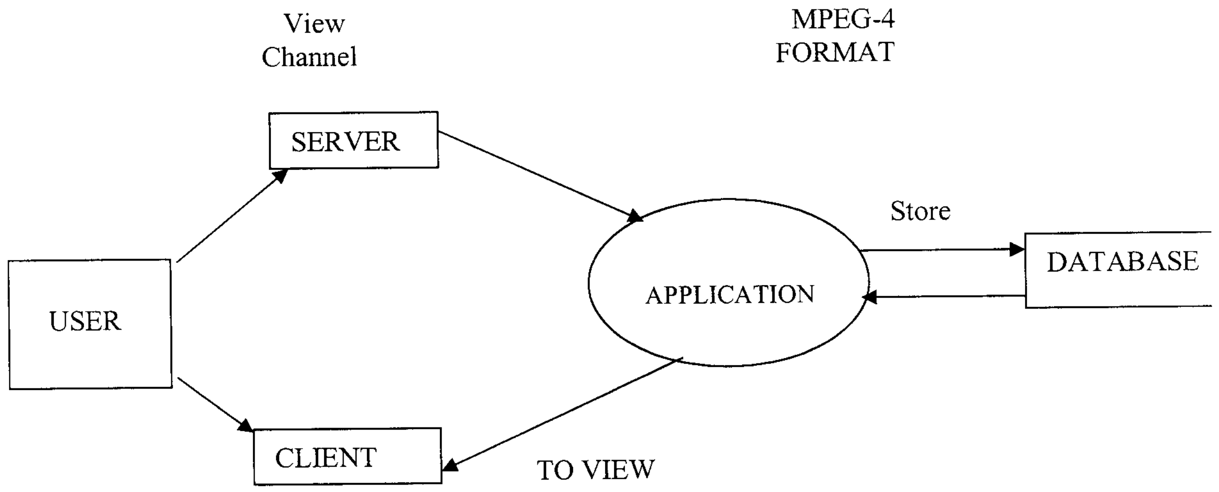
DATA FLOW DIAGRAM FOR SUBSCRIBER MANAGEMENT

LEVEL 1.1: DATA FLOW DIAGRAM



DATA FLOW DIAGRAM FOR SUBSCRIBER MANAGEMENT

Level1.2: DATA FLOW DIAGRAM



DATA FLOW DIAGRAM FOR SUBSCRIBER MANAGEMENT

5. SYSTEM TESTING AND IMPLEMENTATION

5.1 SYSTEM TESTING

System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently before live operation commences. Testing is vital to the success of the system. System testing makes a logical assumption that if all the parts of the system are correct, the goal will be successfully achieved. The candidate system is subject to a variety of tests: on-line response, volume, stress, recovery & security and usability tests.

A series of testing are performed for the proposed system before the system is ready for user acceptance testing steps are:

- Unit testing.
- Integration Testing
- Validation
- Output testing

UNIT TESTING

Unit testing focuses verification efforts on the smallest unit of software design, the module. This is also known as "Module Testing". The modules are tested separately. The testing is carried out during programming stage itself. In this testing step each module is found to be working satisfactorily as regard to the expected output from the module.

INTEGRATION TESTING

Data can be lost across an interface; one module can have an adverse effort on another, sub functions, when combined,, may not produce the desired major functions. Integration testing is a systematic testing for constructing the program structure, while at the same time conducting tests to uncover errors associated within the interface.

The objective is to take unit tested modules and build a program structure. All the modules are combined and tested as a whole. Here correction is difficult because the vast expenses of the entire program complicate the isolation of causes. Thus in the integration testing step, all the errors uncovered are corrected for the next testing steps.

OUTPUT TESTING

After performing the validation testing, the next step is output testing of the proposed system since no system could be useful if it does not produces the required output in the specific format. The outputs generated or displayed by the system under consideration are tested by asking the users about the format required by them. In the case of Subscriber's Management the outputs of the system will be TV channels displayed in the Video capture screen.

The output on the screen is found to be correct as the format was designed in the system design phase according to the user needs. Both the hardware and the software are made to run the developed systems successfully in future.

5.2 SYSTEM IMPLEMENTATION

Implementation is the final and important phase. It involves user training, system testing and successful running of the developed proposed system. The user tests the developed system and changes are made according to their needs. The testing phase involves the testing of developed system using various kinds of data.

An elaborate testing of data is prepared and the system is tested using that test data. While testing, errors are noted and corrections are made. The users are trained to operate the developed system. Both hardware and software securities are made to run the developed system successfully in future.

Implementation is the process of converting a new or received system design into an operational one. It is the key stage in achieving a successful new system because; usually it involves a lot of upheaval in the user department. It must therefore be carefully planned and controlled. Apart from planning a two major tasks for implementation are – education and training of users and testing of the system. Education of users should really take place much earlier in the project i.e. when they are involved in the investigation and design work. Training has to be given to the user regarding the new system.

6. CONCLUSION

The software developed through the project can be used to manage the subscription of subscribed TV channels. Using this software the user can view the available TV channels. The system also provides capturing of program from the channel currently being viewed by the user. It stores the file in a compressed format using the standard MPEG-4 compressor.

7. SCOPE FOR FUTURE DEVELOPMENT

The scope of the project is still extendable. The programming techniques used in the design of the system provide a scope for further expansion, and implementation of any changes, which may occur in the future. In further release of the Subscriber's Management, it can be updated with the functionalities that a commercial TV tuner card will have. The duration of channels viewed by each subscriber could be updated on the remote database.

8. BIBLIOGRAPHY

1. Charlie Petzold, “Programming Windows 95”, Microsoft Press. (144-167)

2. MSDN Library, Edition - October 2001, Microsoft Corporation.

3. Evangelos Petroutsos, “Mastering Visual Basic 6”, SYBEX Inc, May 1998. (229-250)

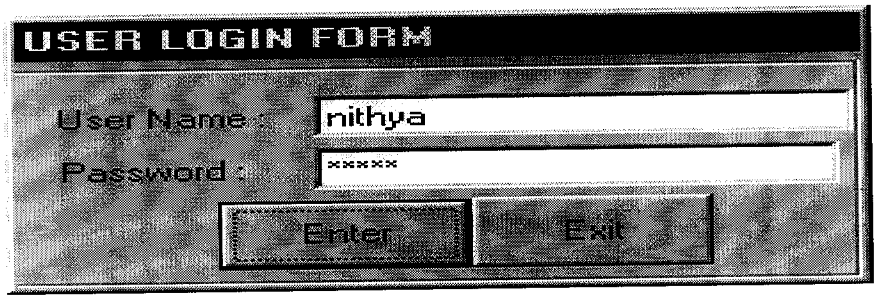
4. Roger S. Pressman, “Software Engineering a Practitioner’s Approach”,

McGraw-Hill International Edition (2001). (105-160)

9. ANNEXURE

9.1 SAMPLE SCREENS

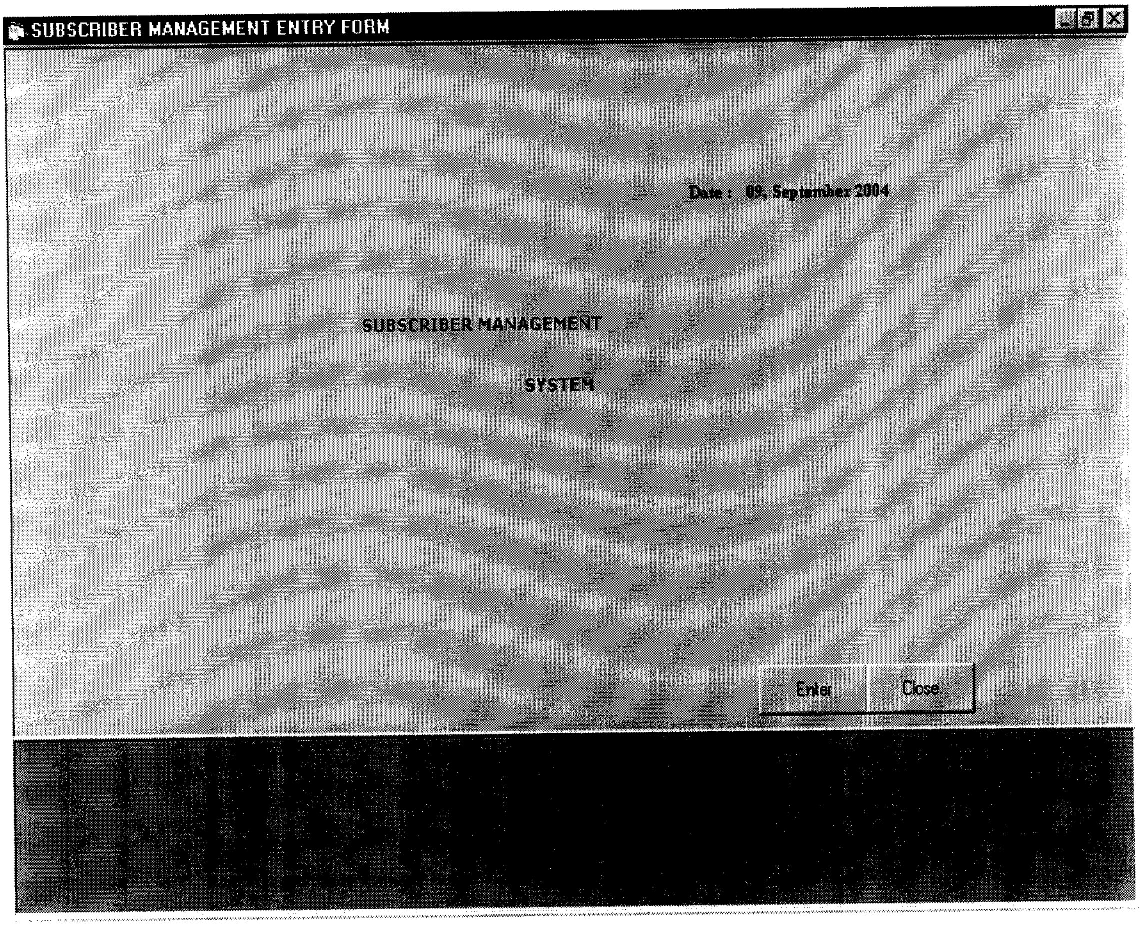
LOGIN FORM


USER LOGIN FORM

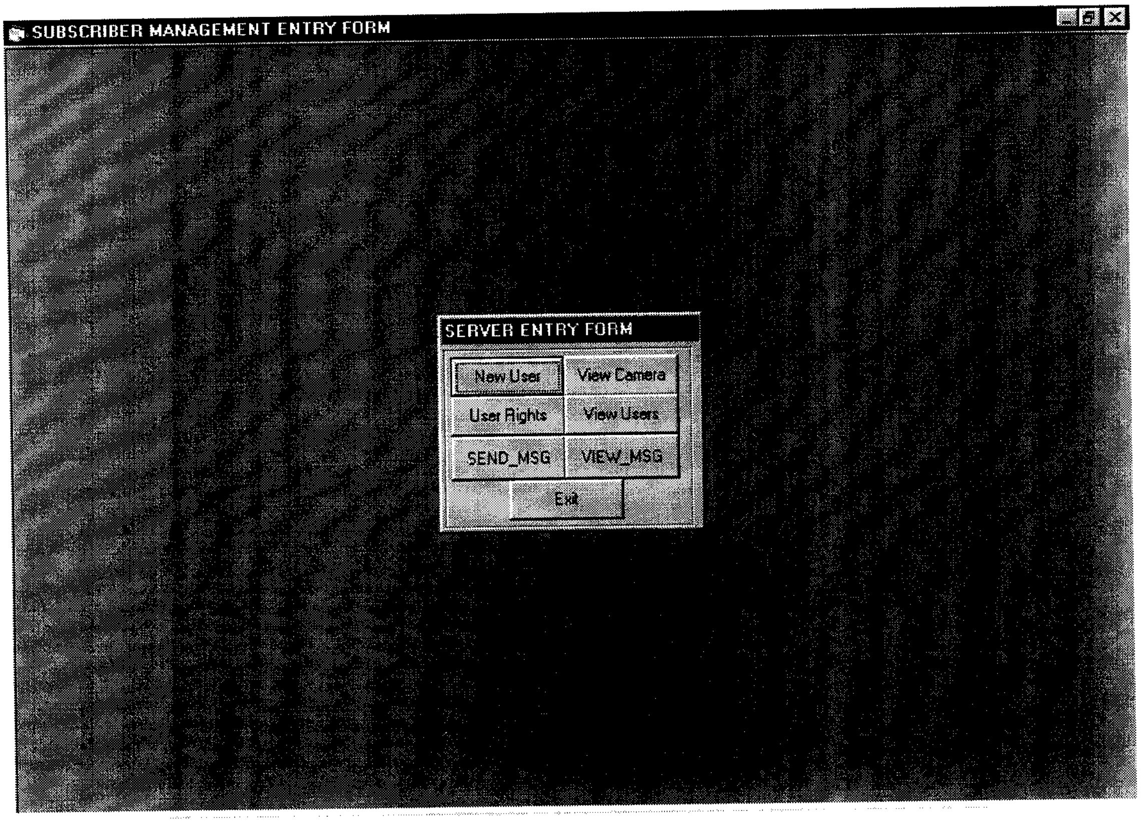
User Name : nithya

Password : ******

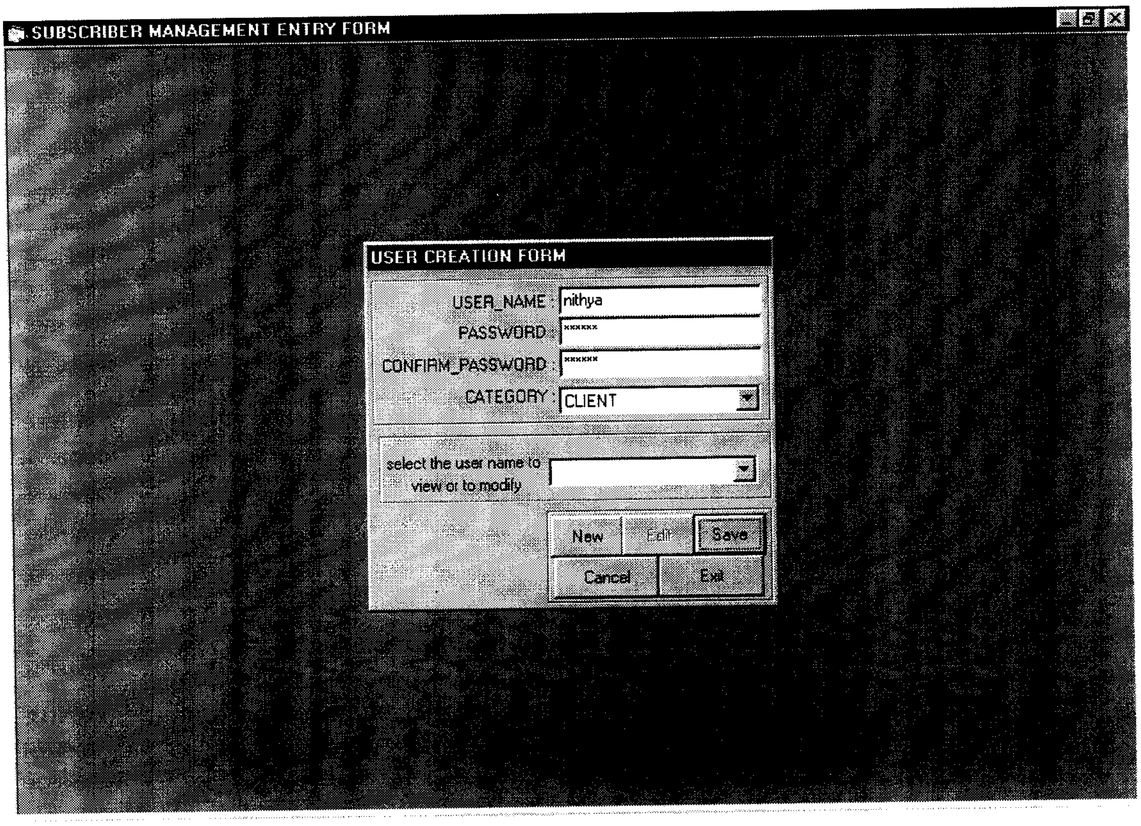
ENTRY FORM


SUBSCRIBER MANAGEMENT ENTRY FORM
Date : 09, September 2004
SUBSCRIBER MANAGEMENT
SYSTEM
Enter
Close

SERVER ENTRY FORM

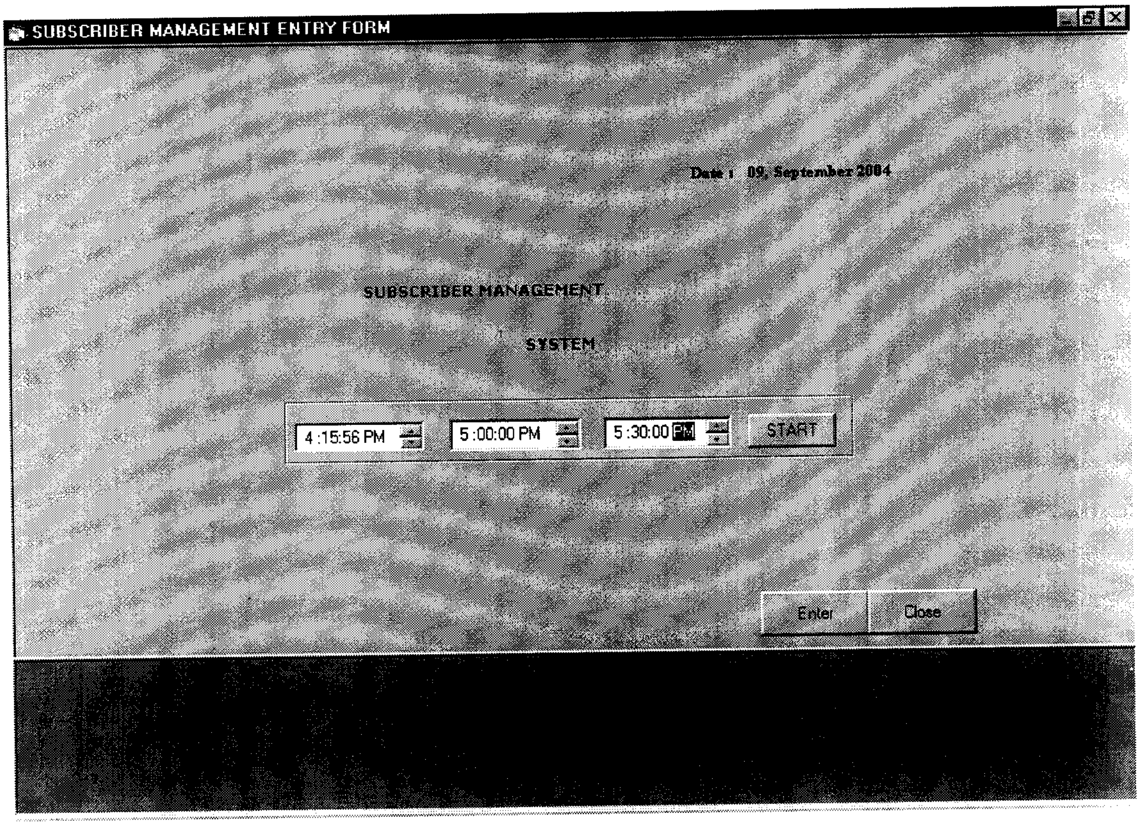


USER CREATION FORM



USER_NAME :	nithya
PASSWORD :	******
CONFIRM_PASSWORD :	******
CATEGORY :	CLIENT
select the user name to view or to modify	

AUTO STORING FORM


SUBSCRIBER MANAGEMENT ENTRY FORM

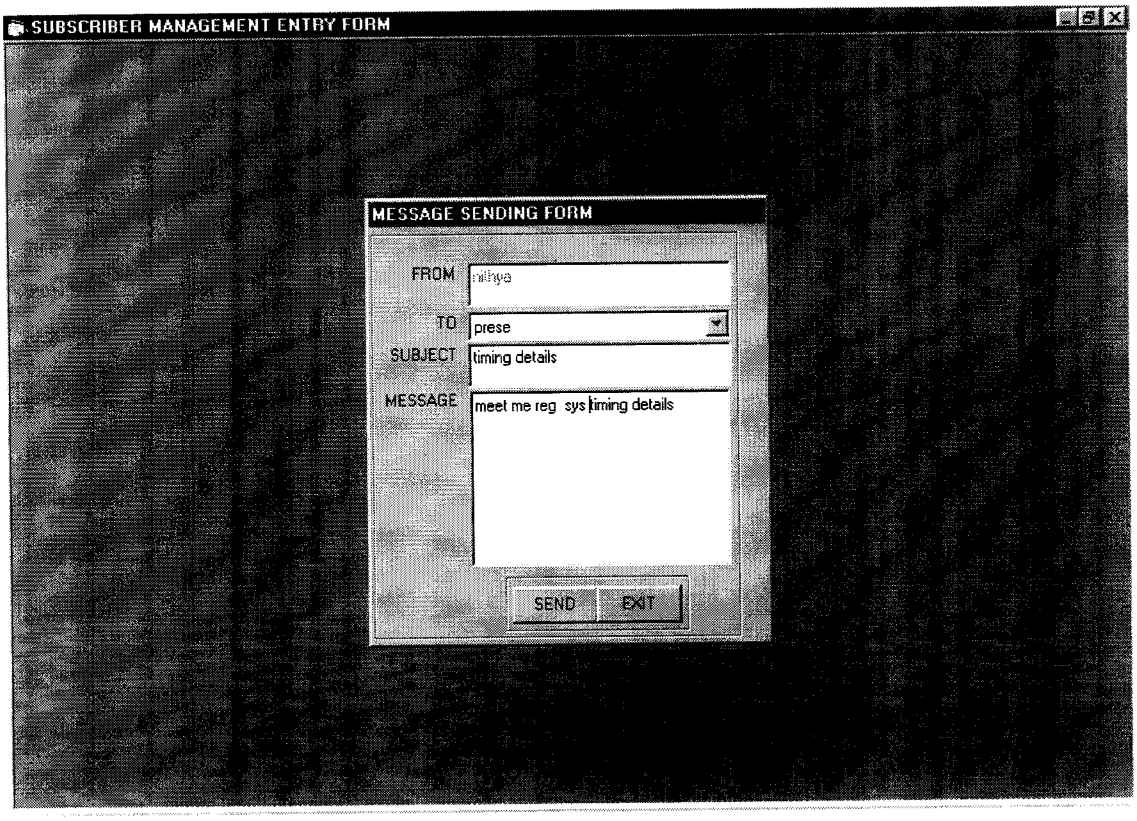
Date : 09, September 2004

SUBSCRIBER MANAGEMENT

SYSTEM

4 :15:56 PM 5 :00:00 PM 5 :30:00 PM

MESSAGE SENDING FORM FOR SERVER



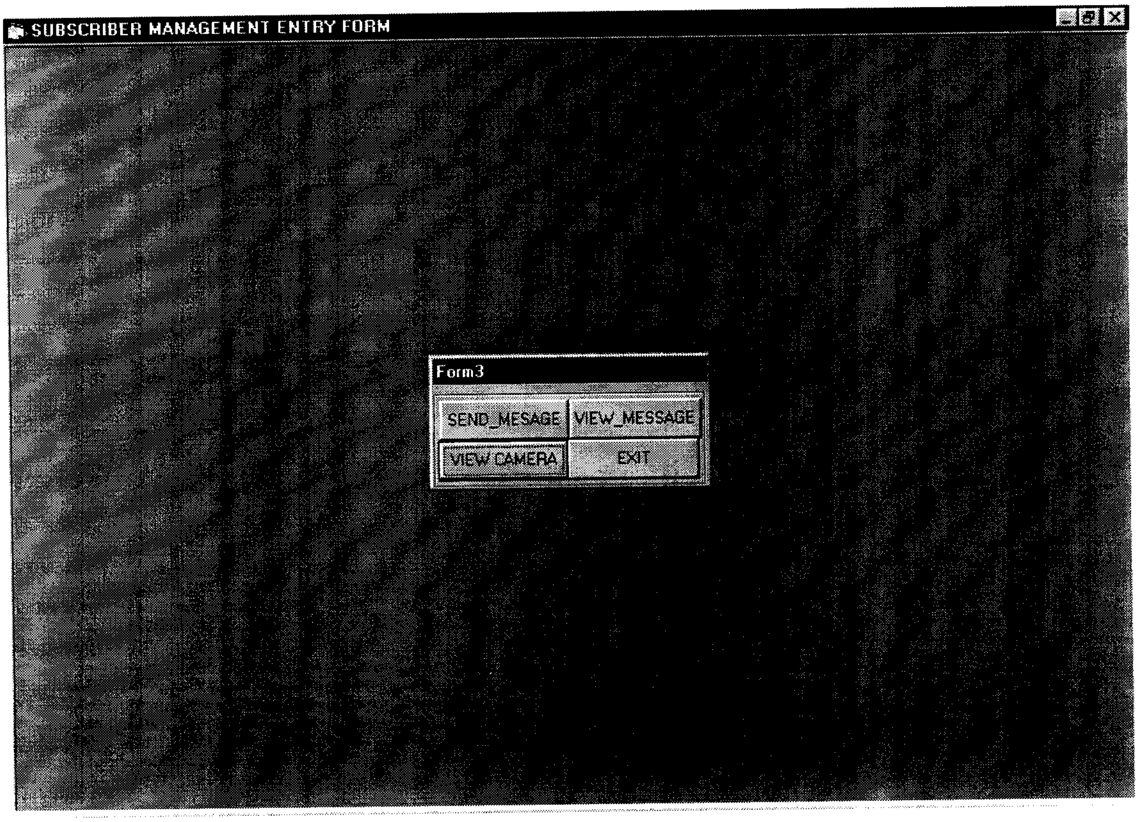
FROM nithya

TO prese

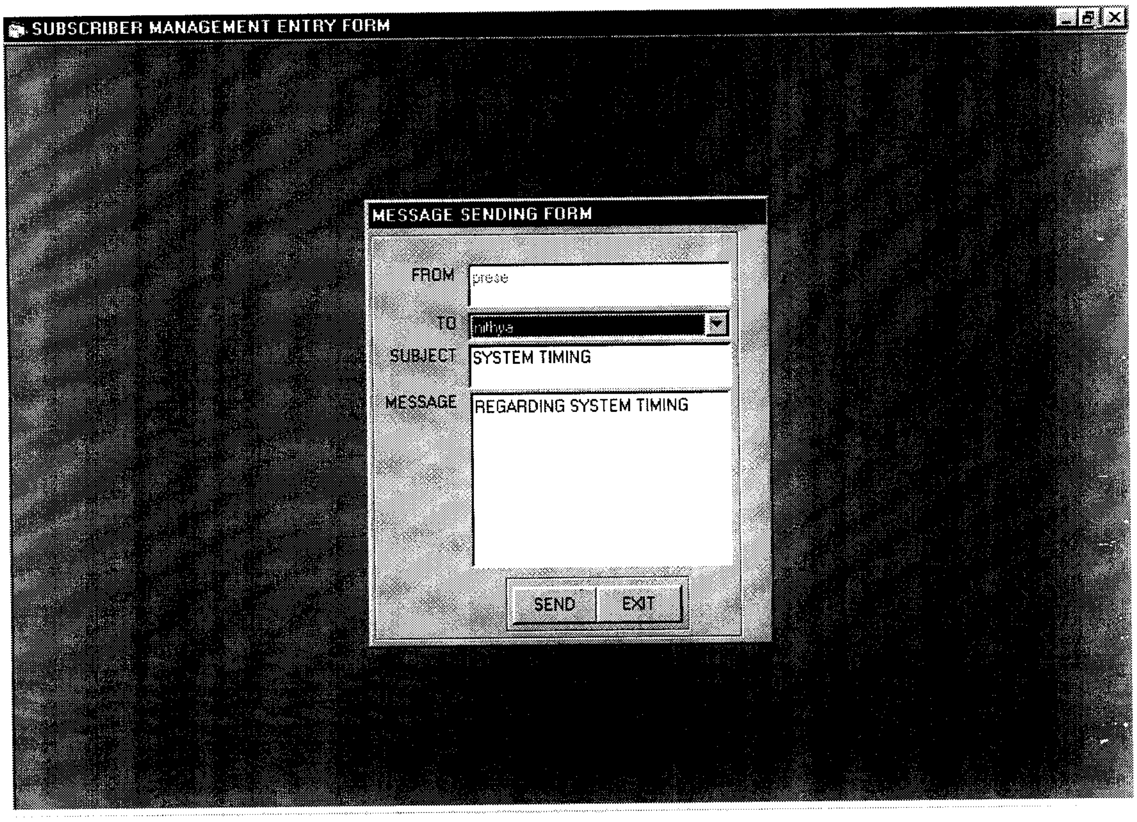
SUBJECT timing details

MESSAGE meet me reg sys timing details

CLIENT ENTRY FORM



MESSAGE SENDIND FORM FOR CLIENTS



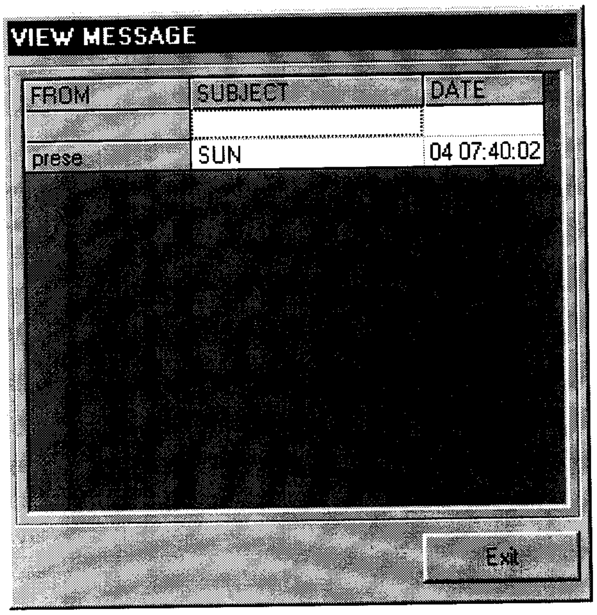
FROM prese

TO nithya

SUBJECT SYSTEM TIMING

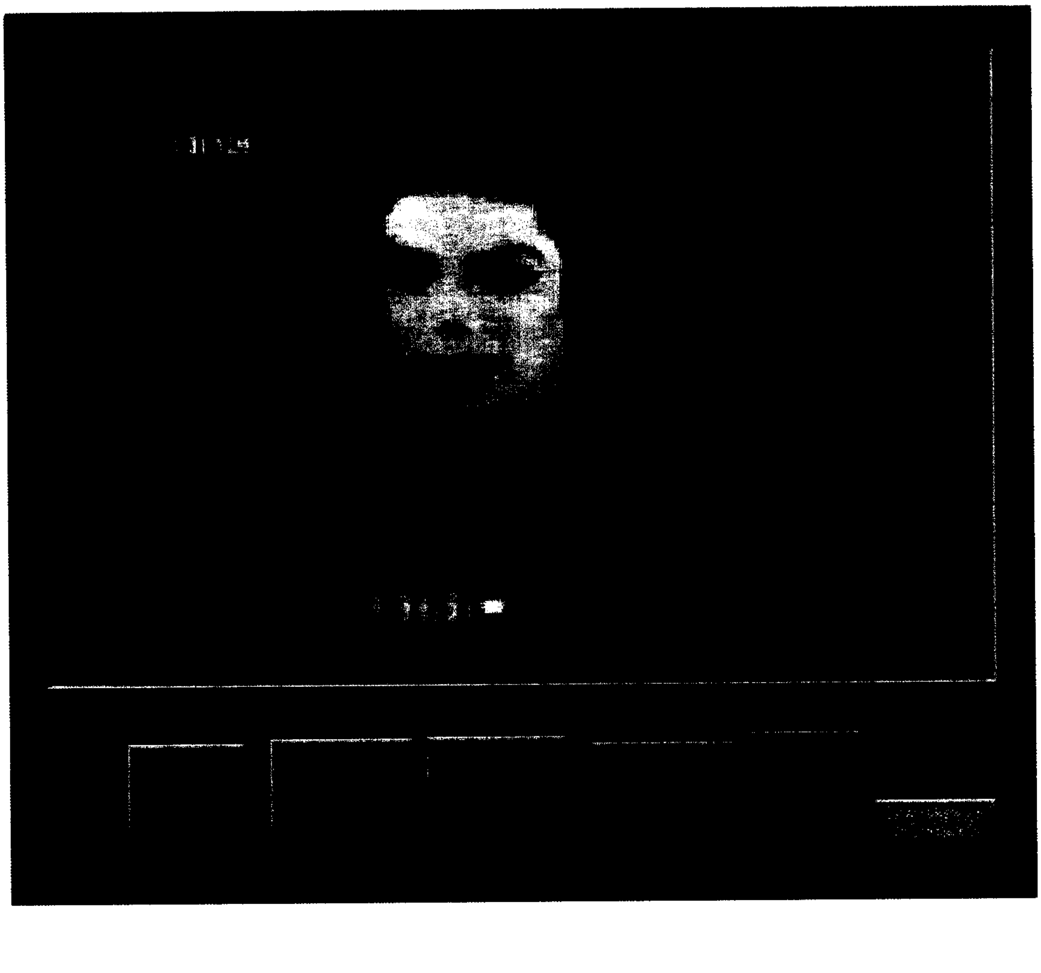
MESSAGE REGARDING SYSTEM TIMING

VIEW MESSAGE FORM



FROM	SUBJECT	DATE
prese	SUN	04 07:40:02

B.B.C WORLD



SAMPLE CODE

This is one of the operational phases in the system development life cycle. This phase deals with coding of the proposed system using the development tools. The programming environment includes the tools provided by the Visual C++. The coding depends on the logical and detailed design of the system in the system design phase. The phase ends in an operational model of the proposed system.

Microsoft Visual C++ provides an Integrated Development Environment (IDE) for developing applications. Win 32 application programming interface (API) functions are used in the program development.

The Direct X SDK is a part of Microsoft Software Development Kit (SDK) .The DirectX is a specialization on the multimedia programming. The basic knowledge of COM is needed to use the DirectX API's.

Win 32 Dynamic Link Libraries programming environment in VC++ is used to create DLL file. Core functionality of the project is implemented as a DLL file.

Microsoft Visual Basic provides an Integrated Development Environment (IDE) for developing applications for Windows Platform. Visual Basic is used to develop the user interface. The whole project is converted into EXE file using Visual Basic. The DLL file created using Visual C++ is declared in Visual Basic which calls the functions as per the user requirement

```
//***********************************************************//
FindCompressor function obtains the required compressor codec into the interface
//***********************************************************
HRESULT FindCompressor(IBaseFilter **pComp)
{
// Local Declaration of HRESULT..... (HRESULT -> Return code of Methods )
HRESULT hr;
// Local Interface Declarations
        IBaseFilter         *pTemp = NULL;
        ICreateDevEnum      *pCDEnum = NULL;
        IEnumMoniker        *pClassEnumMoniker = NULL;
        IMoniker            *pMoniker = NULL;
        IPropertyBag  *pBag = NULL;
// Local Variable declaration to compare the Compressor codec name
        char                 *mystr;
        WCHAR                     CDFriendlyName[120];
        VARIANT                   VarName;
        ULONG                     cFetched;
// Creates an instance of the Interface for the compressor
hr                                                                    =
CoCreateInstance(CLSID_SystemDeviceEnum,NULL,CLSCTX_INPROC_SERVER,II
D_ICreateDevEnum,(void**)&pCDEnum);
// Calls the create class enumerator method on the interface to enumerate the available
video compressor category
hr = pCDEnum-
>CreateClassEnumerator(CLSID_VideoCompressorCategory,&pClassEnumMoniker,0);
// check for the success of the previous method
if (hr == S_OK)
{
// while loop executes till the required compressor codec is obtained
while(pClassEnumMoniker->Next(1,&pMoniker,&cFetched) == S_OK)
```

```
}
// Moniker interface calls the method to bind the obtained compressor to Propertybag
Interface
pMoniker->BindToStorage(0,0,IID_IPropertyBag,(void**)&pBag);
// Initialize the variant varaiable
        VariantInit(&VarName);
        // Declare the Variant variable type
        VarName.vt =  VT_BSTR;
// Property Bag Interface calls the Read Method that reads the firendly name of the codec
into the varaint varaible
pBag->Read(L"FriendlyName",&VarName,NULL);
//Copy the Varaint information into the wide character varaible
lstrcpyW(CDFriendlyName,VarName.bstrVal);
// Initialize local varaible for to check the while condition
int i = 0;
// copy the wide character string name to string varaible
while(CDFriendlyName[i] !='\0')
{
mystr[i] = CDFriendlyName[i];
i++;
}
// Initialize thelast chartacter as null terminated string
mystr[i] = '\0';
// Clear the variant varaible
VariantClear(&VarName);

// Check the required codec name matches the codec obtained from the Interface PBag
if (lstrcmp(mystr,Compressor) == 0)
{
// The Required codec is obtained then bind the codec to the Temp Interface
```

```
hr = pMoniker->BindToObject(NULL,NULL,IID_IBaseFilter,(void**)&pTemp);
// assign the temp interface to Compressor interface

*pComp = pTemp;

// exit from the function

return hr;
}
}
}

return hr;
}

//********************************************************
// FindTVTuner function obtains the TVTuner Codec into the interface
//********************************************************

HRESULT FindTVTuner(IBaseFilter **pTVTuner)
{

// Local Declaration of HRESULT..... (HRESULT -> Return code of Methods )

HRESULT hr;

// Local Interface Declarations

IBaseFilter *pTemp = NULL;
ICreateDevEnum  *pTVTunerEnum =NULL;
IEnumMoniker *pTVTunerClassEnum = NULL;
```

```
IMoniker *pTVTunerMoniker =NULL;

// Local Variable declaration to compare the Compressor codec name

ULONG cFetched;

// Creates an instance of the Interface for the TV Tuner Interface

hr = CoCreateInstance (CLSID_SystemDeviceEnum, NULL, CLSCTX_INPROC,
IID_ICreateDevEnum, (void ** ) &pTVTunerEnum);

if (FAILED(hr))
{
return hr;
}

// Calls the create class enumerator method on the interface to enumerate the available
TV Tuner category


hr = pTVTunerEnum->CreateClassEnumerator (AM_KSCATEGORY_TVTUNER,
&pTVTunerClassEnum, 0);

if (FAILED(hr))
{
return hr;
}

if (pTVTunerClassEnum == NULL)
{
return E_FAIL;
}

// if condition checks the required TV Tuner codec is obtained

if (S_OK == (pTVTunerClassEnum->Next (1, &pTVTunerMoniker, &cFetched)))
{
// The Required codec is obtained then bind the codec to the Temp Interface

hr = pTVTunerMoniker->BindToObject(0,0,IID_IBaseFilter, (void**)&pTemp);

if (FAILED(hr))
{
return hr;
}
}

else
{
return E_FAIL;
}

// assign the temp interface to Compressor interface

*pTVTuner=pTemp;
return hr;}
```

```
//**********************************************************************/
/ Start capture Function starts the recording of the currently displayed channel
//**********************************************************************

BOOL FUNCEXPORT StartCapture(HWND phwnd)
{
// Initialize the global variable with the window handle phwnd

ghwnd = phwnd;

// Local Declaration of HRESULT..... (HRESULT -> Return code of Methods )

HRESULT hr;

// If Capture is true then close all interfaces and restart the filter graph

if (CaptureFlag == TRUE) {CloseInterfaces();}

// CaptureGraphBuilder  function is called to build the filter graph

Flag = CapBuildCaptureGraph();
if (Flag == FALSE)
{
return FALSE;
}

// Create the file that allows to store the recorded file into the specified location

hr = pCGBuilder-
>SetOutputFileName(&MEDIASUBTYPE_Avi,L"C:\\Windows\\Desktop\\TV.avi",&pp
f,&pFileSink);
//RenderStream Method of Capture graph builder reneders the video of TV Channel by
passing
//the required parameters

hr = pCGBuilder-
>RenderStream(&PIN_CATEGORY_CAPTURE,NULL,pSource,Comp,ppf);

//hr = pCGBuilder-
>RenderStream(&PIN_CATEGORY_PREVIEW,NULL,pSource,NULL,NULL);

// pVideoWindow interface sets the videow window owner and style  where the video of
TV channels will be displayed

pVideoWindow->put_Owner((OAHWND)ghwnd);
pVideoWindow->put_WindowStyle(WS_CHILD | WS_CLIPSIBLINGS);

RECT grc;
GetClientRect(ghwnd,&grc);

//  pVideoWindow interface sets the videow window where the video of TV channels will
be displayed

pVideoWindow->SetWindowPosition(0,0,grc.right,grc.bottom);

//pMediacontrol interface is responsible to run the filter graph that allows the whole
functions of the TV Application to execute

pMediaControl->Run();
return TRUE;

}
```

```
//*************************************************************************
// Start Increment preview Function starts the preview based on the incremented channel
number
//*************************************************************************
BOOL FUNCEXPORT StartIncrementPreview(HWND phwnd)
{
// Initialize the global variable with the window handle phwnd

ghwnd = phwnd;

// local varaible declaration

char strchno[5];



if (CHNo > 42)
{
CHNo = 0;
}

// Local Declaration of HRESULT..... (HRESULT -> Return code of Methods )

HRESULT hr;

// CaptureGraphBuilder  function is called to build the filter graph
Flag = BuildCaptureGraph();
if (Flag == FALSE)
{
        return FALSE;
}
```

```
//        wsprintf(strchno,"%d",channelArray[CHNo]);
//MessageBox(NULL,strchno,"The Req CHNo Successfully Collected....",MB_OK);

// put_Channel method of IAMTVTuner interface assigns the channel number to be
displayed

pTV->put_Channel(channelArray[CHNo],NULL,NULL);

// Incerment Channel number

CHNo++;

//RenderStream Method of Capture graph builder reneders the video of TV Channel by
passing
//the required parameters

hr = pCGBuilder-
>RenderStream(&PIN_CATEGORY_PREVIEW,NULL,pSource,NULL,NULL);

// pVideoWindow interface sets the videow window owner and style  where the video of
TV channels will be displayed

pVideoWindow->put_Owner((OAHWND)ghwnd);
pVideoWindow->put_WindowStyle(WS_CHILD | WS_CLIPSIBLINGS);

//pVideoWindow->put_FullScreenMode(OATRUE);

RECT grc;
GetClientRect(ghwnd,&grc);
```

```
//  pVideoWindow interface sets the videow window where the video of TV channels will
be displayed

pVideoWindow->SetWindowPosition(0,0,grc.right,grc.bottom);

//pMediacontrol interface is responsible to run the filter graph that allows the whole
functions of the TV Application to execute

pMediaControl->Run();
return TRUE;
}
//**********************************************************************
```