```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QDebug>

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    connect(ui-
>pushButtonAddAudio,SIGNAL(clicked()),this,SLOT(pushButtonAddAudio_clicked()));
    connect(ui-
>pushButtonAddVideo,SIGNAL(clicked()),this,SLOT(pushButtonAddVideo_clicked()));
    connect(ui-
>pushButtonRemoveAudio,SIGNAL(clicked()),this,SLOT(pushButtonRemoveAudio_clicked()));
    connect(ui-
>pushButtonRemoveVideo,SIGNAL(clicked()),this,SLOT(pushButtonRemoveVideo_clicked()));
    connect(ui-
>pushButtonUpdateLibrary,SIGNAL(clicked()),this,SLOT(pushButtonUpdate_clicked()));

    connect(ui-
>treeWidgetLibraryDisplay,SIGNAL(doubleClicked(QModelIndex)),this,SLOT(treeLibraryDisplay
_doubleClicked()));
    connect(ui-
>treeWidgetLibraryDisplay,SIGNAL(itemClicked(QTreeWidgetItem*,int)),this,SLOT(treeLibrary
Display_itemClicked(QTreeWidgetItem*,int)));


    connect(ui->toolButtonFullScreen,SIGNAL(clicked()),this,SLOT(toFullScreen()));
    connect(ui->toolButtonVolume,SIGNAL(clicked()),this,SLOT(toggleMute()));
    connect(ui->toolButtonToScreen,SIGNAL(clicked()),this,SLOT(toVideoMode()));

    connect(ui->toolButtonNext,SIGNAL(clicked()),this,SLOT(setNext_clicked()));
    connect(ui->toolButtonPrevious,SIGNAL(clicked()),this,SLOT(setPrevious_clicked()));


    ui->treeWidgetLibraryDisplay->addAction(ui->actionAddtoQueue);
    connect(ui-
>actionAddtoQueue,SIGNAL(triggered()),this,SLOT(treeLibraryDisplay_Addto_Queue()));

    ui->treeWidgetQueue->addAction(ui->actionRemoveFromQueue);
    connect(ui-
>actionRemoveFromQueue,SIGNAL(triggered()),this,SLOT(treeQueue_RemoveFromQueue()));

    connect(ui-
>treeWidgetCatergoryChooser,SIGNAL(doubleClicked(QModelIndex)),this,SLOT(treeCategoryChos
ser_doubleClicked()));

    connect(ui-
>treeWidgetQueue,SIGNAL(itemDoubleClicked(QTreeWidgetItem*,int)),this,SLOT(treeWidgetQueu
e_onDoubleClicked(QTreeWidgetItem*)));

    connect(ui-
>toolButtonClearQueue,SIGNAL(clicked()),this,SLOT(toolButtonClearQueue_Clicked()));


    mPlay.addFile(":/ui/toolButton/icon/Resources/ui/toolButton/icon/play.png");
    mPause.addFile(":/ui/toolButton/icon/Resources/ui/toolButton/icon/pause.png");
    mVolume.addFile(":/ui/toolButton/icon/Resources/ui/toolButton/icon/volume.png");
    mVolumeMuted.addFile(":/ui/toolButton/icon/Resources/ui/toolButton/icon/
volumeMuted.png");

    ui->toolButtonPlayPause->setIcon(mPlay);

    ui->treeWidgetCatergoryChooser->expandAll();
    ui->toolButtonFullScreen->setIcon(ui->widgetPlayControls->style()-
>standardIcon(QStyle::SP_TitleBarMaxButton));
    ui->toolButtonToScreen->setIcon(ui->widgetPlayControls->style()-
>standardIcon(QStyle::SP_TitleBarNormalButton));
```

```
    mManagerTimer.setSingleShot(true);
    connect(&mManagerTimer,SIGNAL(timeout()),this,SLOT(runManager()));
    mManagerTimer.start(mManagerTimerValue);

    ui->treeWidgetQueue->hideColumn(1);

    //Database Intializations and connections
    mDBAudio = new LibraryManagerAudio(this);
    mDBVideo = new LibraryManagerVideo(this);

    connect(mDBAudio,SIGNAL(updateTreeWidgetLibraryDisplay(vector<vector<QString>
>)),this,SLOT(updateTreeViewAudio(vector<vector<QString> >)));
    connect(mDBAudio,SIGNAL(updatePath(QString)),this,SLOT(getSelectedAudioPath(QString))
);

    connect(mDBVideo,SIGNAL(updateTreeWidgetLibraryDisplay(vector<vector<QString>
>)),this,SLOT(updateTreeViewVideo(vector<vector<QString> >)));
    connect(mDBVideo,SIGNAL(updatePath(QString)),this,SLOT(getSelectedVideoPath(QString))
);

    isParentAudio=true;

    mDBAudio->getSource();
    mDBVideo->getSource();

    ui->treeWidgetLibraryDisplay->hideColumn(10);

    //Initialization and Connection with Gstreamer
    mGstDriver = new QtGStreamerDriver(this);

    connect(mGstDriver, SIGNAL(positionChanged()), this, SLOT(onPositionChanged()));
    connect(mGstDriver, SIGNAL(stateChanged()), this, SLOT(onStateChanged()));

    onStateChanged();


    connect(ui->toolButtonPlayPause,SIGNAL(clicked()),this,SLOT(setPlayPause_clicked()));
    connect(this,SIGNAL(setPlayState()),mGstDriver,SLOT(play()));
    connect(this,SIGNAL(setPauseState()),mGstDriver,SLOT(pause()));
    connect(this,SIGNAL(setStopState()),mGstDriver,SLOT(stop()));

    ui->horizontalSliderMediaPosition->setTracking(false);
    connect(ui-
>horizontalSliderMediaPosition,SIGNAL(valueChanged(int)),this,SLOT(positionSliderMoved(in
t)));

    ui->horizontalSliderVolume->setMaximum(10);

    connect(ui-
>horizontalSliderVolume,SIGNAL(valueChanged(int)),this,SLOT(setVolume(int)));
    ui->horizontalSliderVolume->setValue(05);
    ui->horizontalSliderVolume->setSliderPosition(5);
    ui->horizontalSliderVolume->setEnabled(true);

    //    Linking Video Widget

    mVideoWidget = new VideoSink(this,mGstDriver);
    mVideoWidget->show();
    mVideoWidget->setVisible(false);


    connect(mVideoWidget,SIGNAL(closeMain()),this,SLOT(shutdown()));


    connect(mVideoWidget,SIGNAL(setVideoMode(bool)),this,SLOT(setVideoMode(bool)));
    connect(mVideoWidget,SIGNAL(setVolumeAtMain(int)),this,SLOT(setVolume(int)));
```

```
    connect(mVideoWidget,SIGNAL(nextClicked()),this,SLOT(setNext_clicked()));
    connect(mVideoWidget,SIGNAL(prevClicked()),this,SLOT(setPrevious_clicked()));
    connect(this,SIGNAL(setVolumeAtVideo(int)),mVideoWidget,SLOT(setVolumeSlider(int)));
    connect(this,SIGNAL(setNowPlayingVideo(QString)),mVideoWidget,SLOT(setNowPlaying(QStr
ing)));
    connect(this,SIGNAL(goFullScreen()),mVideoWidget,SLOT(toFullScreen()));

}

MainWindow::~MainWindow()
{
    delete ui;
}


// member functions/ methods

bool caseInsensitiveLessThan(QTreeWidgetItem * s1 , QTreeWidgetItem * s2)
{
    return s1->text(0).toLower() < s2->text(0).toLower();
}

void MainWindow::sortTreeViewAudio()
{
    if(isParentAudio==false)
        return;
    if(mTreeViewDataAudio.empty()==true)
    {
        ui->treeWidgetLibraryDisplay->clear();
        return;
    }

    ui->treeWidgetLibraryDisplay->clear();


    QTreeWidgetItem *vRootItem , *vChildItem;

    QString vCurrentRoot;

    int vIndex;

    vector<QTreeWidgetItem *> vRootList;

    switch(mSortStateAudio)
    {
    case(Folder):
    {
        vIndex=0;
        break;
    }
    case(Artist):
    {
        vIndex=6;
        break;
    }
    case(Album):
    {
        vIndex=3;
        break;
    }
    case(Year):
    {
        vIndex=11;
        break;
    }
    case(Genre):
    {
        vIndex=10;
        break;
```

```
    }
    }

    QFont vTempFont;
    vTempFont.setBold(true);

    vCurrentRoot = mTreeViewDataAudio[0][vIndex];
    if(mSortStateAudio==Folder)
    {
        vRootItem = new QTreeWidgetItem(ui->treeWidgetLibraryDisplay);
        //    vRootItem->setText(0,QFileInfo(vCurrentRoot).fileName()+" :
"+QFileInfo(vCurrentRoot).filePath());
        vRootItem->setText(0,QFileInfo(vCurrentRoot).fileName());
        vRootItem->setFont(0,vTempFont);
        vRootItem->setText(10,QFileInfo(vCurrentRoot).filePath());
        vRootList.push_back(vRootItem);
    }
    else
    {
        vRootItem = new QTreeWidgetItem(ui->treeWidgetLibraryDisplay);
        vRootItem->setText(0,vCurrentRoot);
        vRootItem->setFont(0,vTempFont);
        vRootList.push_back(vRootItem);
    }
    bool foundRoot=false;
    int vRootPos;

    for(unsigned long long i =0 ; i < mTreeViewDataAudio.size(); ++i)
    {
        foundRoot=false;
        vCurrentRoot=mTreeViewDataAudio[i][vIndex];
        for(unsigned long long j=0; j < vRootList.size(); ++j)
        {
            if(mSortStateAudio!=Folder)
            {
                if( vRootList[j]->text(0) == mTreeViewDataAudio[i][vIndex] )
                {
                    vRootPos=j;
                    foundRoot=true;
                    break;
                }
            }
            else
            {
                if( vRootList[j]->text(10) == mTreeViewDataAudio[i][vIndex] )
                {
                    vRootPos=j;
                    foundRoot=true;
                    break;
                }
            }
        }

        if(foundRoot==false)
        {
            if(mSortStateAudio==Folder)
            {
                vRootItem = new QTreeWidgetItem();
                vRootItem->setText(0,QFileInfo(vCurrentRoot).fileName());
                vRootItem->setFont(0,vTempFont);
                vRootItem->setText(10,QFileInfo(vCurrentRoot).filePath());
                vRootList.push_back(vRootItem);
                vRootPos=vRootList.size()-1;
            }
            else
            {
                vRootItem = new QTreeWidgetItem();
                vRootItem->setText(0,vCurrentRoot);
                vRootItem->setFont(0,vTempFont);
```

```
                vRootList.push_back(vRootItem);
                vRootPos=vRootList.size()-1;
            }
        }

        //        if (vCurrentRoot != mTreeViewData[i][vIndex])
        //        {
        //            vCurrentRoot = mTreeViewData[i][vIndex];
        //            vRootItem = new QTreeWidgetItem(ui->treeWidgetLibraryDisplay);
        //            //            vRootItem-
>setText(0,QFileInfo(vCurrentRoot).fileName()+": "+QFileInfo(vCurrentRoot).filePath());
        //            vRootItem->setText(0,QFileInfo(vCurrentRoot).fileName());
        //            vRootItem->setFont(0,vTempFont);
        //            vRootItem->setText(10,QFileInfo(vCurrentRoot).filePath());
        //        }


        vChildItem = new QTreeWidgetItem();
        vChildItem->setText(0,mTreeViewDataAudio[i][1]);
        vChildItem->setText(1,mTreeViewDataAudio[i][4]);
        vChildItem->setText(2,mTreeViewDataAudio[i][3]);
        vChildItem->setText(3,mTreeViewDataAudio[i][5]);
        vChildItem->setText(4,mTreeViewDataAudio[i][7]);
        vChildItem->setText(5,mTreeViewDataAudio[i][6]);
        vChildItem->setText(6,mTreeViewDataAudio[i][10]);
        vChildItem->setText(7,mTreeViewDataAudio[i][11]);
        vChildItem->setText(8,mTreeViewDataAudio[i][8]);
        vChildItem->setText(9,mTreeViewDataAudio[i][9]);
        vChildItem->setText(10,mTreeViewDataAudio[i][2]);
        vRootList[vRootPos]->addChild(vChildItem);
        //        for(unsigned long long i=0; i < mTreeViewData.size();++i)

    }


    //    if(mSortStateAudio!=Folder)
    //    {
    //        std::sort(vRootList.begin(),vRootList.end());
    qSort(vRootList.begin(),vRootList.end(),caseInsensitiveLessThan);
    for(unsigned long long j=0; j < vRootList.size(); ++j)
    {
        vRootItem = new QTreeWidgetItem(ui->treeWidgetLibraryDisplay);
        vRootItem->setText(0,vRootList[j]->text(0));
        vRootItem->setFont(0,vTempFont);
        vRootItem->addChildren(vRootList[j]->takeChildren());
    }
    qDeleteAll(vRootList);
    //    }

    ui->treeWidgetLibraryDisplay->resizeColumnToContents(0);

}

void MainWindow::sortTreeViewVideo()
{
    if(isParentAudio==true)
        return;

    if(mTreeViewDataVideo.empty()==true)
    {    ui->treeWidgetLibraryDisplay->clear();
        return;
    }

    ui->treeWidgetLibraryDisplay->clear();


    QTreeWidgetItem *vRootItem , *vChildItem;

    QString vCurrentRoot;
```

```
    vector<QTreeWidgetItem *> vRootList;

    QFont vTempFont;
    vTempFont.setBold(true);

    vCurrentRoot = mTreeViewDataVideo[0][0];
    //    if(mSortStateVideo==FolderVideo)
    //    {
    vRootItem = new QTreeWidgetItem(ui->treeWidgetLibraryDisplay);
    //    vRootItem->setText(0,QFileInfo(vCurrentRoot).fileName()+" :
"+QFileInfo(vCurrentRoot).filePath());
    vRootItem->setText(0,QFileInfo(vCurrentRoot).fileName());
    vRootItem->setFont(0,vTempFont);
    vRootItem->setText(10,QFileInfo(vCurrentRoot).filePath());
    vRootList.push_back(vRootItem);
    //    }
    bool foundRoot=false;
    int vRootPos;

    for(unsigned long long i =0 ; i < mTreeViewDataVideo.size(); ++i)
    {
        foundRoot=false;
        vCurrentRoot=mTreeViewDataVideo[i][0];
        for(unsigned long long j=0; j < vRootList.size(); ++j)
        {
            if( vRootList[j]->text(10) == mTreeViewDataVideo[i][0] )
            {
                vRootPos=j;
                foundRoot=true;
                break;
            }

        }

        if(foundRoot==false)
        {
            //            if(mSortStateAudio==Folder)
            //            {
            vRootItem = new QTreeWidgetItem();
            vRootItem->setText(0,QFileInfo(vCurrentRoot).fileName());
            vRootItem->setFont(0,vTempFont);
            vRootItem->setText(10,QFileInfo(vCurrentRoot).filePath());
            vRootList.push_back(vRootItem);
            vRootPos=vRootList.size()-1;

        }

        //        if (vCurrentRoot != mTreeViewData[i][vIndex])
        //        {
        //            vCurrentRoot = mTreeViewData[i][vIndex];
        //            vRootItem = new QTreeWidgetItem(ui->treeWidgetLibraryDisplay);
        //            //            vRootItem-
>setText(0,QFileInfo(vCurrentRoot).fileName()+": "+QFileInfo(vCurrentRoot).filePath());
        //            vRootItem->setText(0,QFileInfo(vCurrentRoot).fileName());
        //            vRootItem->setFont(0,vTempFont);
        //            vRootItem->setText(10,QFileInfo(vCurrentRoot).filePath());
        //        }


        vChildItem = new QTreeWidgetItem();
        vChildItem->setText(0,mTreeViewDataVideo[i][1]);
        //        vChildItem->setText(1,mTreeViewDataVideo[i][4]);
        //        vChildItem->setText(2,mTreeViewDataVideo[i][3]);
        //        vChildItem->setText(3,mTreeViewDataVideo[i][5]);
        //        vChildItem->setText(4,mTreeViewDataVideo[i][7]);
        //        vChildItem->setText(5,mTreeViewDataVideo[i][6]);
        //        vChildItem->setText(6,mTreeViewDataVideo[i][10]);
        //        vChildItem->setText(7,mTreeViewDataVideo[i][11]);
```

```
        //        vChildItem->setText(8,mTreeViewDataVideo[i][8]);
        //        vChildItem->setText(9,mTreeViewDataVideo[i][9]);
        vChildItem->setText(10,mTreeViewDataVideo[i][2]);
        vRootList[vRootPos]->addChild(vChildItem);
        //        for(unsigned long long i=0; i < mTreeViewData.size();++i)

    }

    qSort(vRootList.begin(),vRootList.end(),caseInsensitiveLessThan);
    for(unsigned long long j=0; j < vRootList.size(); ++j)
    {
        vRootItem = new QTreeWidgetItem(ui->treeWidgetLibraryDisplay);
        vRootItem->setText(0,vRootList[j]->text(0));
        vRootItem->setFont(0,vTempFont);
        vRootItem->addChildren(vRootList[j]->takeChildren());
    }
    qDeleteAll(vRootList);

    ui->treeWidgetLibraryDisplay->resizeColumnToContents(0);
}

int MainWindow::setSliderOnClick(QSlider * vQSlider , int vClickedPosition)
{
    Qt::MouseButtons vMouseButton = QApplication::mouseButtons();
    QPoint vMousePos = vQSlider->mapFromGlobal(QCursor::pos());
    bool isClicked = (vMouseButton &Qt::LeftButton) &&
            (vMousePos.x() >=0 && vMousePos.y() >=0 &&
             vMousePos.x() < vQSlider->size().width() &&
             vMousePos.y() < vQSlider->size().height());

    if(isClicked == true)
    {
        float vPosRatio = vMousePos.x() / (float)vQSlider->size().width();
        int vSliderRange = vQSlider->maximum()-vQSlider->minimum();
        int vSliderPositionUnderMouse = vQSlider->minimum() + vSliderRange *vPosRatio;

        if(vSliderPositionUnderMouse != vClickedPosition)
        {
            vQSlider->setValue(vSliderPositionUnderMouse);
            return vSliderPositionUnderMouse;
        }
    }
    return vClickedPosition;
}



// Getter Methods

int MainWindow::getSortStateAudio()
{
    return mSortStateAudio;
}

// Setter Methods

void MainWindow::setSortStateAudio(treeWidgetSortStatesAudio vState)
{
    switch(vState)
    {
    case(Folder):
    {
        mSortStateAudio=Folder;
        break;
    }
    case(Artist):
    {
        mSortStateAudio=Artist;
        break;
```

```
    }
    case(Album):
    {
        mSortStateAudio=Album;
        break;
    }
    case(Year):
    {
        mSortStateAudio=Year;
        break;
    }
    case(Genre):
    {
        mSortStateAudio=Genre;
        break;
    }
    default:
    {
        throw std::string("Audio State Cannot be Matched");

    }
    }


}

//SLOTS
void MainWindow::getSelectedAudioPath(QString vPath)
{

    bool doPathExist=false;

    for(int i= 0; i < ui->listWidgetAudioLibrary->count();++i)
    {
        if(ui->listWidgetAudioLibrary->item(i)->text()==vPath)
        {
            doPathExist=true;
        }

    }
    if(doPathExist==false)
    {
        ui->listWidgetAudioLibrary->addItem(vPath);
        mDBAudio->setInitiatorPath(vPath);
        qRegisterMetaType<QFileInfoList>("QFileInfoList");
        qRegisterMetaType<vector<vector<QString> > >("vector<vector<QString> >");
        mDBAudio->start(QThread::HighPriority);
    }
    else
    {
        QMessageBox::information(NULL,"Warning","Path : "+vPath+"Already Exist");
    }

}

void MainWindow::getSelectedVideoPath(QString vPath)
{
    bool doPathExist=false;

    for(int i= 0; i < ui->listWidgetVideoLibrary->count();++i)
    {
        if(ui->listWidgetVideoLibrary->item(i)->text()==vPath)
        {
            doPathExist=true;
        }

    }
    if(doPathExist==false)
    {
```

```
        ui->listWidgetVideoLibrary->addItem(vPath);

        mDBVideo->setInitiatorPath(vPath);
        qRegisterMetaType<QFileInfoList>("QFileInfoList");
        qRegisterMetaType<vector<vector<QString> > >("vector<vector<QString> >");
        mDBVideo->start(QThread::HighestPriority);

    }
    else
    {
        QMessageBox::information(NULL,"Warning","Path : "+vPath+"Already Exist");
    }

    //    ui->listWidgetVideoLibrary->addItem(vPath);
}

void MainWindow::pushButtonAddAudio_clicked()
{
    mDialog = new SelectFileDialog(this);
    connect
(mDialog,SIGNAL(selectedPath(QString)),this,SLOT(getSelectedAudioPath(QString)));
    mDialog->exec();
}

void MainWindow::pushButtonRemoveAudio_clicked()
{
    mDBAudio->setDestroyerPath(ui->listWidgetAudioLibrary->currentItem()->text());
    mDBAudio->start(QThread::HighestPriority);

    qDeleteAll(ui->listWidgetAudioLibrary->selectedItems());
}

void MainWindow::pushButtonAddVideo_clicked()
{
    mDialog = new SelectFileDialog(this);
    connect
(mDialog,SIGNAL(selectedPath(QString)),this,SLOT(getSelectedVideoPath(QString)));
    mDialog->exec();
}

void MainWindow::pushButtonRemoveVideo_clicked()
{
    mDBVideo->setDestroyerPath(ui->listWidgetVideoLibrary->currentItem()->text());
    mDBVideo->start(QThread::HighestPriority);

    qDeleteAll(ui->listWidgetVideoLibrary->selectedItems());
}

void MainWindow::updateTreeViewAudio(vector<vector<QString> > vTreeViewData)
{
    mTreeViewDataAudio.clear();
    mTreeViewDataAudio=vTreeViewData;
    sortTreeViewAudio();
}

void MainWindow::updateTreeViewVideo(vector<vector<QString> > vTreeViewData)
{
    mTreeViewDataVideo.clear();
    mTreeViewDataVideo=vTreeViewData;
    sortTreeViewVideo();
}


void MainWindow::onStateChanged()
{

    QGst::State vNewState = mGstDriver->getState();

    if(vNewState == QGst::StateNull || vNewState == QGst::StatePaused)
```

```
    {
        ui->toolButtonPlayPause->setIcon(mPlay);
        isPlaying=false;
    }
    else if(vNewState == QGst::StatePlaying)
    {
        ui->toolButtonPlayPause->setIcon(mPause);
        mTempTime = 0;
        mTempTime = mTempTime + ( mGstDriver->getDuration().hour() * 60);
        mTempTime = ( mTempTime +  mGstDriver->getDuration().minute() ) * 60;
        mTempTime = mTempTime + (mGstDriver->getDuration().second());
        ui->horizontalSliderMediaPosition->setMaximum(mTempTime);
        isPlaying=true;
    }



    ui->toolButtonNext->setEnabled(vNewState != QGst::StateNull);
    ui->toolButtonPrevious->setEnabled(vNewState != QGst::StateNull);
    ui->horizontalSliderMediaPosition->setEnabled(vNewState != QGst::StateNull);

    if(vNewState == QGst::StatePaused || vNewState == QGst::StatePlaying)
    {
        ui->labelTime->show();
        ui->labelLength->show();
        ui->labelTime->setEnabled(true);
        ui->horizontalSliderMediaPosition->setEnabled(true);
    }
    else
    {
        ui->labelTime->hide();
        ui->labelLength->hide();
    }


    //if we are in Null state, call onPositionChanged() to restore
    //the position of the slider and the text on the label


    if (vNewState == QGst::StateNull) {
        onPositionChanged();

        ui->labelNowPlaying->setText("");
        emit(setNowPlayingVideo(""));

    }
}

void MainWindow::onPositionChanged()
{

    if (mGstDriver->getState() != QGst::StateReady && mGstDriver->getState() !=
QGst::StateNull)
    {
        mPlayBacklength = mGstDriver->getDuration();
        if(mPlayBacklength.hour()==0)
        {
            ui->labelLength->setText(mPlayBacklength.toString("mm:ss"));
        }
        else
        {
            ui->labelLength->setText(mPlayBacklength.toString("HH:mm:ss"));
        }
        mPlayBackcurpos = mGstDriver->getPosition();


        if(mPlayBackcurpos.toString() == mPlayBacklength.toString())
        {
            if(mNowPlaying != NULL && ui->treeWidgetQueue->itemBelow(mNowPlaying))
```

```
            {
                treeWidgetQueue_onDoubleClicked(ui->treeWidgetQueue-
>itemBelow(mNowPlaying));
            }
        }
    }

    if(mPlayBackcurpos.hour() ==0)
    {
        ui->labelTime->setText(mPlayBackcurpos.toString("mm:ss"));
    }
    else
    {
        ui->labelTime->setText(mPlayBackcurpos.toString("HH:mm:ss"));
    }

    if(mGstDriver->getState()!= QGst::StateNull)
    {
        mTempTime = 0;
        mTempTime = mTempTime + ( mPlayBackcurpos.hour() * 60);
        mTempTime = ( mTempTime +  mPlayBackcurpos.minute() ) * 60;
        mTempTime = mTempTime + (mPlayBackcurpos.second());
        if(ui->horizontalSliderMediaPosition->isSliderDown()==false)
        {
            ui->horizontalSliderMediaPosition->setSliderPosition(mTempTime);
        }
    }
    else
    {
        ui->horizontalSliderMediaPosition->setValue(0);
        ui->horizontalSliderMediaPosition->setSliderPosition(0);
        //        QMessageBox::information(this,"State Changed","Playing");


    }
}


void MainWindow::setPlayPause_clicked()
{
    if(isPlaying==true)
    {
        isPlaying=false;
        emit(setPauseState());
    }
    else
    {
        isPlaying=true;
        emit(setPlayState());
    }

}

void MainWindow::setNext_clicked()
{
    if(mNowPlaying != NULL && ui->treeWidgetQueue->itemBelow(mNowPlaying))
    {
        treeWidgetQueue_onDoubleClicked(ui->treeWidgetQueue->itemBelow(mNowPlaying));
    }
}
void MainWindow::setPrevious_clicked()
{
    if(mNowPlaying != NULL && ui->treeWidgetQueue->itemAbove(mNowPlaying))
    {
        treeWidgetQueue_onDoubleClicked(ui->treeWidgetQueue->itemAbove(mNowPlaying));
    }

}
```

```
void MainWindow::treeLibraryDisplay_doubleClicked()
{
    QTreeWidgetItem *vItem;
    QTreeWidgetItemIterator vItemIterator(ui->treeWidgetQueue);

    bool doItemExist=false;

    while(*vItemIterator)
    {
        if( (*vItemIterator)->text(1) != ui->treeWidgetLibraryDisplay->currentItem()->text(10) )
        {
            (*vItemIterator)->setBackgroundColor(0,Qt::white);
            (*vItemIterator)->setSelected(false);
        }
        else
        {
            doItemExist=true;
            (*vItemIterator)->setBackgroundColor(0,Qt::green);
            (*vItemIterator)->setSelected(true);
            mNowPlaying=(*vItemIterator);
        }
        vItemIterator++;
    }

    if(ui->treeWidgetLibraryDisplay->currentItem()->parent() != NULL)
    {

        emit(setStopState());

        if(isParentAudio==false)
        {

            if(doItemExist==false)
            {
                vItem = new QTreeWidgetItem(ui->treeWidgetQueue);
                vItem->setText(0,ui->treeWidgetLibraryDisplay->currentItem()->text(0));
                vItem->setText(1,ui->treeWidgetLibraryDisplay->currentItem()->text(10));
                vItem->setBackgroundColor(0,Qt::green);

                mNowPlaying=vItem;
            }
            ui->labelNowPlaying->setText(ui->treeWidgetLibraryDisplay->currentItem()->text(0));
            emit(setNowPlayingVideo(ui->treeWidgetLibraryDisplay->currentItem()->text(0)));
            toVideoMode();
        }
        else
        {


            if(ui->treeWidgetLibraryDisplay->currentItem()->text(3) != "")
            {
                if(doItemExist==false)
                {
                    vItem = new QTreeWidgetItem(ui->treeWidgetQueue);
                    vItem->setText(0,ui->treeWidgetLibraryDisplay->currentItem()->text(3));
                    vItem->setText(1,ui->treeWidgetLibraryDisplay->currentItem()->text(10));
                    vItem->setBackgroundColor(0,Qt::green);
                    vItem->setSelected(true);
                    mNowPlaying=vItem;
                }
                ui->labelNowPlaying->setText(ui->treeWidgetLibraryDisplay->currentItem()->text(3));
                emit(setNowPlayingVideo(ui->treeWidgetLibraryDisplay->currentItem()->text(3)));
```

```
            }
            else
            {
                if(doItemExist==false)
                {
                    vItem = new QTreeWidgetItem(ui->treeWidgetQueue);
                    vItem->setText(0,ui->treeWidgetLibraryDisplay->currentItem()-
>text(0));
                    vItem->setText(1,ui->treeWidgetLibraryDisplay->currentItem()-
>text(10));
                    vItem->setBackgroundColor(0,Qt::green);
                    vItem->setSelected(true);
                    mNowPlaying=vItem;
                }
                ui->labelNowPlaying->setText(ui->treeWidgetLibraryDisplay->currentItem()-
>text(0));
                emit(setNowPlayingVideo(ui->treeWidgetLibraryDisplay->currentItem()-
>text(0)));
            }
        }


        mGstDriver->setPath(ui->treeWidgetLibraryDisplay->currentItem()->text(10));
        //        ui->toolButtonPlayPause->click();
        emit(setPlayState());
    }

    //    QMessageBox::information(this,"Testing",vCurrentItem->text(10));
}

void MainWindow::treeLibraryDisplay_itemClicked(QTreeWidgetItem * vItem,int vColumn)
{
    ui->treeWidgetLibraryDisplay->resizeColumnToContents(vColumn);
}

void MainWindow::positionSliderMoved(int vSliderValue)
{

    vSliderValue=setSliderOnClick(ui->horizontalSliderMediaPosition,vSliderValue);
    mTempTime = vSliderValue;
    int vSeconds = mTempTime % 60;
    mTempTime = mTempTime/60 ;
    int vMinutes = mTempTime % 60;
    int vHours = mTempTime/60;
    QTime vPosition(vHours,vMinutes,vSeconds);
    mGstDriver->setPosition(vPosition);
}

void MainWindow::setVolume(int vSliderValue)
{
    vSliderValue = setSliderOnClick(ui->horizontalSliderVolume,vSliderValue);
    mGstDriver->setVolume(vSliderValue);
    ui->horizontalSliderVolume->setSliderPosition(vSliderValue);
    emit(setVolumeAtVideo(vSliderValue));
}

void MainWindow::treeCategoryChosser_doubleClicked()
{
    if(ui->treeWidgetCatergoryChooser->currentItem()->parent() != NULL)
    {
        if(ui->treeWidgetCatergoryChooser->currentItem()->parent()->text(0)!="Video")
        {
            ui->treeWidgetLibraryDisplay->showColumn(1);
            ui->treeWidgetLibraryDisplay->showColumn(2);
            ui->treeWidgetLibraryDisplay->showColumn(3);
            ui->treeWidgetLibraryDisplay->showColumn(5);
            ui->treeWidgetLibraryDisplay->showColumn(6);
            ui->treeWidgetLibraryDisplay->showColumn(7);
            ui->treeWidgetLibraryDisplay->showColumn(8);
```

```
            ui->treeWidgetLibraryDisplay->showColumn(9);
            ui->treeWidgetLibraryDisplay->hideColumn(10);

            isParentAudio=true;
            switch(ui->treeWidgetCatergoryChooser->currentIndex().row())
            {
            case(0):
            {
                mSortStateAudio=Folder;
                break;
            }
            case(1):
            {
                mSortStateAudio=Artist;
                break;
            }
            case(2):
            {
                mSortStateAudio=Album;
                break;
            }
            case(3):
            {
                mSortStateAudio=Genre;
                break;
            }
            case(4):
            {
                mSortStateAudio=Year;
                break;
            }
            }
            sortTreeViewAudio();

        }
        else
        {
            isParentAudio=false;
            sortTreeViewVideo();
            ui->treeWidgetLibraryDisplay->showColumn(10);
            ui->treeWidgetLibraryDisplay->hideColumn(1);
            ui->treeWidgetLibraryDisplay->hideColumn(2);
            ui->treeWidgetLibraryDisplay->hideColumn(3);
            ui->treeWidgetLibraryDisplay->hideColumn(5);
            ui->treeWidgetLibraryDisplay->hideColumn(6);
            ui->treeWidgetLibraryDisplay->hideColumn(7);
            ui->treeWidgetLibraryDisplay->hideColumn(8);
            ui->treeWidgetLibraryDisplay->hideColumn(9);
        }

    }
    else
    {
        if(ui->treeWidgetCatergoryChooser->currentItem()->text(0)=="Video")
        {
            isParentAudio=false;

        }
        else
        {
            isParentAudio=true;
        }
    }

}

void MainWindow::treeLibraryDisplay_Addto_Queue()
{
    QTreeWidgetItem *vItem,*vTemp;
```

```
    QTreeWidgetItemIterator vItemIterator(ui->treeWidgetQueue);
    bool doItemExist=false;

    for(QList<QListWidget*>::size_type i = 0; i<ui->treeWidgetLibraryDisplay-
>selectedItems().size(); ++i)
    {
        vTemp=ui->treeWidgetLibraryDisplay->selectedItems().at(i);
        vItemIterator=QTreeWidgetItemIterator(ui->treeWidgetQueue);
        if(vTemp->parent()==NULL)
        {
            long j=0;
            if(ui->treeWidgetQueue->topLevelItemCount()==0)
            {
                vItem = new QTreeWidgetItem(ui->treeWidgetQueue);
                vItem->setText(0,vTemp->child(0)->text(0));
                vItem->setText(1,vTemp->child(0)->text(10));
                treeWidgetQueue_onDoubleClicked(vItem);
                j=1;
            }

            for(;j<vTemp->childCount();++j)
            {
                while(*vItemIterator)
                {
                    if( (*vItemIterator)->text(1) == vTemp->child(j)->text(10) )
                    {
                        doItemExist=true;
                        break;
                    }
                    vItemIterator++;
                }
                if(doItemExist==false)
                {
                    vItem = new QTreeWidgetItem(ui->treeWidgetQueue);
                    vItem->setText(0,vTemp->child(j)->text(0));
                    vItem->setText(1,vTemp->child(j)->text(10));
                }
                doItemExist=false;
            }

        }
        else
        {
            if(ui->treeWidgetQueue->topLevelItemCount()==0)
            {
                vItem = new QTreeWidgetItem(ui->treeWidgetQueue);
                vItem->setText(0,vTemp->text(0));
                vItem->setText(1,vTemp->text(10));
                treeWidgetQueue_onDoubleClicked(vItem);

            }
            else
            {
                while(*vItemIterator)
                {
                    if( (*vItemIterator)->text(1) == vTemp->text(10) )
                    {
                        doItemExist=true;
                        break;
                    }
                    vItemIterator++;
                }
                if(doItemExist==false)
                {
                    vItem = new QTreeWidgetItem(ui->treeWidgetQueue);
                    vItem->setText(0,vTemp->text(0));
                    vItem->setText(1,vTemp->text(10));
                }
            }
```

```
        }
    }
}

void MainWindow::shutdown()
{
    MainWindow::close();
}

void MainWindow::toggleMute()
{
    if(ui->toolButtonVolume->isChecked()==true)
    {
        mCurrentVolume=ui->horizontalSliderVolume->value();
        setVolume(0);
        ui->toolButtonVolume->setIcon(mVolumeMuted);
    }
    else
    {
        setVolume(mCurrentVolume);
        ui->toolButtonVolume->setIcon(mVolume);
    }
}

void MainWindow::toFullScreen()
{
    isVideoModeON=true;
    //emit(setVideoWidget(mGstDriver));
    emit(goFullScreen());
    mVideoWidget->show();
    MainWindow::hide();
}

void MainWindow::toVideoMode()
{
    isVideoModeON=true;
    //    emit(setVolume(ui->horizontalSliderVolume->value()));

    MainWindow::hide();
    mVideoWidget->show();
}

void MainWindow::setVideoMode(bool vValue)
{
    isVideoModeON=vValue;
    MainWindow::show();
}

void MainWindow::pushButtonUpdate_clicked()
{
    mDBAudio->setManagerOnline();
    mDBAudio->start(QThread::HighestPriority);
    mDBVideo->setManagerOnline();
    mDBVideo->start(QThread::HighestPriority);
    mManagerTimerCounter=1;
}

void MainWindow::runManager()
{
    if(mManagerTimerCounter < 7)
    {
        mDBAudio->setManagerOnline();
        mDBAudio->start(QThread::HighestPriority);
        mDBVideo->setManagerOnline();
        mDBVideo->start(QThread::HighestPriority);
        mManagerTimerCounter++;
        mManagerTimer.start(mManagerTimerValue*mManagerTimerCounter);
    }
}
```

```
void MainWindow::treeWidgetQueue_onDoubleClicked(QTreeWidgetItem * vItem)
{
    mNowPlaying = vItem;

    QTreeWidgetItemIterator vItemIterator(ui->treeWidgetQueue);

    while(*vItemIterator)
    {
        if( (*vItemIterator)->text(1) != vItem->text(1) )
        {
            (*vItemIterator)->setBackgroundColor(0,Qt::white);
            (*vItemIterator)->setSelected(false);
        }
        else
        {
            (*vItemIterator)->setBackgroundColor(0,Qt::green);
            (*vItemIterator)->setSelected(true);
            mNowPlaying=(*vItemIterator);
        }
        vItemIterator++;
    }
    emit(setStopState());
    mGstDriver->setPath(vItem->text(1));
    emit(setPlayState());
    if(isParentAudio==false)
    {

        emit(setNowPlayingVideo(vItem->text(0)));
        toVideoMode();
    }
    else
    {
        ui->labelNowPlaying->setText(vItem->text(0));
        emit(setNowPlayingVideo(vItem->text(0)));
    }


}

void MainWindow::toolButtonClearQueue_Clicked()
{
    ui->treeWidgetQueue->clear();
    emit(setStopState());
    mGstDriver->setPath("");
}

void MainWindow::treeQueue_RemoveFromQueue()
{
    qDebug()<<ui->treeWidgetQueue->selectedItems().at(0)->text(1);
    if(ui->treeWidgetQueue->selectedItems().at(0)->text(1)==mNowPlaying->text(1))
    {
        emit(setStopState());
        mNowPlaying=ui->treeWidgetQueue->itemBelow(ui->treeWidgetQueue-
>selectedItems().at(0));
        mGstDriver->setPath("");
        emit(setPlayState());
    }
    qDeleteAll(ui->treeWidgetQueue->selectedItems());
}
```