Cross Platform Media Center

Enrollment No	-	**101207**
Name of Student	-	**Piyush Modi**
Name of Supervisor	-	**Mr. Suman Shah**

May 2014

Submitted in the fulfillment of the Degree of

Bachelor of Technology

Department of Computer Science Engineering

Jaypee University of Information and Technology

Waknaghat, Solan – 173234, H.P

Table of Contents

CERTIFICATE

This is to certify that the work titled “**CROSS PLATFORM MEDIA CENTER**” submitted by “**PIYUSH MODI**” in fulfillment for the award of degree of B.Tech of Jaypee University of Information Technology, Waknaghat has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor	……………………..
Name of Supervisor	……………………..
Designation	……………………..
Date	……………………..

Acknowledgement

I would like to express my greatest gratitude to the people who have helped & supported me throughout my project. I would like to owe my heartiest thanks to my mentor **Mr. Suman Shah** for his continuous support for the project and providing guidelines for making this project.

I would like to thank my group members for contributing their ideas & thoughts and dividing this project into modules to be made individually.

Signature of the student	……………………..
Name of Student	……………………..
Date	……………………..

Abstract

Many Media Centers exist till date, but a lot of them lack some features like extension support for music, video and image files and many of them are not platform independent.

I have made two modules of Media Center i.e, a Video Player Module with support of all the major extensions and a LAN Chat Module. Both of them are platform independent. Both of them can be compiled in most of the available platforms like Windows, Linux, Mac, Solaris etc. Video Player Module consists of a video player with some run-time video effects like colorize, blur effect etc. LAN Chat Module connects the users over a Local Area Network to provide the functionality of chatting. The module can be run in any of the platform.

The project is made by making use of Qt Framework which provides the Cross Platform Functionality.

____________________	____________________
Signature of Student	Signature of Supervisor
Name :	Name :
Date :	Date :

Chapter – 1

Introduction

Cross Platform Media Center

1.1) Introduction

Now a days, the main question that concerns majority of the people is their modes of entertainment after their long hours of hectic careers. They mostly engage themselves in listening to music, watching videos or movies, surfing websites and connecting themselves socially. Seeing the strong desires of our upcoming generations for recreation, many softwares developing companies focused on developing softwares and games that could fulfill the desires of our generations. Progressing further in the development, the idea to aggregate various entertaining stuffs in a single interface evolved. Taking this into consideration the idea of Media Center came up which involved a dedicated media player for various recreational stuffs like listening to music, watching videos or movies, viewing images etc. in a single interface. People had an advantage on these media centers as they had to get acquainted with just a single interface. Also the main motive of media centers was to look more interactive and user friendly so that people could feel the ease of using it.

As more and more development is required in every field, developers started to take views of the general people about what more new features they would want to add to the media centers to make it more recreational. So, the features like Online TV Shows, Chatting, Network streaming etc. were implemented and more media centers with added features were developed and are still being developed. People started liking them as they demand more and more stuffs in a single interface which saves their time and make them easily get acquainted with the interface.

Taking these points into focus, a lot of media centers have been made till date but a lot of them either misses some of the functionality or they are less interactive or have a lot of bugs. Missing functionalities may include lack of supported extensions, lack of some module or less customization. Many media centers are also platform dependent. So, there is still a lot of development required in this field.

1.2) Problem Statement

Many media centers which lack a lot of features in them are lack of extension support and platform independence. Different people use different platforms and so they want media centers of their choice to be platform independent. Making our media center cross platform will make it to run in all the major platforms. Also, we are trying to support most of the extensions in our media center whether it is for music, video or image. We are also adding the modules like LAN Messaging and LAN Video Chat which will make it more entertaining. There is no such media center upto date with such support.

Also, most of the media centers lack audio presets and live video effects. The addition of these functionalities in our media center will make it more entertaining and attractive.

1.3) Objectives

The main objectives of our project are :

- To make a Video Player as a Module of Media Center.
  Player should support all the major video extensions and functionalities such as playlists, live video effects, library management etc.

- To make a LAN Chat Application as a Module of Media Center.
  To allow the users to chat over LAN

1.4) Approach

I am making this project by using Qt Framework. Qt is based on C++ and has a rich APIs for GUIs, Networking, Graphics etc. that is cross-platform. The framework supports all the major platforms which may include Windows, Linux, Mac OS, Symbian, Solaris etc. The development is still going on in Qt and two stable versions are released every year with added features. It can also be ported to Android and iOS but this requires the installation of Qt libraries in the respective platform. The development is still going on to make it portable in these two operating systems natively and the development is expected to be carried out further.

The source code needs to be recompiled at different platforms to make the target Executable. With the rich support of GUI and Graphics libraries, Video Player module will be made by using these libraries. LAN Chatting Module will require the use of network APIs as well.

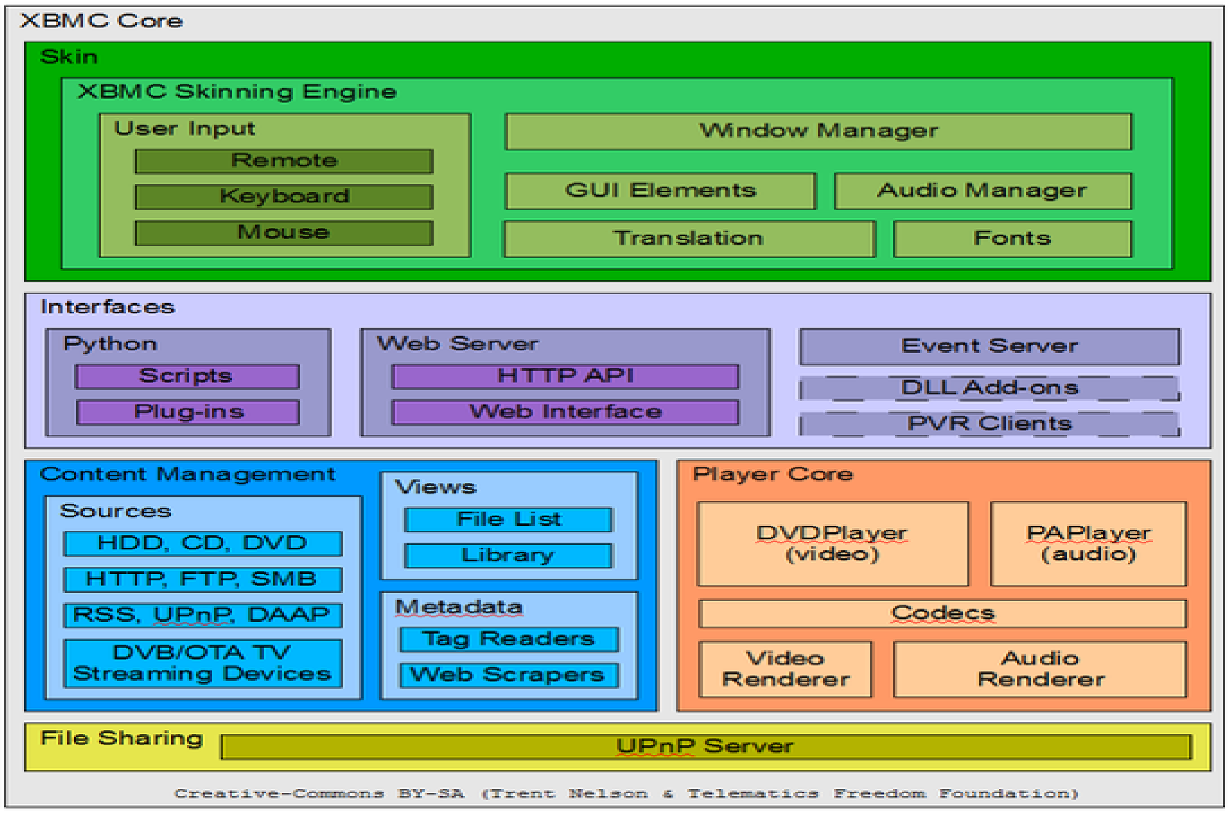
Chapter – 2

Literature Survey

Cross Platform Media Center

Comparisons of some Media Centers

2.1.1). XBMC Media Center :



XMBC Media Center Architecture

XBMC is a free and open source media player developed by the **XBMC Foundation**, a non-profit technology consortium. XBMC is available for multiple operating-systems and hardware platforms, with a user interface for use with televisions and remote controls. It allows users to play and view most videos, music, such as podcasts from the internet, and all common digital media files from local and network storage media.

The software was originally created as a media center application named “Xbox Media Center” for the original Xbox game console, but is today officially available, under the name “*XBMC*”, as a native application for Android, Linux, BSD, Mac OS X, iOS, and Microsoft Windows operating systems.

Features :

1). Audio, video and pictures playback and handling.

2). Live TV with EPG and PVR/DVR Frontend.

3). Add-ons Manager.

4). Supports a large number of platforms and architectures.

Limitations :

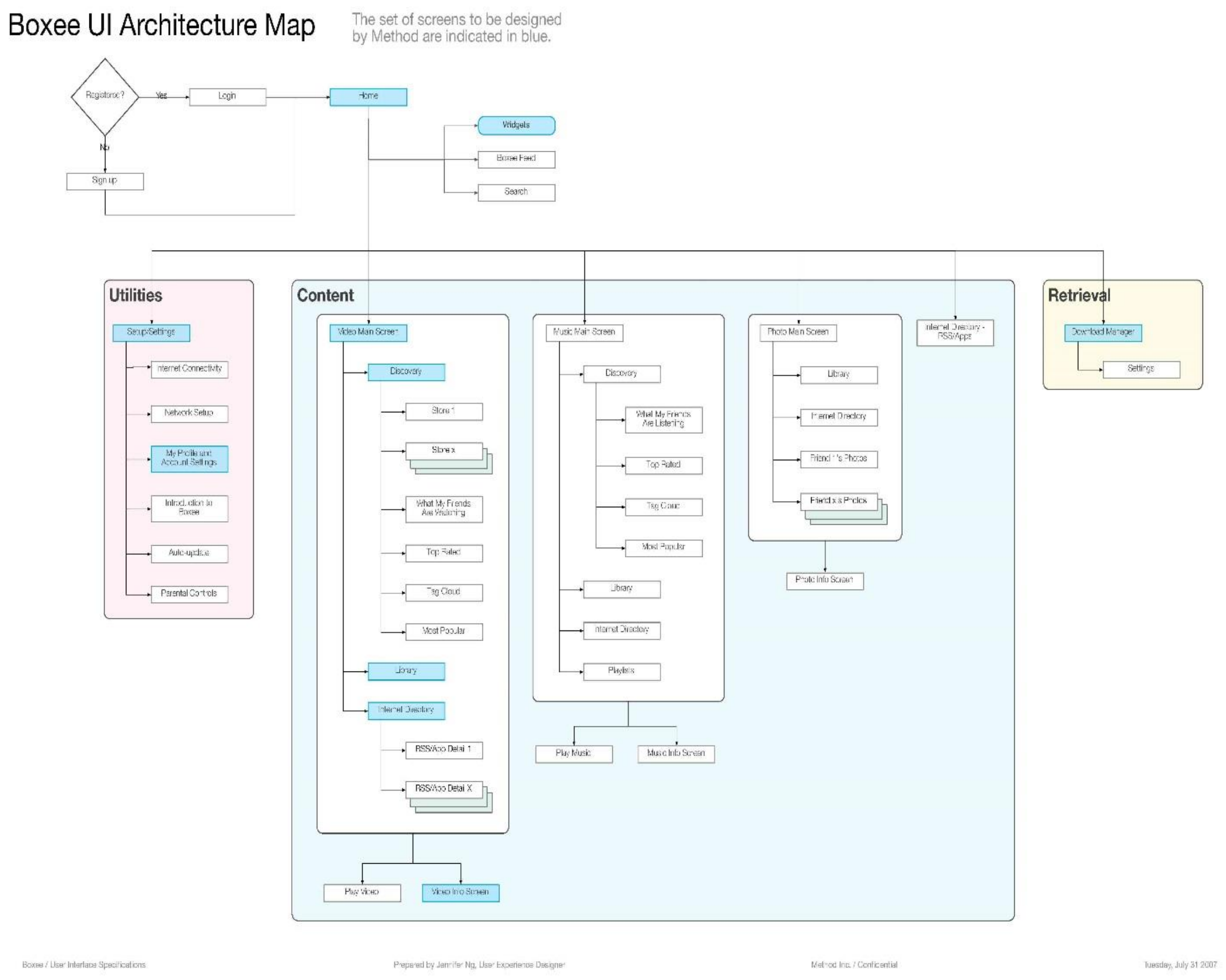
1). XBMC’s own internal cross-platform video and audio players can not play any audio or video files that are protected/encrypted with DRM.

2). It does not support some of the audio and video extensions.

3). TV-tuner support requires use of a third-party plugin.

4). XBMC does not currently support binary addons from third-party developers.

2.1.2) Boxee Media Center:



Boxee is a cross-platform freeware HTPC (Home Theater PC) software application with a 10-foot user interface and social networking features designed for the living-room TV that enables its users to view, rate and recommend content to their friends through many social network services and interactive media related features. Boxee is a media center software which uses XMBC as an application framework for its GUI and media player core platform, together with some custom and proprietary additions.

Features:

1). Audio/video playback and handling.

2). Boxee AppBox Add-on Store and Plugin Apps.

3). Social Networking Layers.

Limitations:

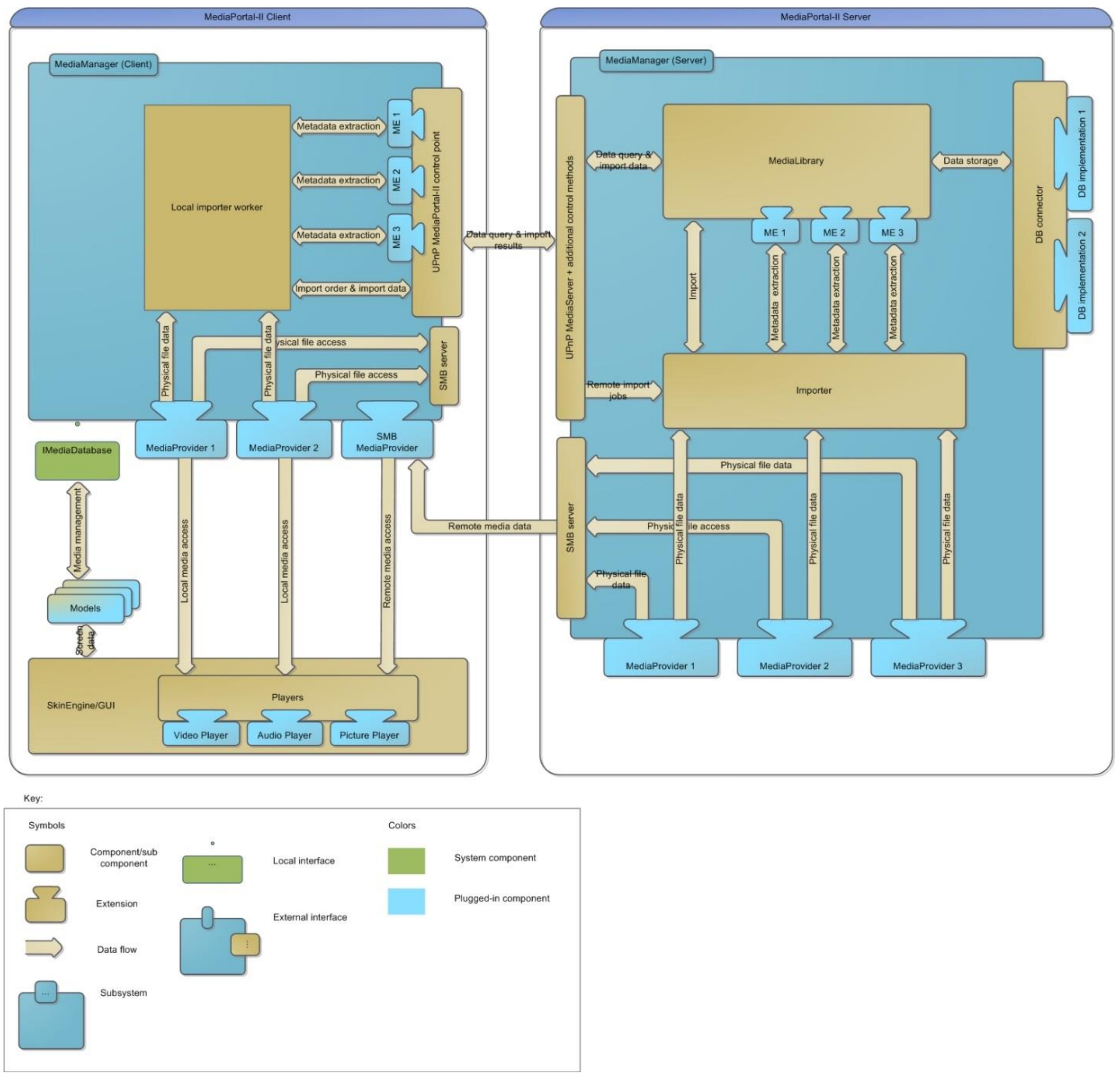
1). Boxee Box does not support Flash Player 11.

2). Boxee cannot play any audio/music files protected/encrypted with Digital rights management (DRM).

3). Boxee is currently available for x86-based platforms, and a x86-64 version is available for Windows. Boxee is not yet available for ARM, PowerPC, or MIPS processor architectures.

4). Boxee requires a DirectX 9.0 or OpenGL 1.4 with GLSL (or newer) hardware accelerated graphics GPU and matching device drivers.

2.1.3) Media Portal



MediaPortal is an open-source media center (HTPC) software project, often considered an alternative to Windows Media Center. It provides a10-foot user interface for performing typical PVR/TiVo functionality, including playing, pausing, and recording live TV; playing

DVDs, videos, and music; viewing pictures; and other functions. Plugins allow it to perform additional tasks, such as watching online video, listening to music from online services such as Last.fm, and launching other applications such as games.

Features:

1). Music, Video and Image playback and handling.

2). Recording, pause and time shifting of TV and Radio broadcasts.

3). Integrated weather forecasts.

4) .Built-in RSS Reader.

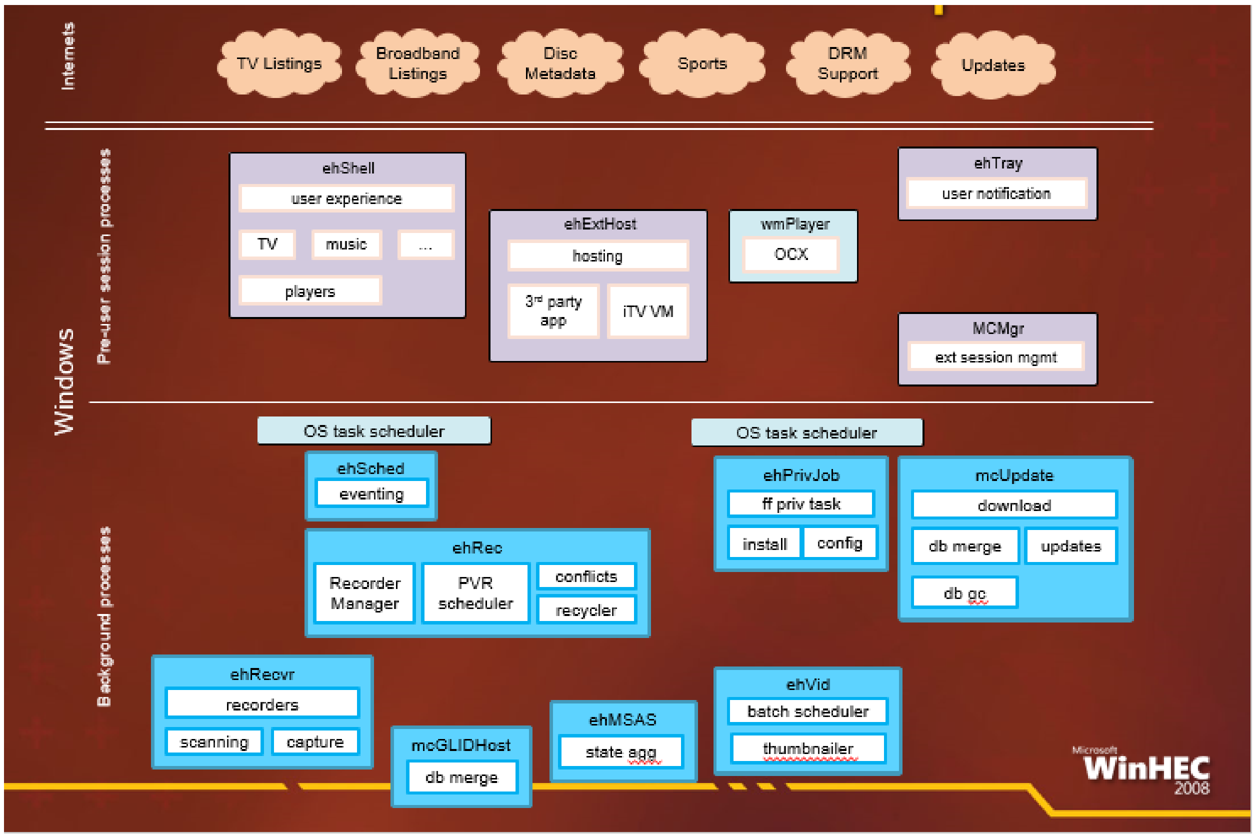
5). Supports all the device controls supported by windows.

Limitations:

1). Designed only for Windows OS.

2). Lacks support of many audio and video extensions.

2.1.4) Windows Media Center



Windows Media Center is a digital video recorder and media player developed by Microsoft. It is an application that allows users to view and record live television, as well as organize and play music and videos. The application is included in various versions of Windows like, Windows Vista Home Premium and Ultimate, and all editions of Windows 7 except for Starter and Home Basic. It is also available for Windows 8 Pro as an add-on.

Windows Media Center can play slideshows, videos and music from local hard drives, optical drives and network locations. Users can stream television programs and films through selected services such as Netflix. Content can be played back on computer monitors or on television sets through the use of devices called Windows Media Center Extenders.

Features:

1). Video, Audio and picture handling and Playback.

2). Windows Media Center organizes and displays videos and music found on both local and networked computers.

3). Windows Media Center allows synchronization with certain portable devices. These devices include Windows Mobile Pocket PCs, smartphones, Portable Media Centers and other players that can sync with Windows Media Player.

Limitations:

1). Lacks support of many audio and video extensions.

2). Poor Performance in High Definition Video Playback.

3). Designed only for Windows.

Qt

2.2.1) What is Qt?

Qt is a full development framework with tools designed to streamline the creation of applications and user interfaces for desktop, embedded and mobile platforms.

Qt Framework contains intuitive C++ API along with QML (CSS/Javascript – like programming) for rapid UI creation.

Qt is a cross-platform application framework that is widely used for developing application software with a graphical user interface (GUI) (in which cases Qt is classified as a *widget toolkit*), and also used for developing non-GUI programs such as command-line tools and consoles for servers.

With Qt, you can reuse code efficiently to target multiple platforms with one code base. The modular C++ class library and developer tools easily enables developers to create applications for one platform and easily build and run to deploy on another platform.

Qt uses standard C++ but makes extensive use of a special code generator(called the *Meta Object Compiler*, or *moc*) together with several macros to enrich the language. Qt can also be used in several other programming languages via language bindings. It runs on the major desktop platforms and some of the mobile platforms. It has extensive internationalization support. Non-GUI features include SQL database access, XML parsing, thread management, network support, and a unified cross-platform application programming interface (API) for file handling.

2.2.2) <u>About Qt:</u>

<u>Original Author :</u> Haavard Nord and Eirik Chambe-Eng

<u>Developer :</u> Qt Project, Digia

<u>Initial Release :</u> 20 May 1995

<u>Stable Release :</u> August 28, 2013; version 5.1.1

<u>Native Language :</u> C++

<u>Platform :</u> Cross-Platform (Windows, Macintosh, Linux, Symbian etc.)

<u>License :</u> LGPL (Open-Source Version), Qt Commercial License (Commercial Qt License)

2.2.3) Applications:

VLC Media Player (Developed by Video Lan Project, 2001)

KDE (Founder – Matthias Ettrech, 1996)

Symbian OS (Developed by Accenture, 1997)

Skype (Developed by Microsoft Skype Division, 2003)

Google Earth (Developed by Google Inc., 2001)

Adobe Photoshop (Developed by Adobe Systems, 1990)

Amazon Kindle (Developed by Amazon, 2007)

AutoDesk Maya (Developed by Autodesk Inc., 1998)

Blackberry (Developed by Blackberry)

2.2.4) Literature Review of Qt:

The Qt Framework was initially developed by Haavard Nord and Eirik Chambe-Eng. They were working on a C++ Database application for ultrasound images that needed to be able to run with a GUI on Unix, Macintosh and Windows. So, they decided to make a object-oriented display system later which came out to be a cross-platform GUI framework.

In 1991, Haavard started writing classes for GUI and collaborated with Erik who came up with the idea of “signals and slots” to make a powerful GUI.

In 1993, Haavard and Erik had developed Qt’s first graphics Kernel and they implemented their own widgets.

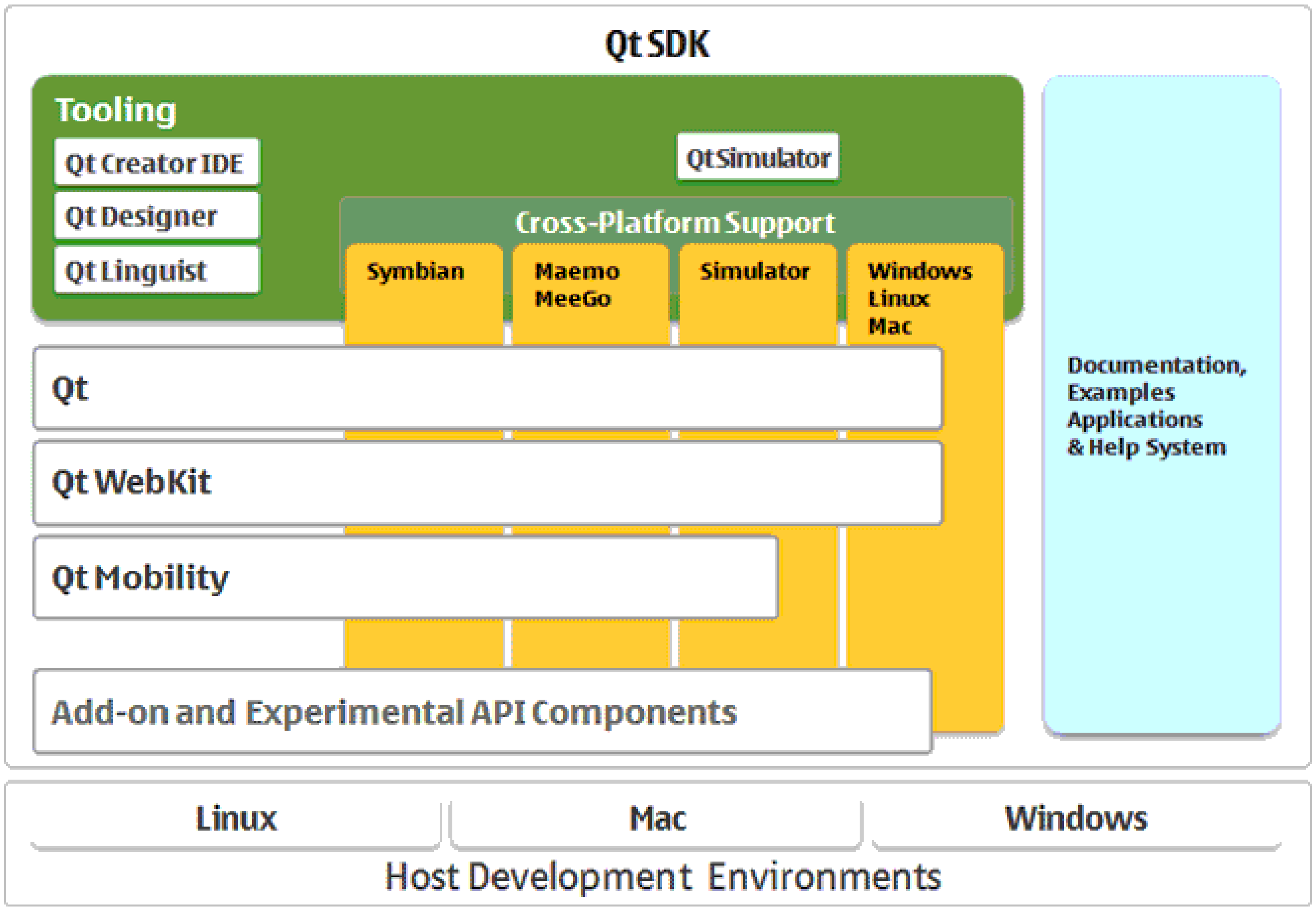
In 1995, these two came up with the name Qt for the framework and company was incorporated as TrollTech in 1995. The first version of Qt was released.

In 2008, Nokia acquired Trolltech and focused on the development for Symbian phones and extended the Qt’s port to Symbian as well.

In September 2012, Digia acquired Qt and focused on further porting it to more platforms.

Currently, the framework is under development to be ported in Android and iOS natively as well.

2.2.5) Architecture Of Qt:



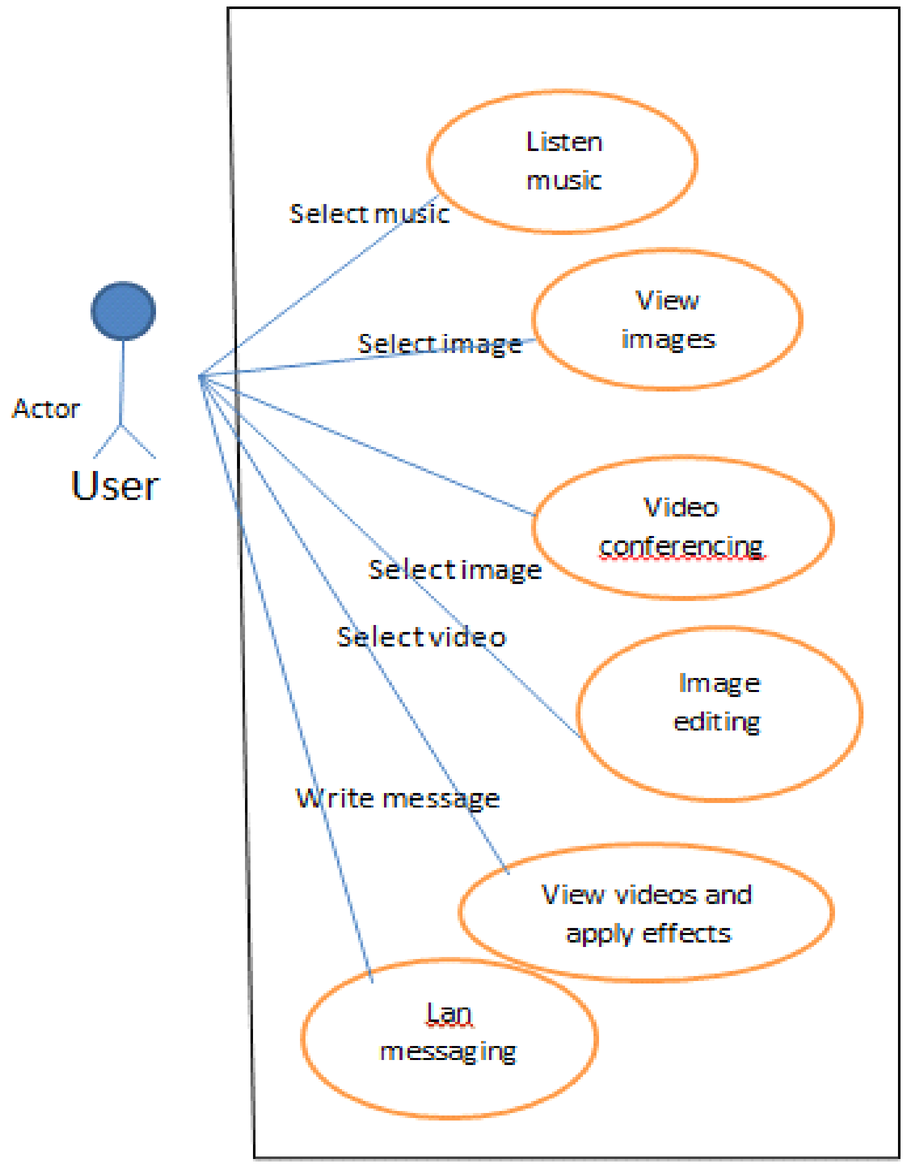
Chapter – 3

Implementation

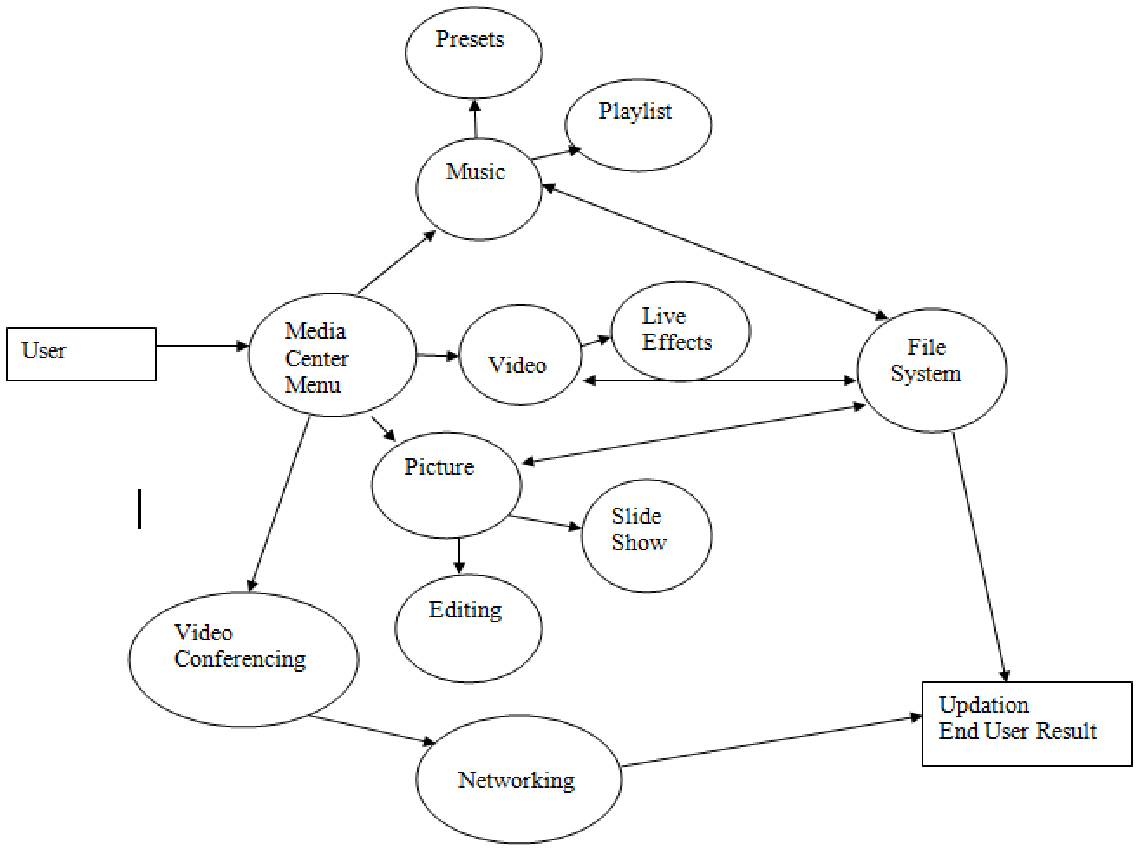
Cross Platform Media Center

Design of Media Center

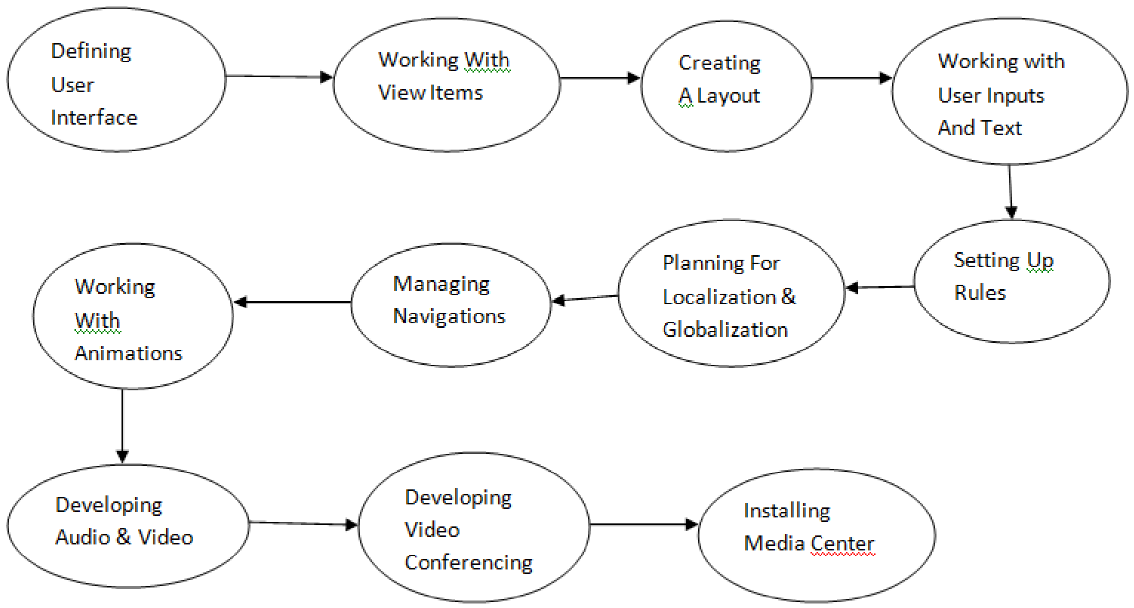
Use Case Diagram



Media center



Level 2 DFD



Process Flow

Modules

3.2.1) Video Module :

One of the module in the Media Center for Video playback and handling and applying live video effects.

Makes use of QMediaPlayer class i.e, a class in Qt for decoding video formats and throws the video onto a Video Widget.

Also supports application of live video effects like Blur Effect and Colorize Effect at runtime.

The Video can be added by making use of file dialog. Multiple files can also be added from the file dialog at once. Drag and drop action is also supported to add files. Also contains a slider to adjust the position of the video playback.

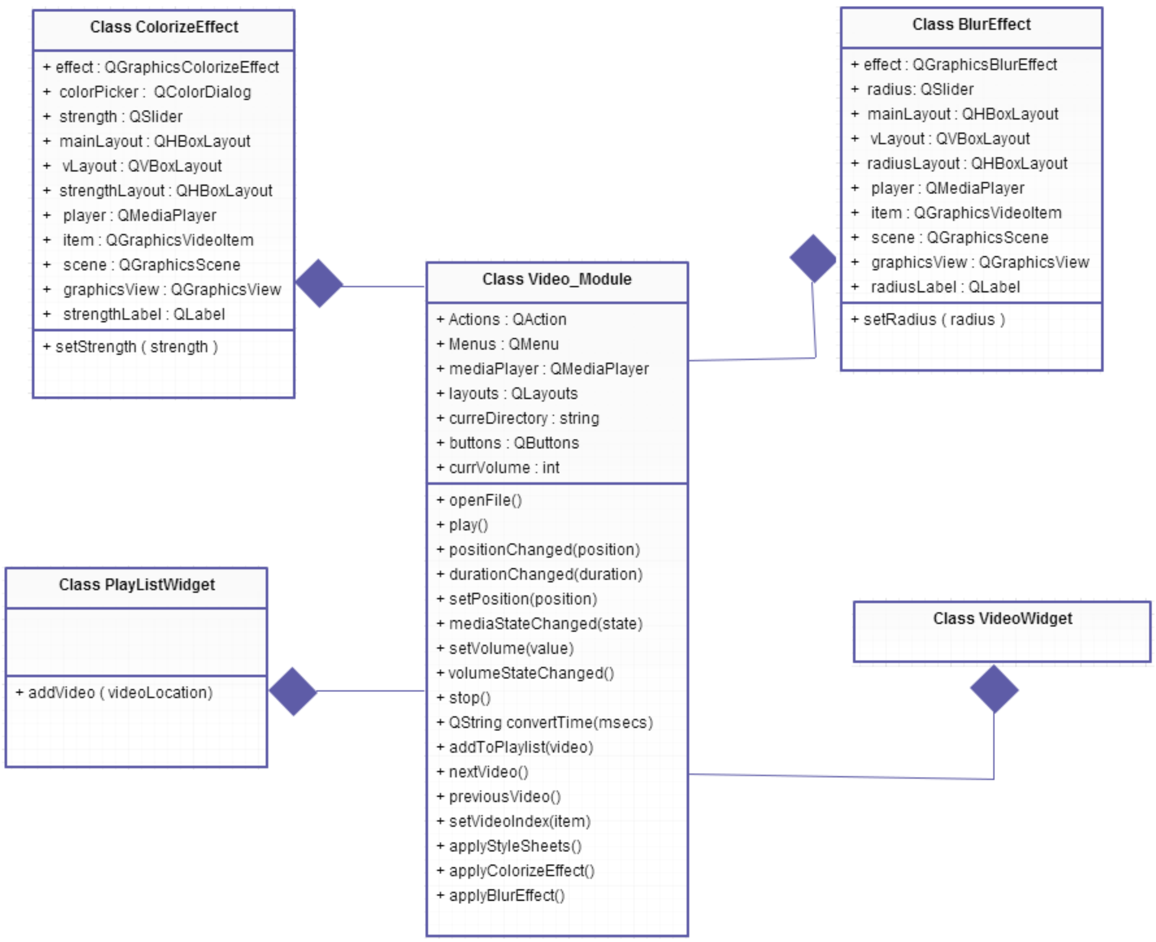
Functionalities :

Extensions Support : Supports all the major video extensions.

Live Video Effect : Live video effcts can be applied like Colorize, Blur, Motion Blur etc. The video is rendered with the applied effect at runtime.

Playlist : Contains a playlist manager with drag and drop actions supported.

Class Diagram :



Code:

1) Video_module.h :

```
#ifndef video_module_H
#define video_module_H

#include<QtWidgets>
#include<QtCore>
#include<QVideoWidget>
#include<QtGui>
#include<qmediaplayer.h>
#include<QMediaPlaylist>


class VideoWidget;
class PlaylistWidget;


class video_module:public QWidget
{
   Q_OBJECT
public:
   video_module();


public slots:
   void openFile();
   void play();
   void positionChanged(qint64 position);
   void durationChanged(qint64 duration);
   void setPosition(int position);
   void mediaStateChanged(QMediaPlayer::State state);
   void setVolume(int value);
   void volumeStateChanged();
   void stop();
   QString convertTime(qint64 msecs);
   void addToPlaylist(QString video);
   void nextVideo();
   void previousVideo();
   void setVideoIndex(QListWidgetItem *item);
   void applyStyleSheets();
   void applyColorizeEffect();
   void applyBlurEffect();
```

```
protected:
    void dropEvent(QDropEvent *event);
    void dragMoveEvent(QDragMoveEvent *event);
    void dragEnterEvent(QDragEnterEvent *event);
    void keyPressEvent( QKeyEvent *event );
    void resizeEvent(QResizeEvent *event);
    void closeEvent(QCloseEvent *event);
    void mouseDoubleClickEvent(QMouseEvent *event);


private:
    void createMenus();
    void createActions();
    void createPlaylistContextMenu();




    QMenuBar *menuBar;

    QMenu *fileMenu;
    QMenu *effectsMenu;
    QMenu *helpMenu;

    QAction *openAction;
    QAction *exitAction;
    QAction *aboutAction;
    QAction *removeAction;

    QAction *colorizeEffect;
    QAction *blurEffect;
    QAction *motionBlurEffect;

    VideoWidget *videoWidget;

    QMediaPlayer *mediaPlayer;

    QMediaPlaylist *currentPlaylist;
    PlaylistWidget *playlistWidget;

    QVBoxLayout *mainLayout;
    QHBoxLayout *controlsLayout;
    QHBoxLayout *volumeLayout;
    QHBoxLayout *positionLayout;
    QGridLayout *playerLayout;
    QWidget *centralWidget;

    QString currDir;

    QPushButton *playButton;
    QPushButton *stopButton;
    QPushButton *volumeButton;
    QPushButton *nextButton;
    QPushButton *previousButton;
    QSlider *videoPosition;
    QSlider *volumeSlider;

    QLabel *playingTime;
    QLabel *totalTime;
    QLabel *playlistLabel;

    QFileDialog *fileDialog;

    QLabel *test;

    int currVolume;



};

#endif // video_module_H
```

2) Video_module.cpp

```
 #include "video_module.h"
#include "playlist_widget.h"
#include "video_widget.h"
#include "video_effects.h"
#include<QtGui>
#include<QtWidgets>




video_module::video_module()
{
```

```
setWindowTitle("Media Center - Video");

test=new QLabel;

menuBar=new QMenuBar;

createActions();
createMenus();

currVolume=100;

videoWidget=new VideoWidget;
videoWidget->setStyleSheet("background:black");
videoWidget->setContentsMargins(0,0,0,0);
videoWidget->setMinimumSize(200,200);


playButton=new QPushButton;
playButton->setIcon(style()->standardIcon(QStyle::SP_MediaPlay));

stopButton=new QPushButton;
stopButton->setIcon(style()->standardIcon(QStyle::SP_MediaStop));

volumeButton=new QPushButton;
volumeButton->setIcon(style()->standardIcon(QStyle::SP_MediaVolume));

nextButton=new QPushButton;
nextButton->setIcon(style()->standardIcon(QStyle::SP_MediaSkipForward));

previousButton=new QPushButton;
previousButton->setIcon(style()->standardIcon(QStyle::SP_MediaSkipBackward));

videoPosition=new QSlider(Qt::Horizontal);
videoPosition->setRange(0,0);


volumeSlider=new QSlider(Qt::Horizontal);
volumeSlider->setRange(0,100);
volumeSlider->setFixedWidth(100);
volumeSlider->setValue(currVolume);


volumeLayout=new QHBoxLayout;
volumeLayout->addWidget(volumeButton);
volumeLayout->addWidget(volumeSlider);
volumeLayout->addStretch();
```

```
    playingTime=new QLabel;
    playingTime->setText("00:00");
    totalTime=new QLabel;
    totalTime->setText("00:00");

    playlistLabel=new QLabel;
    playlistLabel->setText("Current Playlist");
    playlistLabel->setStyleSheet("font-size:12px; color:#333; font-weight:bold; padding:5px 0px
0px 0px");



    controlsLayout=new QHBoxLayout;
    controlsLayout->addWidget(playButton);
    controlsLayout->addWidget(previousButton);
    controlsLayout->addWidget(nextButton);
    controlsLayout->addWidget(stopButton);
    controlsLayout->addSpacing(40);
    controlsLayout->addLayout(volumeLayout);
    controlsLayout->addStretch();
    controlsLayout->setContentsMargins(20,0,20,0);



    positionLayout=new QHBoxLayout;
    positionLayout->addWidget(playingTime);
    positionLayout->addWidget(new QLabel("/"));
    positionLayout->addWidget(totalTime);
    positionLayout->addWidget(videoPosition);
    positionLayout->setContentsMargins(20,0,20,0);

    mediaPlayer=new QMediaPlayer(this);


    currentPlaylist=new QMediaPlaylist(this);
    playlistWidget=new PlaylistWidget(this);


    playerLayout=new QGridLayout;
    playerLayout->addWidget(videoWidget,0,0);
    //playerLayout->addWidget(playlistLabel,0,1,1,1,Qt::AlignCenter);
    playerLayout->addWidget(playlistWidget,0,1);
    playerLayout->setColumnStretch(0,8);
    playerLayout->setColumnStretch(1,2);
    mainLayout=new QVBoxLayout;
    mainLayout->setMenuBar(menuBar);
    mainLayout->addLayout(playerLayout);
    mainLayout->addLayout(positionLayout);
    mainLayout->addLayout(controlsLayout);
    mainLayout->addSpacing(20);
    mainLayout->setMargin(0);



    setLayout(mainLayout);

    mediaPlayer->setVideoOutput(videoWidget);
    mediaPlayer->setPlaylist(currentPlaylist);
    currentPlaylist->setPlaybackMode(currentPlaylist->Loop);
    addToPlaylist("file://F:/Study material/Project
resources/Video_Module/videos/bg_fire.mp4");
    mediaPlayer->play();

    currDir="";

    connect(videoPosition, SIGNAL(sliderMoved(int)),this, SLOT(setPosition(int)));
    connect(playButton, SIGNAL(clicked()),this, SLOT(play()));
    connect(stopButton,SIGNAL(clicked()),this,SLOT(stop()));
    connect(mediaPlayer, SIGNAL(positionChanged(qint64)), this,
SLOT(positionChanged(qint64)));
    connect(mediaPlayer, SIGNAL(durationChanged(qint64)), this,
SLOT(durationChanged(qint64)));
    connect(mediaPlayer, SIGNAL(stateChanged(QMediaPlayer::State)),this,
SLOT(mediaStateChanged(QMediaPlayer::State)));
    connect(volumeSlider,SIGNAL(valueChanged(int)),this,SLOT(setVolume(int)));
    connect(volumeButton,SIGNAL(clicked()),this,SLOT(volumeStateChanged()));
    connect(nextButton,SIGNAL(clicked()),this,SLOT(nextVideo()));
    connect(previousButton,SIGNAL(clicked()),this,SLOT(previousVideo()));

connect(playlistWidget,SIGNAL(itemDoubleClicked(QListWidgetItem*)),this,SLOT(setVideoI
ndex(QListWidgetItem*)));


    fileDialog=new QFileDialog(this);

    setContentsMargins(6,0,6,0);
    setMinimumWidth(600);
    setMinimumHeight(450);
    showMaximized();
    setAcceptDrops(true);
    setWindowIcon(QIcon(":/images/firebolt.ico"));

    applyStyleSheets();


}



void video_module::createActions()
{
    openAction=new QAction(tr("&Open"),this);
    openAction->setIcon(QIcon(":/images/open.jpg"));
    openAction->setStatusTip(tr("Open a Video"));
    connect(openAction,SIGNAL(triggered()),this,SLOT(openFile()));

    exitAction=new QAction(tr("E&xit"),this);
    exitAction->setStatusTip(tr("Exit"));
    exitAction->setIcon(QIcon(":/images/exit.png"));
    connect(exitAction,SIGNAL(triggered()),this,SLOT(close()));

    colorizeEffect=new QAction(tr("Colorize"),this);
    connect(colorizeEffect,SIGNAL(triggered()),this,SLOT(applyColorizeEffect()));

    blurEffect=new QAction(tr("Blur"),this);
    connect(blurEffect,SIGNAL(triggered()),this,SLOT(applyBlurEffect()));

    motionBlurEffect=new QAction(tr("Motion Blur"),this);

    aboutAction=new QAction(tr("About"),this);
    aboutAction->setIcon(QIcon(":/images/firebolt.ico"));

    removeAction=new QAction(tr("Remove Video"),this);

}

void video_module::createMenus()
{
    fileMenu=menuBar->addMenu(tr("&File"));
    fileMenu->addAction(openAction);
    fileMenu->addAction(exitAction);

    effectsMenu=menuBar->addMenu(tr("&Effects"));
    effectsMenu->addAction(colorizeEffect);
    effectsMenu->addAction(blurEffect);
    effectsMenu->addAction(motionBlurEffect);

    helpMenu=menuBar->addMenu(tr("&Help"));
    helpMenu->addAction(aboutAction);

}

void video_module::createPlaylistContextMenu()
{
    playlistWidget->addAction(removeAction);
    playlistWidget->setContextMenuPolicy(Qt::CustomContextMenu);

}

void video_module::applyStyleSheets()
{

}



void video_module::openFile()
{

    fileDialog->setDirectory(QDir::homePath());
    fileDialog->setFileMode(QFileDialog::ExistingFiles);
    fileDialog->setNameFilter(tr("Video Files"));
    QStringList fileNames;
    if(fileDialog->exec())
    {
            int lastIndex=currentPlaylist->mediaCount();
            fileNames=fileDialog->selectedFiles();

            foreach(QString videoName,fileNames)
        if (!videoName.isEmpty())
        {
            addToPlaylist(videoName);

        }
        currentPlaylist->setCurrentIndex(lastIndex);
        playlistWidget->setCurrentRow(lastIndex);
        mediaPlayer->play();
        fileDialog->saveState();
    }

}
```

```
void video_module::play()
{
   switch(mediaPlayer->state()) {
   case QMediaPlayer::PlayingState:
      mediaPlayer->pause();
      break;
   default:
      mediaPlayer->play();

      break;
   }
}

void video_module::stop()
{
   mediaPlayer->stop();
}

void video_module::addToPlaylist(QString video)
{
   currentPlaylist->addMedia(QUrl::fromLocalFile(video));
   playlistWidget->addVideo(video);
}

void video_module::mediaStateChanged(QMediaPlayer::State state)
{
   switch(state) {
   case QMediaPlayer::PlayingState:
      playButton->setIcon(style()->standardIcon(QStyle::SP_MediaPause));
      break;
   default:
      playButton->setIcon(style()->standardIcon(QStyle::SP_MediaPlay));
      break;
   }
}

void video_module::volumeStateChanged()
{
   int volume=mediaPlayer->volume();
   if(volume!=0)
   {
      mediaPlayer->setVolume(0);
      volumeButton->setIcon(style()->standardIcon(QStyle::SP_MediaVolumeMuted));
   }
   else
```

```
    {
        mediaPlayer->setVolume(currVolume);
        volumeButton->setIcon(style()->standardIcon(QStyle::SP_MediaVolume));
    }
}

void video_module::positionChanged(qint64 position)
{
    videoPosition->setValue(position);
    playingTime->setText(convertTime(position));
}

void video_module::durationChanged(qint64 duration)
{
    videoPosition->setRange(0, duration);
    totalTime->setText(convertTime(duration));
}

void video_module::nextVideo()
{
    currentPlaylist->next();
}

void video_module::previousVideo()
{
    currentPlaylist->previous();
}

void video_module::setPosition(int position)
{
    mediaPlayer->setPosition(position);
}

void video_module::setVolume(int value)
{
    mediaPlayer->setVolume(value);
    currVolume=value;
}

void video_module::dropEvent(QDropEvent *event)
{
    event->accept();
    QUrl url;
    foreach(url,event->mimeData()->urls())
    {
    addToPlaylist(url.toLocalFile());
    }
    mediaPlayer->play();
    playlistWidget->setCurrentRow(currentPlaylist->currentIndex());
}
void video_module::dragEnterEvent(QDragEnterEvent *event)
{
    event->accept();
}
void video_module::dragMoveEvent(QDragMoveEvent *event)
{
    event->accept();
}
void video_module::keyPressEvent(QKeyEvent * event)
{
    if(event->key()==Qt::Key_F && mediaPlayer->state()==QMediaPlayer::PlayingState &&
!videoWidget->isFullScreen())
    {
        videoWidget->setFullScreen(true);
        this->hide();
    }
}

void video_module::mouseDoubleClickEvent(QMouseEvent *)
{
    videoWidget->setFullScreen(true);
    this->hide();
}

void  video_module::closeEvent(QCloseEvent *event)
{
    event->accept();
    delete playlistWidget;
    close();
}

void video_module::resizeEvent(QResizeEvent *event)
{
    event->accept();
}

QString video_module::convertTime(qint64 msecs)
{
    QString formattedTime;


        qint64 minutes = (msecs)/(1000*60);
        qint64 seconds = (msecs-(minutes*1000*60))/1000;
        formattedTime.append(QString( "%1" ).arg(minutes, 2, 10, QLatin1Char('0')) + ":" +
                        QString( "%1" ).arg(seconds, 2, 10, QLatin1Char('0')));
        return formattedTime;

}

void video_module::setVideoIndex(QListWidgetItem *item)
{
    currentPlaylist->setCurrentIndex(playlistWidget->row(item));
}

void video_module::applyColorizeEffect()
{
    this->play();
    ColorizeEffect *effect=new ColorizeEffect(mediaPlayer-
>currentMedia().canonicalUrl().toDisplayString());
    connect(effect,SIGNAL(destroyed()),this,SLOT(play()));
}
void video_module::applyBlurEffect()
{
    this->play();
    BlurEffect *effect=new BlurEffect(mediaPlayer-
>currentMedia().canonicalUrl().toDisplayString());
    connect(effect,SIGNAL(destroyed()),this,SLOT(play()));
}
```

3) Playlist_widget.h

```
#ifndef PLAYLIST_WIDGET_H
#define PLAYLIST_WIDGET_H


#include<QtWidgets>

class PlaylistWidget:public QListWidget
{
    Q_OBJECT
public:
    PlaylistWidget(QWidget *parent=0);
    void addVideo(QString video);
};
```

```
#endif // PLAYLIST_WIDGET_H
```

4) Playlist_widget.cpp

```
 #include "playlist_widget.h"

PlaylistWidget::PlaylistWidget(QWidget *parent):QListWidget(parent)
{

   setMinimumWidth(200);
   setAlternatingRowColors(true);
   setStyleSheet("QListWidget::item{padding:20px;}");


}

void PlaylistWidget::addVideo(QString video)
{
   QFileInfo fileInf(QUrl::fromLocalFile(video).toString());
   QString fileName = fileInf.fileName();
   QListWidgetItem *newVideoItem=new QListWidgetItem;
   newVideoItem->setText(fileName);
   newVideoItem->setIcon(QIcon(":/images/firebolt.ico"));
   addItem(newVideoItem);
}
```

5) Video_effects.h

```
 #ifndef VIDEO_EFFECTS_H
#define VIDEO_EFFECTS_H

#include<QtGui>
#include<QtWidgets>
#include<QtMultimedia>
#include<QtMultimediaWidgets>


class ColorizeEffect:public QWidget
{
   Q_OBJECT
public:
   ColorizeEffect(QString FileName, QWidget *parent=0);
```

```
public slots:
   void setStrength(int strength);
   void closeEvent(QCloseEvent *);

private:
   QGraphicsColorizeEffect *effect;
   QColorDialog *colorPicker;
   QSlider *strength;
   QHBoxLayout *mainLayout;
   QVBoxLayout *vLayout;
   QHBoxLayout *strengthLayout;
   QMediaPlayer *player;
   QGraphicsVideoItem *item;
   QGraphicsScene *scene;
   QGraphicsView *graphicsView;
   QLabel *strengthLabel;

};

/****************************************************************************
***/


class BlurEffect:public QWidget
{
   Q_OBJECT
public:
   BlurEffect(QString FileName, QWidget *parent=0);

public slots:
   void setRadius(int strength);
   void closeEvent(QCloseEvent *);

private:
   QGraphicsBlurEffect *effect;
   QSlider *radius;
   QHBoxLayout *mainLayout;
   QVBoxLayout *vLayout;
   QHBoxLayout *radiusLayout;
   QMediaPlayer *player;
   QGraphicsVideoItem *item;
   QGraphicsScene *scene;
   QGraphicsView *graphicsView;
   QLabel *radiusLabel;
```

```
};

#endif // VIDEO_EFFECTS_H
```

6) Video_effects.cpp

```
 #include "video_effects.h"

ColorizeEffect::ColorizeEffect(QString FileName,QWidget *parent):QWidget(parent)
{
    colorPicker=new QColorDialog;
    colorPicker->setOption(QColorDialog::NoButtons);
    strength=new QSlider(Qt::Horizontal);

    mainLayout=new QHBoxLayout;
    vLayout=new QVBoxLayout;
    strengthLayout=new QHBoxLayout;
    player = new QMediaPlayer(this);

    item = new QGraphicsVideoItem;
    scene=new QGraphicsScene;
    graphicsView=new QGraphicsView(scene);
    graphicsView->setStyleSheet("background:black");
    graphicsView->setFrameShape(QFrame::NoFrame);

    effect=new QGraphicsColorizeEffect;
    effect->setColor(QColor(255,0,0));

    strengthLabel=new QLabel;
    strengthLabel->setText("Strength: ");
    strengthLabel->setStyleSheet("font-weight:bold; color:#333");

    scene->addItem(item);
    player->setVideoOutput(item);
    player->setMedia(QUrl::fromLocalFile(FileName));

    item->setSize(QSizeF(800,600));
    item->setGraphicsEffect(effect);

    strength->setRange(0,100);
    strength->setValue(20);
    effect->setStrength(0.2);

    strengthLayout->addWidget(strengthLabel);
    strengthLayout->addWidget(strength);
    mainLayout->addWidget(graphicsView);
    vLayout->addWidget(colorPicker);
    vLayout->addLayout(strengthLayout);
    mainLayout->addLayout(vLayout);

    setLayout(mainLayout);
    setContentsMargins(0,0,0,0);


connect(colorPicker,SIGNAL(currentColorChanged(QColor)),effect,SLOT(setColor(QColor)));
    connect(strength,SIGNAL(sliderMoved(int)),this,SLOT(setStrength(int)));

    showMaximized();
    player->play();

}

void ColorizeEffect::setStrength(int strength)
{
    effect->setStrength(qreal(strength)/100);
}

void ColorizeEffect::closeEvent(QCloseEvent *)
{
    delete this;
}


/*****************************************************************************
***********/


BlurEffect::BlurEffect(QString FileName,QWidget *parent):QWidget(parent)
{

    radius=new QSlider(Qt::Horizontal);

    mainLayout=new QHBoxLayout;
    vLayout=new QVBoxLayout;
    radiusLayout=new QHBoxLayout;
    player = new QMediaPlayer(this);

    item = new QGraphicsVideoItem;
    scene=new QGraphicsScene;
    graphicsView=new QGraphicsView(scene);
```

```
   graphicsView->setStyleSheet("background:black");
   graphicsView->setFrameShape(QFrame::NoFrame);

   effect=new QGraphicsBlurEffect;


   radiusLabel=new QLabel;
   radiusLabel->setText("Blur Radius: ");
   radiusLabel->setStyleSheet("font-weight:bold; color:#333");

   scene->addItem(item);
   player->setVideoOutput(item);
   player->setMedia(QUrl::fromLocalFile(FileName));

   item->setSize(QSizeF(800,600));
   item->setGraphicsEffect(effect);

   radius->setRange(0,100);
   radius->setValue(5);
   effect->setBlurRadius(5);

   radiusLayout->addWidget(radiusLabel);
   radiusLayout->addWidget(radius);
   mainLayout->addWidget(graphicsView);
   vLayout->addLayout(radiusLayout);
   mainLayout->addLayout(vLayout);

   setLayout(mainLayout);
   setContentsMargins(0,0,0,0);

   connect(radius,SIGNAL(sliderMoved(int)),this,SLOT(setRadius(int)));

   showMaximized();
   player->play();

}

void BlurEffect::setRadius(int strength)
{
   effect->setBlurRadius(strength);
}

void BlurEffect::closeEvent(QCloseEvent *)
{
   delete this;
}
```

7) Video_widget.h

```
  #ifndef VIDEO_WIDGET_H
#define VIDEO_WIDGET_H

#include<QtWidgets>
#include<QtCore>
#include<QVideoWidget>
#include<QtGui>

class VideoWidget:public QVideoWidget
{
   Q_OBJECT
public:
   VideoWidget(QWidget *parent=0);

protected:
   void keyPressEvent( QKeyEvent * event );
   void mouseDoubleClickEvent(QMouseEvent *event);

};


#endif // VIDEO_WIDGET_H
```

8) Video_widget.cpp

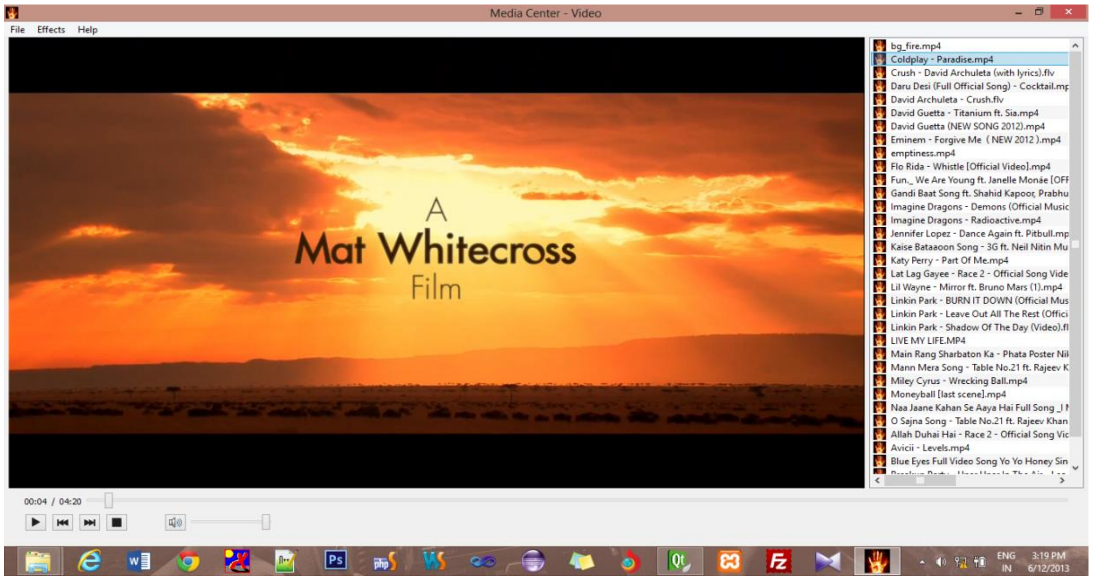
```
 #include "video_widget.h"

VideoWidget::VideoWidget(QWidget *parent):QVideoWidget(parent)
{

}



void VideoWidget::keyPressEvent(QKeyEvent *event)
{
   if(event->key()==Qt::Key_F && this->isFullScreen())
   {
      this->setFullScreen(false);
      this->showNormal();
      this->parentWidget()->show();
   }
```

```
    else if(event->key()==Qt::Key_F && !this->isFullScreen())
    {
        this->setFullScreen(true);
        this->parentWidget()->hide();
    }

}

void VideoWidget::mouseDoubleClickEvent(QMouseEvent *event)
{
    event->accept();
    if(this->isFullScreen())
    {
        this->setFullScreen(false);
        this->showNormal();
        this->parentWidget()->show();
    }
    else
    {
        this->setFullScreen(true);
        this->parentWidget()->hide();
    }
}
```

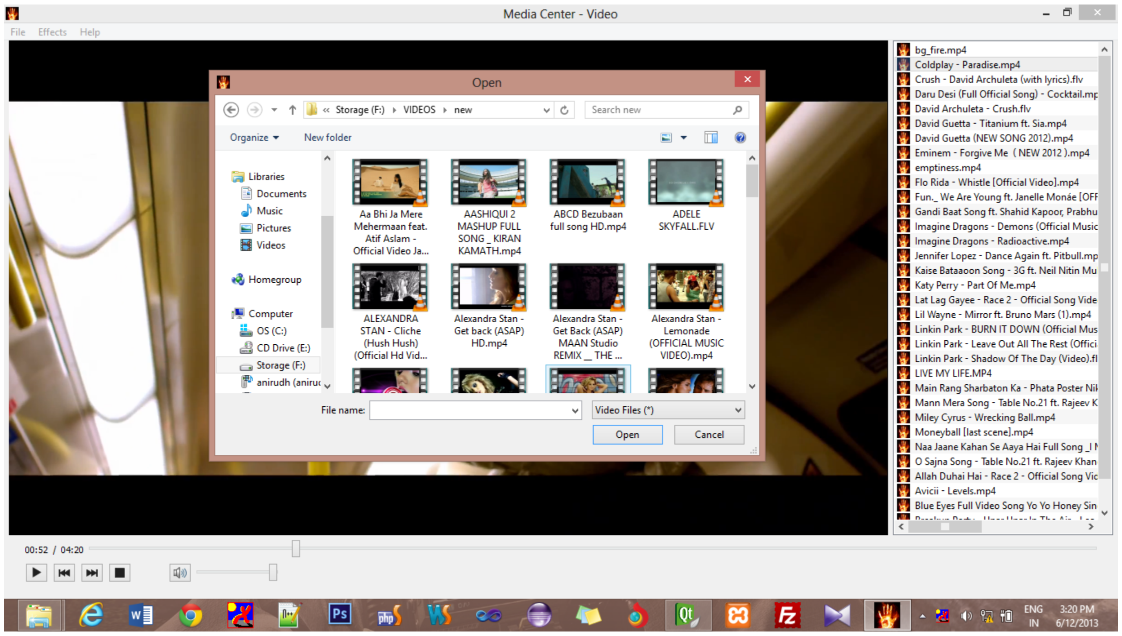
9) Main.cpp

```
#include "video_module.cpp"
#include<QtCore>



int main(int argc, char **argv)
{
    QApplication a(argc,argv);
    a.setApplicationName("FireBolt Media Center");
    a.setOrganizationName("BT");
    video_module s1;

    return a.exec();
}
```

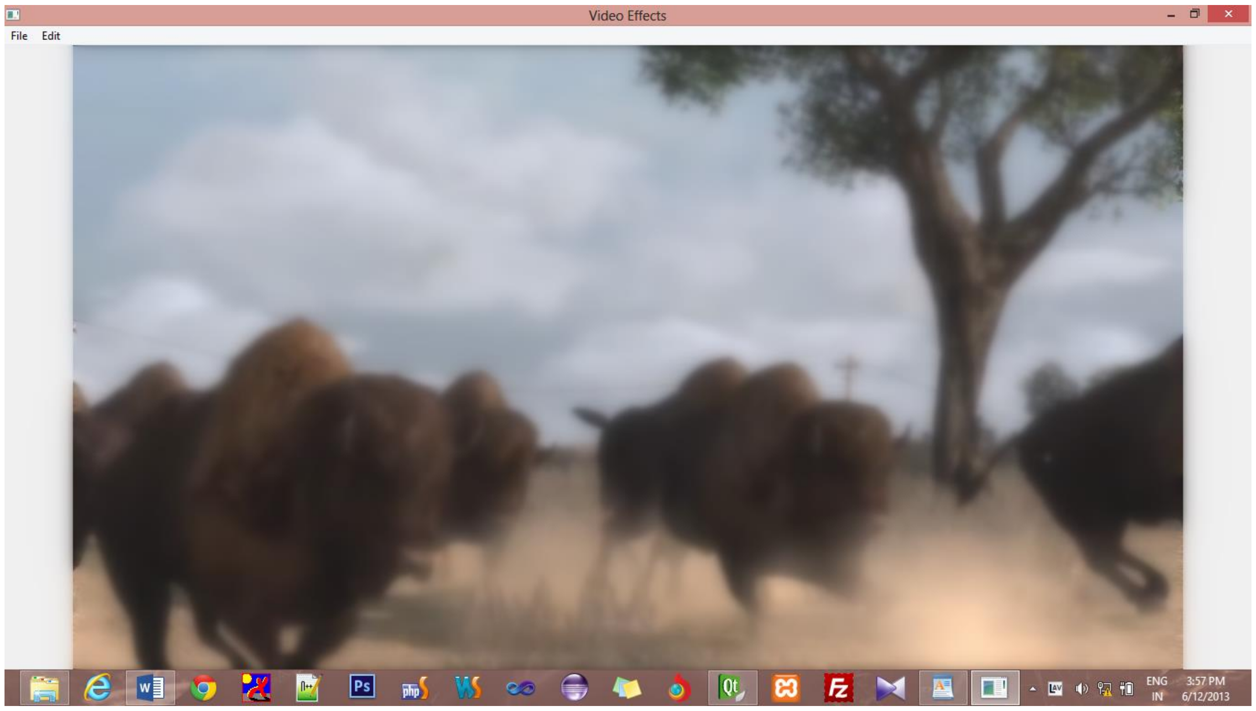
Output & Screenshots :


Media Center - Video
File
Effects
Help
A Mat Whitecross Film
00:04 / 04:20
bg_fire.mp4
Coldplay - Paradise.mp4
Crush - David Archuleta (with lyrics).flv
Daru Desi (Full Official Song) - Cocktail.mp
David Archuleta - Crush.flv
David Guetta - Titanium ft. Sia.mp4
David Guetta (NEW SONG 2012).mp4
Eminem - Forgive Me (NEW 2012).mp4
emptiness.mp4
Flo Rida - Whistle [Official Video].mp4
Fun._ We Are Young ft. Janelle Monáe [OFF
Gandi Baat Song ft. Shahid Kapoor, Prabhu
Imagine Dragons - Demons (Official Music
Imagine Dragons - Radioactive.mp4
Jennifer Lopez - Dance Again ft. Pitbull.mp
Kaise Bataaoon Song - 3G ft. Neil Nitin Mu
Katy Perry - Part Of Me.mp4
Lat Lag Gayee - Race 2 - Official Song Vide
Lil Wayne - Mirror ft. Bruno Mars (1).mp4
Linkin Park - BURN IT DOWN (Official Mus
Linkin Park - Leave Out All The Rest (Offici
Linkin Park - Shadow Of The Day (Video).fl
LIVE MY LIFE.MP4
Main Rang Sharbaton Ka - Phata Poster Nil
Mann Mera Song - Table No.21 ft. Rajeev K
Miley Cyrus - Wrecking Ball.mp4
Moneyball [last scene].mp4
Naa Jaane Kahan Se Aaya Hai Full Song _I N
O Sajna Song - Table No.21 ft. Rajeev Khan
Allah Duhai Hai - Race 2 - Official Song Vic
Avicii - Levels.mp4
Blue Eyes Full Video Song Yo Yo Honey Sin
ENG IN
3:19 PM
6/12/2013

Video Player with Playlist.


Media Center - Video
File
Effects
Help
Open
« Storage (F:) › VIDEOS › new
Search new
Organize
New folder
Libraries
Documents
Music
Pictures
Videos
Homegroup
Computer
OS (C:)
CD Drive (E:)
Storage (F:)
anirudh (aniruc
Aa Bhi Ja Mere Mehermaan feat. Atif Aslam - Official Video Ja...
AASHIQUI 2 MASHUP FULL SONG _ KIRAN KAMATH.mp4
ABCD Bezubaan full song HD.mp4
ADELE SKYFALL.FLV
ALEXANDRA STAN - Cliche (Hush Hush) (Official Hd Vid...
Alexandra Stan - Get back (ASAP) HD.mp4
Alexandra Stan - Get Back (ASAP) MAAN Studio REMIX _ THE ...
Alexandra Stan - Lemonade (OFFICIAL MUSIC VIDEO).mp4
File name:
Video Files (*)
Open
Cancel
00:52 / 04:20
ENG IN
3:20 PM
6/12/2013

Open Dialogue in Video Player



Live Blur Video Effect in Video Player

3.2.2) LAN Chat Module :

One of the module in the Media Center for the functionality of LAN Chatting.

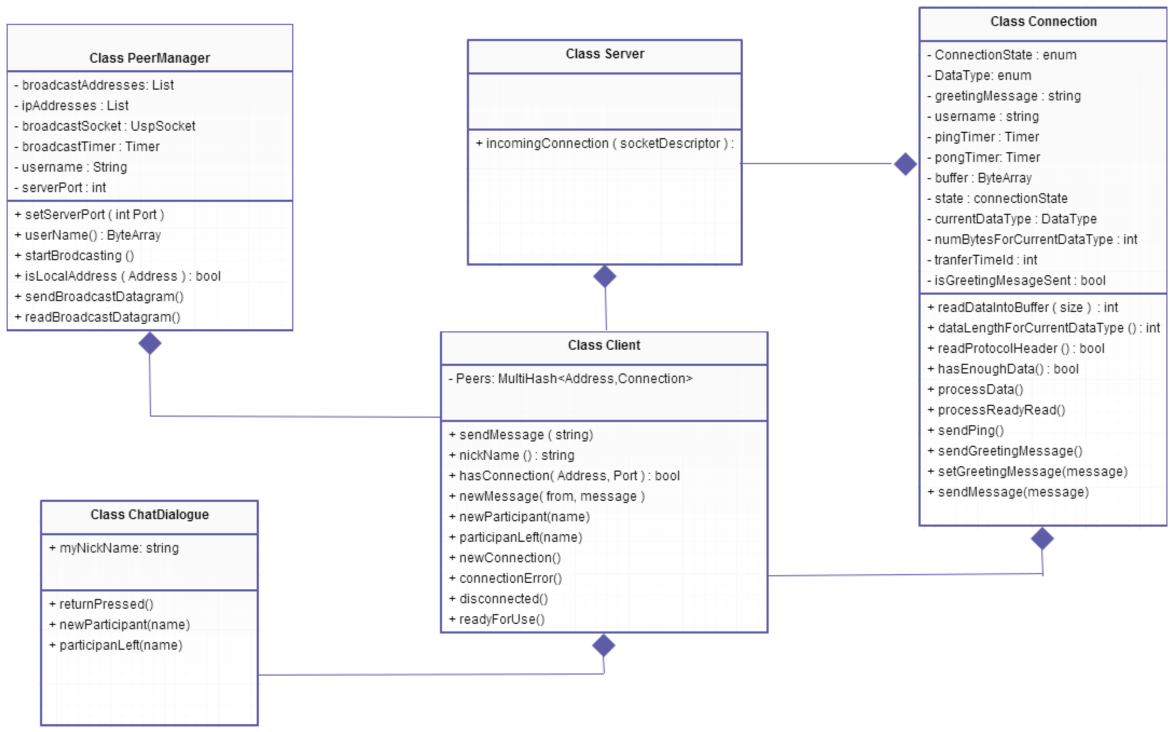
Makes use of QTcpSocket and QTcpServer for making TCP connections and handling them to Chat.

Functionalities :

Cross Platform Support.

Connects all the users in a network to Chat.

Class Diagram :



Code:

1) Connection.h

```
#ifndef CONNECTION_H
#define CONNECTION_H

#include <QHostAddress>
#include <QString>
#include <QTcpSocket>
#include <QTime>
#include <QTimer>

static const int MaxBufferSize = 1024000;

class Connection : public QTcpSocket
{
   Q_OBJECT

public:
   enum ConnectionState
   {
      WaitingForGreeting,
      ReadingGreeting,
      ReadyForUse
   };
   enum DataType
   {
      PlainText,
      Ping,
      Pong,
      Greeting,
      Undefined
   };

   Connection(QObject *parent = 0);

   QString name() const;
   void setGreetingMessage(const QString &message);
   bool sendMessage(const QString &message);

signals:
   void readyForUse();
   void newMessage(const QString &from, const QString &message);
protected:
   void timerEvent(QTimerEvent *timerEvent);

private slots:
   void processReadyRead();
   void sendPing();
   void sendGreetingMessage();

private:
   int readDataIntoBuffer(int maxSize = MaxBufferSize);
   int dataLengthForCurrentDataType();
   bool readProtocolHeader();
   bool hasEnoughData();
   void processData();

   QString greetingMessage;
   QString username;
   QTimer pingTimer;
   QTime pongTime;
   QByteArray buffer;
   ConnectionState state;
   DataType currentDataType;
   int numBytesForCurrentDataType;
   int transferTimerId;
   bool isGreetingMessageSent;
};

#endif
```

2) <u>Connection.cpp</u>

```
#include "connection.h"

#include <QtNetwork>

static const int TransferTimeout = 30 * 1000;
static const int PongTimeout = 60 * 1000;
static const int PingInterval = 5 * 1000;
static const char SeparatorToken = ' ';

Connection::Connection(QObject *parent) : QTcpSocket(parent)
{
    greetingMessage = tr("undefined");
    username = tr("unknown");
    state = WaitingForGreeting;
    currentDataType = Undefined;
    numBytesForCurrentDataType = -1;
    transferTimerId = 0;
    isGreetingMessageSent = false;
    pingTimer.setInterval(PingInterval);

    QObject::connect(this, SIGNAL(readyRead()), this, SLOT(processReadyRead()));
    QObject::connect(this, SIGNAL(disconnected()), &pingTimer, SLOT(stop()));
    QObject::connect(&pingTimer, SIGNAL(timeout()), this, SLOT(sendPing()));
    QObject::connect(this, SIGNAL(connected()),this, SLOT(sendGreetingMessage()));
}

QString Connection::name() const
{
    return username;
}

void Connection::setGreetingMessage(const QString &message)
{
    greetingMessage = message;
}

bool Connection::sendMessage(const QString &message)
{
    if (message.isEmpty())
        return false;

    QByteArray msg = message.toUtf8();
    QByteArray data = "MESSAGE " + QByteArray::number(msg.size()) + ' ' + msg;
    return write(data) == data.size();
}

void Connection::timerEvent(QTimerEvent *timerEvent)
{
    if (timerEvent->timerId() == transferTimerId)
    {
        abort();
        killTimer(transferTimerId);
        transferTimerId = 0;
    }
}

void Connection::processReadyRead()
{
```

```
if (state == WaitingForGreeting)
{
   if (!readProtocolHeader())
   {
      return;
   }
   if (currentDataType != Greeting)
   {
      abort();
      return;
   }
   state = ReadingGreeting;
}

if (state == ReadingGreeting)
{
   if (!hasEnoughData())
   {
      return;
   }

   buffer = read(numBytesForCurrentDataType);
   if (buffer.size() != numBytesForCurrentDataType)
   {
      abort();
      return;
   }

   username = QString(buffer) + ' (' + peerAddress().toString() + ')';

   currentDataType = Undefined;
   numBytesForCurrentDataType = 0;
   buffer.clear();

   if (!isValid())
   {
      abort();
      return;
   }

   if (!isGreetingMessageSent)
      sendGreetingMessage();

   pingTimer.start();
   pongTime.start();
   state = ReadyForUse;
```

```
        emit readyForUse();
    }

    do {
        if (currentDataType == Undefined)
        {
            if (!readProtocolHeader())
                return;
        }
        if (!hasEnoughData())
            return;
        processData();
    } while (bytesAvailable() > 0);
}

void Connection::sendPing()
{
    if (pongTime.elapsed() > PongTimeout)
    {
        abort();
        return;
    }

    write("PING 1 p");
}

void Connection::sendGreetingMessage()
{
    QByteArray greeting = greetingMessage.toUtf8();
    QByteArray data = "GREETING " + QByteArray::number(greeting.size()) + ' ' + greeting;
    if (write(data) == data.size())
        isGreetingMessageSent = true;
}

int Connection::readDataIntoBuffer(int maxSize)
{
    if (maxSize > MaxBufferSize)
        return 0;

    int numBytesBeforeRead = buffer.size();
    if (numBytesBeforeRead == MaxBufferSize)
    {
        abort();
        return 0;
    }
    while (bytesAvailable() > 0 && buffer.size() < maxSize)
    {
        buffer.append(read(1));
        if (buffer.endsWith(SeparatorToken))
            break;
    }
    return buffer.size() - numBytesBeforeRead;
}

int Connection::dataLengthForCurrentDataType()
{
    if (bytesAvailable() <= 0 || readDataIntoBuffer() <= 0
            || !buffer.endsWith(SeparatorToken))
        return 0;

    buffer.chop(1);
    int number = buffer.toInt();
    buffer.clear();
    return number;
}

bool Connection::readProtocolHeader()
{
    if (transferTimerId)
    {
        killTimer(transferTimerId);
        transferTimerId = 0;
    }

    if (readDataIntoBuffer() <= 0)
    {
        transferTimerId = startTimer(TransferTimeout);
        return false;
    }

    if (buffer == "PING ")
    {
        currentDataType = Ping;
    }
    else if (buffer == "PONG ")
    {
        currentDataType = Pong;
    }
    else if (buffer == "MESSAGE ")
    {
        currentDataType = PlainText;
    }
    else if (buffer == "GREETING ") {
        currentDataType = Greeting;
    }
    else
    {
        currentDataType = Undefined;
        abort();
        return false;
    }

    buffer.clear();
    numBytesForCurrentDataType = dataLengthForCurrentDataType();
    return true;
}

bool Connection::hasEnoughData()
{
    if (transferTimerId)
    {
        QObject::killTimer(transferTimerId);
        transferTimerId = 0;
    }

    if (numBytesForCurrentDataType <= 0)
        numBytesForCurrentDataType = dataLengthForCurrentDataType();

    if (bytesAvailable() < numBytesForCurrentDataType || numBytesForCurrentDataType <= 0)
    {
        transferTimerId = startTimer(TransferTimeout);
        return false;
    }

    return true;
}

void Connection::processData()
{
    buffer = read(numBytesForCurrentDataType);
    if (buffer.size() != numBytesForCurrentDataType)
    {
        abort();
        return;
    }

    switch (currentDataType)
    {
    case PlainText:
        emit newMessage(username, QString::fromUtf8(buffer));
        break;
    case Ping:
        write("PONG 1 p");
        break;
    case Pong:
        pongTime.restart();
        break;
    default:
        break;
    }

    currentDataType = Undefined;
    numBytesForCurrentDataType = 0;
    buffer.clear();

}
```

3) PeerManager.h

```
#ifndef PEERMANAGER_H
#define PEERMANAGER_H

#include <QByteArray>
#include <QList>
#include <QObject>
#include <QTimer>
#include <QUdpSocket>

class Client;
class Connection;

class PeerManager : public QObject
{
    Q_OBJECT

public:
    PeerManager(Client *client);
    void setServerPort(int port);
    QByteArray userName() const;
    void startBroadcasting();
    bool isLocalHostAddress(const QHostAddress &address);

signals:
    void newConnection(Connection *connection);

private slots:
    void sendBroadcastDatagram();
    void readBroadcastDatagram();

private:
    void updateAddresses();

    Client *client;
    QList<QHostAddress> broadcastAddresses;
    QList<QHostAddress> ipAddresses;
    QUdpSocket broadcastSocket;
    QTimer broadcastTimer;
    QByteArray username;
    int serverPort;
};

#endif
```

4) **PeerManager.cpp**

```
#include <QtNetwork>

#include "client.h"
#include "connection.h"
#include "peermanager.h"

static const qint32 BroadcastInterval = 2000;
static const unsigned broadcastPort = 45000;

PeerManager::PeerManager(Client *client) : QObject(client)
{
    this->client = client;

    QStringList envVariables;
    envVariables << "USERNAME.*" << "USER.*" << "USERDOMAIN.*"
                 << "HOSTNAME.*" << "DOMAINNAME.*";
    QStringList environment = QProcess::systemEnvironment();
    foreach (QString string, envVariables) {
        int index = environment.indexOf(QRegExp(string));
        if (index != -1) {
            QStringList stringList = environment.at(index).split('=');
            if (stringList.size() == 2) {
                username = stringList.at(1).toUtf8();
                break;
            }
        }
    }

    if (username.isEmpty())
        username = "unknown";

    updateAddresses();
    serverPort = 0;

    broadcastSocket.bind(QHostAddress::Any, broadcastPort, QUdpSocket::ShareAddress
                         | QUdpSocket::ReuseAddressHint);
    connect(&broadcastSocket, SIGNAL(readyRead()),
            this, SLOT(readBroadcastDatagram()));

    broadcastTimer.setInterval(BroadcastInterval);
    connect(&broadcastTimer, SIGNAL(timeout()),
            this, SLOT(sendBroadcastDatagram()));
}

void PeerManager::setServerPort(int port)
{
    serverPort = port;
}

QByteArray PeerManager::userName() const
{
    return username;
}

void PeerManager::startBroadcasting()
{
    broadcastTimer.start();
}

bool PeerManager::isLocalHostAddress(const QHostAddress &address)
{
    foreach (QHostAddress localAddress, ipAddresses) {
        if (address == localAddress)
            return true;
    }
    return false;
}

void PeerManager::sendBroadcastDatagram()
{
    QByteArray datagram(username);
    datagram.append('@');
    datagram.append(QByteArray::number(serverPort));

    bool validBroadcastAddresses = true;
    foreach (QHostAddress address, broadcastAddresses) {
        if (broadcastSocket.writeDatagram(datagram, address,
                                          broadcastPort) == -1)
            validBroadcastAddresses = false;
    }

    if (!validBroadcastAddresses)
        updateAddresses();
}

void PeerManager::readBroadcastDatagram()
{
    while (broadcastSocket.hasPendingDatagrams()) {
        QHostAddress senderIp;
        quint16 senderPort;
        QByteArray datagram;
        datagram.resize(broadcastSocket.pendingDatagramSize());
        if (broadcastSocket.readDatagram(datagram.data(), datagram.size(),
                                         &senderIp, &senderPort) == -1)
            continue;

        QList<QByteArray> list = datagram.split('@');
        if (list.size() != 2)
            continue;

        int senderServerPort = list.at(1).toInt();
        if (isLocalHostAddress(senderIp) && senderServerPort == serverPort)
            continue;

        if (!client->hasConnection(senderIp)) {
            Connection *connection = new Connection(this);
            emit newConnection(connection);
            connection->connectToHost(senderIp, senderServerPort);
        }
    }
}

void PeerManager::updateAddresses()
{
    broadcastAddresses.clear();
    ipAddresses.clear();
    foreach (QNetworkInterface interface, QNetworkInterface::allInterfaces()) {
        foreach (QNetworkAddressEntry entry, interface.addressEntries()) {
            QHostAddress broadcastAddress = entry.broadcast();
            if (broadcastAddress != QHostAddress::Null && entry.ip() !=
QHostAddress::LocalHost) {
                broadcastAddresses << broadcastAddress;
                ipAddresses << entry.ip();
            }
        }
    }
}
```

5) Client.h

```
#ifndef CLIENT_H
#define CLIENT_H

#include <QAbstractSocket>
#include <QHash>
#include <QHostAddress>

#include "server.h"

class PeerManager;

class Client : public QObject
{
    Q_OBJECT

public:
    Client();
    void sendMessage(const QString &message);
    QString nickName() const;
    bool hasConnection(const QHostAddress &senderIp, int senderPort = -1) const;

signals:
    void newMessage(const QString &from, const QString &message);
    void newParticipant(const QString &nick);
    void participantLeft(const QString &nick);

private slots:
    void newConnection(Connection *connection);
    void connectionError(QAbstractSocket::SocketError socketError);
    void disconnected();
    void readyForUse();

private:
    void removeConnection(Connection *connection);

    PeerManager *peerManager;
    Server server;
    QMultiHash<QHostAddress, Connection *> peers;
};

#endif
```

6) Client.cpp

```
#include <QtNetwork>

#include "client.h"
#include "connection.h"
#include "peermanager.h"

Client::Client()
{
    peerManager = new PeerManager(this);
    peerManager->setServerPort(server.serverPort());
    peerManager->startBroadcasting();

    QObject::connect(peerManager, SIGNAL(newConnection(Connection*)),this,
SLOT(newConnection(Connection*)));
    QObject::connect(&server, SIGNAL(newConnection(Connection*)),this,
SLOT(newConnection(Connection*)));
}

void Client::sendMessage(const QString &message)
{
    if (message.isEmpty())
    {
        return;
    }

    QList<Connection *> connections = peers.values();
    foreach (Connection *connection, connections)
    {
        connection->sendMessage(message);
    }
}

QString Client::nickName() const
{
    return QString(peerManager->userName()) + " (" + QHostInfo::localHostName() + ')';

}

bool Client::hasConnection(const QHostAddress &senderIp, int senderPort) const
{
    if (senderPort == -1)
    {
        return peers.contains(senderIp);
    }

    if (!peers.contains(senderIp))
    {
        return false;
    }

    QList<Connection *> connections = peers.values(senderIp);
    foreach (Connection *connection, connections)
    {
        if (connection->peerPort() == senderPort)
            return true;
    }

    return false;
}

void Client::newConnection(Connection *connection)
{
    connection->setGreetingMessage(peerManager->userName());

    connect(connection, SIGNAL(error(QAbstractSocket::SocketError)),
            this, SLOT(connectionError(QAbstractSocket::SocketError)));
    connect(connection, SIGNAL(disconnected()), this, SLOT(disconnected()));
    connect(connection, SIGNAL(readyForUse()), this, SLOT(readyForUse()));
}

void Client::readyForUse()
{
    Connection *connection = qobject_cast<Connection *>(sender());
    if (!connection || hasConnection(connection->peerAddress(),connection->peerPort()))
    {
        return;
    }

    connect(connection, SIGNAL(newMessage(QString,QString)),this,
SIGNAL(newMessage(QString,QString)));

    peers.insert(connection->peerAddress(), connection);
    QString nick = connection->name();
    if (!nick.isEmpty())
    {
        emit newParticipant(nick);
    }
}

void Client::disconnected()
{
    if (Connection *connection = qobject_cast<Connection *>(sender()))
        removeConnection(connection);
}

void Client::connectionError(QAbstractSocket::SocketError)
{
    if (Connection *connection = qobject_cast<Connection *>(sender()))
        removeConnection(connection);
}

void Client::removeConnection(Connection *connection)
{
    if (peers.contains(connection->peerAddress()))
    {
        peers.remove(connection->peerAddress());
        QString nick = connection->name();
        if (!nick.isEmpty())
            emit participantLeft(nick);
    }
    connection->deleteLater();
}
```

7) Server.h

```
#ifndef SERVER_H
#define SERVER_H

#include <QTcpServer>

class Connection;

class Server : public QTcpServer
{
   Q_OBJECT

public:
   Server(QObject *parent = 0);

signals:
   void newConnection(Connection *connection);

protected:
   void incomingConnection(qintptr socketDescriptor);
};

#endif
```

8) Server.cpp

```
#include <QtNetwork>

#include "connection.h"
#include "server.h"

Server::Server(QObject *parent)
   : QTcpServer(parent)
{
   listen(QHostAddress::Any);
}

void Server::incomingConnection(qintptr socketDescriptor)
{
   Connection *connection = new Connection(this);
   connection->setSocketDescriptor(socketDescriptor);
   emit newConnection(connection);
}
```

9) Chat_diaogue.h

```
#ifndef CHATDIALOG_H
#define CHATDIALOG_H

#include "ui_chatdialog.h"
#include "client.h"

class ChatDialog : public QDialog, private Ui::ChatDialog
{
    Q_OBJECT

public:
    ChatDialog(QWidget *parent = 0);

public slots:
    void appendMessage(const QString &from, const QString &message);

private slots:
    void returnPressed();
    void newParticipant(const QString &nick);
    void participantLeft(const QString &nick);


private:
    Client client;
    QString myNickName;
    QTextTableFormat tableFormat;
};

#endif
```

10) Chat_dialogue.cpp

```
#include <QtWidgets>

#include "chatdialog.h"
```

```
ChatDialog::ChatDialog(QWidget *parent) : QDialog(parent)
{
   setupUi(this);

   lineEdit->setFocusPolicy(Qt::StrongFocus);
   textEdit->setFocusPolicy(Qt::NoFocus);
   textEdit->setReadOnly(true);
   listWidget->setFocusPolicy(Qt::NoFocus);
   listWidget->setStyleSheet("QListWidget::item{padding:10px;}");

   connect(lineEdit, SIGNAL(returnPressed()), this, SLOT(returnPressed()));
   connect(lineEdit, SIGNAL(returnPressed()), this, SLOT(returnPressed()));
   connect(&client, SIGNAL(newMessage(QString,QString)),this,
SLOT(appendMessage(QString,QString)));
   connect(&client, SIGNAL(newParticipant(QString)),this, SLOT(newParticipant(QString)));
   connect(&client, SIGNAL(participantLeft(QString)),this, SLOT(participantLeft(QString)));

   setWindowTitle("Lan Chat Module");

   myNickName = client.nickName();
   newParticipant(myNickName);
   tableFormat.setBorder(0);

}

void ChatDialog::appendMessage(const QString &from, const QString &message)
{
   if (from.isEmpty() || message.isEmpty())
   {
      return;
   }

   QTextCursor cursor(textEdit->textCursor());
   cursor.movePosition(QTextCursor::End);
   QTextTable *table = cursor.insertTable(1, 2, tableFormat);
   table->cellAt(0, 0).firstCursorPosition().insertText('<' + from + "> ");
   table->cellAt(0, 1).firstCursorPosition().insertText(message);
   QScrollBar *bar = textEdit->verticalScrollBar();
   bar->setValue(bar->maximum());
}

void ChatDialog::returnPressed()
{
   QString text = lineEdit->text();
   if (text.isEmpty())
      return;
```

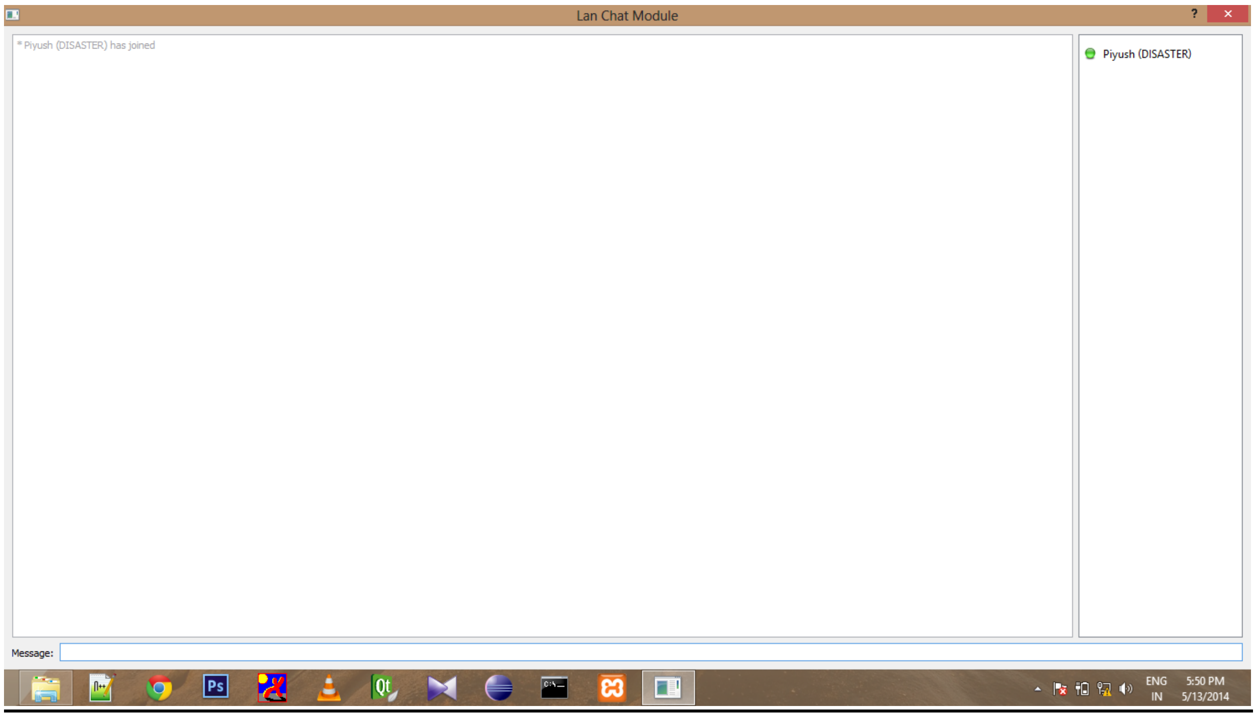
```
    /*if (text.startsWith(QChar('/')))
    {
        QColor color = textEdit->textColor();
        textEdit->setTextColor(Qt::red);
        textEdit->append(tr("! Unknown command: %1").arg(text.left(text.indexOf(' '))));
        textEdit->setTextColor(color);
    }*/
    else
    {
        client.sendMessage(text);
        appendMessage(myNickName, text);
    }

    lineEdit->clear();
}

void ChatDialog::newParticipant(const QString &nick)
{
    if (nick.isEmpty())
        return;

    QColor color = textEdit->textColor();
    textEdit->setTextColor(Qt::gray);
    textEdit->append(tr("* %1 has joined").arg(nick));
    textEdit->setTextColor(color);
    QListWidgetItem *item=new QListWidgetItem(listWidget);
    item->setText(nick);
    item->setIcon(QIcon(":/resources/online.png"));
    listWidget->addItem(item);
}

void ChatDialog::participantLeft(const QString &nick)
{
    if (nick.isEmpty())
        return;

    QList<QListWidgetItem *> items = listWidget->findItems(nick,Qt::MatchExactly);
    if (items.isEmpty())
        return;

    delete items.at(0);
    QColor color = textEdit->textColor();
    textEdit->setTextColor(Qt::gray);
    textEdit->append(tr("* %1 has left").arg(nick));
    textEdit->setTextColor(color);
```

```
}
```

11) main.cpp

```
#include <QApplication>

#include "chatdialog.h"



int main(int argc, char *argv[])
{
   QApplication app(argc, argv);

   ChatDialog dialog;
   dialog.showMaximized();
   return app.exec();
}
```

Output:



```
* Piyush (DISASTER) has joined
```

References

- Qt Reference Manual - http://qt-project.org/doc/qt-5.1/qtdoc/reference-overview.html
- C++ GUI Development with Qt4 (TextBook) – Published in 2005 by Jasmin Blanchette
- Qt's discussion Forum - http://qt-project.org/forums
- Architecture Reference Xbmc: *http://static.telematicsfreedom.org*
- Architecture Reference Windows Media Center: Microsoft Win Hec 2008 by Jonathan Hutchinson & Luigi Capriotti (18 September 2008)
- Architecture Reference Qt: http://www.cetoni.de/development/software/qt-framework.html
- Applications of Qt: www.wikipedia.com