# Файл mainwindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QPushButton>

#include <QVBoxLayout>

#include <QApplication>

#include <QMediaPlayer>

#include <QVideoWidget>

#include <QSlider>

#include <QListWidgetItem>

#include "timelabel.h"

#include <QSettings>

#include "PlaylistWidget.h"

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow {

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~*MainWindow*();

private slots:

void openFile();

void loadVideo(const QString &videoFilePath);

void onPlaylistItemClicked(const QString &videoFilePath);

void togglePlayPause();

void on\_btnStop\_clicked();

void on\_btnBackward\_clicked();

void on\_btnForward\_clicked();

void on\_btnShowPlaylist\_clicked();

void loadPlaylist();

void savePlaylist(const QString &playlistPath, const QList<QString> &filePaths);

void saveCurrentPlaylist();

void openFolderAndAddToPlaylist();

void onSpeedSliderChanged(int value);

void setPlaybackSpeed025();

void setPlaybackSpeed05();

void setPlaybackSpeed1();

void setPlaybackSpeed15();

void setPlaybackSpeed175();

void setPlaybackSpeed2();

void onVolumeSliderChanged(int value);

void increaseVolume();

void decreaseVolume();

void offVolume();

void updateCurrentTime(qint64 position);

void updateTotalDuration(qint64 duration);

void saveLastVideoPosition();

void loadLastVideoPosition();

void *closeEvent*(QCloseEvent \*event);

void closeApp();

void showHelpMessage();

private:

Ui::MainWindow \*ui;

QMediaPlayer \*videoPlayer;

QVideoWidget \*videoWidget;

QWidget \*videoContainer;

QWidget \*controlWidget;

QSlider \*slider;

QSlider \*volume;

QSlider \*speedSlider;

qint64 Mduration;

TimeLabel \*startTimeLabel;

TimeLabel \*endTimeLabel;

PlaylistWidget \*playlistWidget;

bool isPlaylistVisible;

QMap<QString, qint64> videoPositions;

QString lastVideoFilePath;

QSettings \*appSettings;

};

#endif

# Файл PlaylistWidget.h

# #ifndef PLAYLISTWIDGET\_H

# #define PLAYLISTWIDGET\_H

# #include <QListWidget>

# #include <QMediaPlaylist>

# #include <QFileInfo>

# class PlaylistWidget : public QListWidget

# {

# Q\_OBJECT

# public:

# explicit PlaylistWidget(QWidget \*parent = nullptr);

# ~PlaylistWidget();

# void addItem(const QString &folderPath, const QString &videoFileName);

# QStringList getPlaylist() const;

# signals:

# void playlistItemClicked(const QString &videoFilePath);

# private slots:

# void handleItemClick(QListWidgetItem \*item);

# private:

# QMediaPlaylist \*playlist;

# };

# #endif

# Файл timelabel.h

#ifndef TIMELABEL\_H

#define TIMELABEL\_H

#include <QLabel>

#include <QMediaPlayer>

#include <QTime>

class TimeLabel : public QLabel

{

Q\_OBJECT

public:

explicit TimeLabel(QWidget \*parent = nullptr);

void setMediaPlayer(QMediaPlayer \*player);

void updateCurrentTime();

void updateTotalDuration();

private:

QMediaPlayer \*mediaPlayer;

qint64 videoDuration;

};

#endif

Файл main.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

Файл mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QFileDialog>

#include <QSlider>

#include "timelabel.h"

#include "PlaylistWidget.h"

#include "PlaylistWidget.cpp"

#include <QLabel>

#include <QFileInfoList>

#include <QFile>

#include <QSettings>

#include <QMessageBox>

#include <QKeyEvent>

#include <iostream>

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent), ui(new Ui::MainWindow), playlistWidget(new PlaylistWidget(this))

{

ui->setupUi(this);

setWindowTitle("VideoPlayer");

isPlaylistVisible = true;

videoPlayer = new QMediaPlayer(this);

videoWidget = new QVideoWidget(this);

speedSlider = new QSlider(Qt::Horizontal, this);

speedSlider->setRange(50, 200);

speedSlider->setValue(100);

speedSlider->setFixedWidth(100);

QFrame \*frame = new QFrame(this);

frame->setStyleSheet("QFrame { background-color: #3498db; }");

frame->setFrameShape(QFrame::Box);

frame->setFrameShadow(QFrame::Sunken);

QVBoxLayout \*centralLayout = new QVBoxLayout;

QHBoxLayout \*buttonLayout = new QHBoxLayout;

QHBoxLayout \*sliderLayout = new QHBoxLayout;

QSlider \*slider = new QSlider(Qt::Horizontal, this);

slider->setRange(0, 100);

slider->setValue(0);

volume = new QSlider(Qt::Horizontal, this);

volume->setRange(0, 100);

volume->setValue(50);

volume->setFixedWidth(100);

QFrame \*fame = new QFrame(this);

fame->setStyleSheet("QFrame { background-color: #0C2558; }");

fame->setFrameShape(QFrame::Box);

fame->setFrameShadow(QFrame::Raised);

QLabel \*volumeLabel = new QLabel("Volume", this);

QLabel \*speedLabel = new QLabel("Speed", this);

volumeLabel->setFixedSize(60, 30);

speedLabel->setFixedSize(55, 30);

volumeLabel->setStyleSheet("color: white;");

speedLabel->setStyleSheet("color: white;");

const QSize buttonSize(100, 30);

ui->btnPlay->setFixedSize(buttonSize);

ui->btnStop->setFixedSize(buttonSize);

ui->btnBackward->setFixedSize(buttonSize);

ui->btnForward->setFixedSize(buttonSize);

ui->btnShowPlaylist->setFixedWidth(80);

videoPlayer->setVideoOutput(videoWidget);

videoWidget->setStyleSheet("background-color: #000000;");

const int playlistWidth = 130;

playlistWidget->setFixedWidth(playlistWidth);

QWidget \*videoAndPlaylistContainer = new QWidget(this);

QHBoxLayout \*videoAndPlaylistLayout = new QHBoxLayout(videoAndPlaylistContainer);

videoAndPlaylistLayout->addWidget(videoWidget);

videoAndPlaylistLayout->addWidget(playlistWidget);

centralLayout->addWidget(videoAndPlaylistContainer);

centralLayout->addWidget(frame);

centralLayout->addWidget(fame);

frame->setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Fixed);

fame->setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Fixed);

ui->btnPlay->setFont(QFont("Arial", 10));

ui->btnStop->setFont(QFont("Arial", 10));

ui->btnBackward->setFont(QFont("Arial", 10));

ui->btnForward->setFont(QFont("Arial", 10));

buttonLayout->addWidget(speedLabel);

buttonLayout->addWidget(speedSlider);

buttonLayout->addWidget(ui->btnShowPlaylist);

buttonLayout->addWidget(ui->btnBackward);

buttonLayout->addWidget(ui->btnPlay);

buttonLayout->addWidget(ui->btnStop);

buttonLayout->addWidget(ui->btnForward);

buttonLayout->addWidget(volumeLabel);

buttonLayout->addWidget(volume);

startTimeLabel = new TimeLabel(this);

endTimeLabel = new TimeLabel(this);

startTimeLabel->setMediaPlayer(videoPlayer);

endTimeLabel->setMediaPlayer(videoPlayer);

sliderLayout->addWidget(startTimeLabel);

sliderLayout->addWidget(slider);

sliderLayout->addWidget(endTimeLabel);

setCentralWidget(new QWidget(this));

centralWidget()->setLayout(centralLayout);

fame->setLayout(sliderLayout);

centralLayout->addWidget(fame);

frame->setLayout(buttonLayout);

centralLayout->addWidget(frame);

buttonLayout->addWidget(ui->videoLabel);

startTimeLabel->setText("00:00:00");

endTimeLabel->setText("---:---");

ui->btnPlay->setIcon(style()->standardIcon(QStyle::SP\_MediaPlay));

ui->btnStop->setIcon(style()->standardIcon(QStyle::SP\_MediaStop));

ui->btnBackward->setIcon(style()->standardIcon(QStyle::SP\_MediaSeekBackward));

ui->btnForward->setIcon(style()->standardIcon(QStyle::SP\_MediaSeekForward));

ui->btnPlay->setText("Play");

ui->btnStop->setText("Stop");

this->menuBar()->setStyleSheet("QMenuBar {"

" background-color: #0C1D51;"

" color: white;"

"}");

this->setStyleSheet("QMenuBar::item:selected {"

" background-color: #1E378B;"

"}");

this->setStyleSheet("QMainWindow {"

" border: 2px solid #0C1D51;"

" background-color: #1E378B;"

"}");

ui->videoLabel->setStyleSheet("QLabel {"

" background-color: #0C2558;"

" padding: 5px;"

" color: white;"

" font-weight: bold;" "}");

startTimeLabel->setStyleSheet("QLabel {"

" background-color: #0C2558;"

" padding: 5px;"

" color: white;"

" font-weight: bold;"

"}");

endTimeLabel->setStyleSheet("QLabel {"

" background-color: #0C2558;"

" padding: 5px;"

" color: white;"

" font-weight: bold;"

"}");

connect(ui->btnStop, &QPushButton::clicked, this, &MainWindow::on\_btnStop\_clicked);

connect(ui->btnBackward, &QPushButton::clicked, this, &MainWindow::on\_btnBackward\_clicked);

connect(ui->btnForward, &QPushButton::clicked, this, &MainWindow::on\_btnForward\_clicked);

connect(ui->actionOpen, &QAction::triggered, this, &MainWindow::openFile);

connect(ui->btnPlay, &QPushButton::clicked, this, &MainWindow::togglePlayPause);

connect(ui->Stop, &QAction::triggered, this, &MainWindow::on\_btnStop\_clicked);

connect(ui->PlayPause, &QAction::triggered, this, &MainWindow::togglePlayPause);

connect(ui->Backward, &QAction::triggered, this, &MainWindow::on\_btnBackward\_clicked);

connect(ui->Forward, &QAction::triggered, this, &MainWindow::on\_btnForward\_clicked);

connect(ui->OpenPlaylist, &QAction::triggered, this, &MainWindow::loadPlaylist);

connect(volume, SIGNAL(valueChanged(int)),videoPlayer, SLOT(setVolume(int)));

connect(volume, &QSlider::valueChanged, this, &MainWindow::onVolumeSliderChanged);

connect(videoPlayer, &QMediaPlayer::durationChanged, slider, &QSlider::setMaximum);

connect(videoPlayer, &QMediaPlayer::positionChanged, slider, &QSlider::setValue);

connect(slider, &QSlider::sliderMoved,videoPlayer,&QMediaPlayer::setPosition);

connect(ui->actionIncreaseVolume, &QAction::triggered, this, &MainWindow::increaseVolume);

connect(ui->actionDecreaseVolume, &QAction::triggered, this, &MainWindow::decreaseVolume);

connect(ui->actionoffVolume, &QAction::triggered, this, &MainWindow::offVolume);

connect(ui->actionSpeed025, &QAction::triggered, this, &MainWindow::setPlaybackSpeed025);

connect(ui->actionSpeed05, &QAction::triggered, this, &MainWindow::setPlaybackSpeed05);

connect(ui->actionSpeed1, &QAction::triggered, this, &MainWindow::setPlaybackSpeed1);

connect(ui->actionSpeed15, &QAction::triggered, this, &MainWindow::setPlaybackSpeed15);

connect(ui->actionSpeed175, &QAction::triggered, this, &MainWindow::setPlaybackSpeed175);

connect(ui->actionSpeed2, &QAction::triggered, this, &MainWindow::setPlaybackSpeed2);

connect(speedSlider, &QSlider::valueChanged, this, &MainWindow::onSpeedSliderChanged);

connect(ui->actionSavePlaylist, &QAction::triggered, this, &MainWindow::saveCurrentPlaylist);

connect(ui->OpenFolder, &QAction::triggered, this, &MainWindow::openFolderAndAddToPlaylist);

connect(playlistWidget, &PlaylistWidget::playlistItemClicked, this, &MainWindow::onPlaylistItemClicked);

connect(videoPlayer, &QMediaPlayer::durationChanged, this, &MainWindow::updateTotalDuration);

connect(videoPlayer, &QMediaPlayer::positionChanged, this, &MainWindow::updateCurrentTime);

connect(ui->actionHelp, &QAction::triggered, this, &MainWindow::showHelpMessage);

connect(ui->Exit, &QAction::triggered, this, &MainWindow::closeApp);

ui->videoLabel->setText("File name");

ui->videoLabel->show();

loadLastVideoPosition();

}

MainWindow::~MainWindow() {

saveLastVideoPosition();

delete ui;

}

void MainWindow::closeApp(){

QCoreApplication::exit(0);

}

void MainWindow::openFile() {

QString fileName = QFileDialog::getOpenFileName(this, "Open Video File");

if (!fileName.isEmpty()) {

videoPlayer->setMedia(QUrl::fromLocalFile(fileName));

loadVideo(fileName);}

}

void MainWindow::loadVideo(const QString &videoFilePath) {

lastVideoFilePath = videoFilePath;

videoPlayer->setMedia(QUrl::fromLocalFile(videoFilePath));

QFileInfo fileInfo(videoFilePath);

QString filename = fileInfo.fileName();

ui->videoLabel->setText("Now Playing: " + filename);

playlistWidget->addItem(fileInfo.path(), filename);

togglePlayPause();

}

void MainWindow::onPlaylistItemClicked(const QString &videoFilePath)

{

lastVideoFilePath = videoFilePath;

videoPlayer->setMedia(QUrl::fromLocalFile(videoFilePath));

QFileInfo fileInfo(videoFilePath);

QString filename = fileInfo.fileName();

videoPlayer->play();

videoPlayer->setPosition(0);

ui->videoLabel->setText("Now Playing: " + filename);

}

void MainWindow::togglePlayPause() {

if (videoPlayer->state() == QMediaPlayer::PlayingState) {

videoPlayer->pause();

} else {

videoPlayer->play();

}

}

void MainWindow::on\_btnStop\_clicked() {

videoPlayer->stop();

}

void MainWindow::updateCurrentTime(qint64 position) {

startTimeLabel->updateCurrentTime();

Q\_UNUSED(position);

}

void MainWindow::updateTotalDuration(qint64 duration) {

endTimeLabel->updateTotalDuration();

Q\_UNUSED(duration);

}

void MainWindow::on\_btnBackward\_clicked()

{

qint64 newPosition = videoPlayer->position() - 5000;

videoPlayer->setPosition(qMax(newPosition, qint64(0)));

qDebug() << "Backward button clicked";

}

void MainWindow::on\_btnForward\_clicked()

{

qint64 newPosition = videoPlayer->position() + 5000;

videoPlayer->setPosition(qMin(newPosition, videoPlayer->duration()));

qDebug() << "Forward button clicked";

}

void MainWindow::on\_btnShowPlaylist\_clicked()

{

if(!isPlaylistVisible)

{

playlistWidget->hide();

isPlaylistVisible = true;

}

else

{

playlistWidget->show();

isPlaylistVisible = false;

}

}

void MainWindow::loadPlaylist() { QString playlistFilePath = QFileDialog::getOpenFileName(this, "Open Playlist", QString(), "Playlist Files (\*.m3u \*.m3u8)");

if (!playlistFilePath.isEmpty()) {

if (videoPlayer->playlist()) {

videoPlayer->playlist()->clear();

} else {

videoPlayer->setPlaylist(new QMediaPlaylist(videoPlayer)); }

videoPlayer->playlist()->load(QUrl::fromLocalFile(playlistFilePath));

playlistWidget->clear();

for (int i = 0; i < videoPlayer->playlist()->mediaCount(); ++i) {

QMediaContent mediaContent = videoPlayer->playlist()->media(i);

QString filePath = QUrl(mediaContent.request().url()).toLocalFile();

QFileInfo fileInfo(filePath);

QString folderPath = fileInfo.path();

QString FileName = fileInfo.fileName();

QString videoFileName = QUrl::fromPercentEncoding(FileName.toUtf8());

playlistWidget->addItem(folderPath, videoFileName);

qDebug() << FileName;

qDebug() << videoFileName;

}

ui->videoLabel->setText("Playlist Loaded: " + QFileInfo(playlistFilePath).fileName());

videoPlayer->setMedia(videoPlayer->playlist()->media(0));

}

}

void MainWindow::onSpeedSliderChanged(int value)

{

qreal playbackRate = static\_cast<qreal>(value) / 100.0;

videoPlayer->setPlaybackRate(playbackRate);

}

void MainWindow::setPlaybackSpeed025()

{

speedSlider->setValue(25);

onSpeedSliderChanged(25);

}

void MainWindow::setPlaybackSpeed05()

{

speedSlider->setValue(50);

onSpeedSliderChanged(50);}

void MainWindow::setPlaybackSpeed1()

{

speedSlider->setValue(100);

onSpeedSliderChanged(100);}

void MainWindow::setPlaybackSpeed15()

{

speedSlider->setValue(150);

onSpeedSliderChanged(150);}

void MainWindow::setPlaybackSpeed175()

{

speedSlider->setValue(175);

onSpeedSliderChanged(175);}

void MainWindow::setPlaybackSpeed2()

{

speedSlider->setValue(200);

onSpeedSliderChanged(200);}

void MainWindow::closeEvent(QCloseEvent \*event) {

saveLastVideoPosition();

event->accept();}

void MainWindow::increaseVolume() {

int currentVolume = volume->value();

int newVolume = qMin(currentVolume + 10, 100); volume->setValue(newVolume);

videoPlayer->setVolume(newVolume);}

void MainWindow::decreaseVolume() {

int currentVolume = volume->value();

int newVolume = qMax(currentVolume - 10, 0); volume->setValue(newVolume);

videoPlayer->setVolume(newVolume);}

void MainWindow::offVolume() {

volume->setValue(0);

videoPlayer->setVolume(0);}

void MainWindow::onVolumeSliderChanged(int value) {

videoPlayer->setVolume(value);}

void MainWindow::saveLastVideoPosition() {

QSettings settings("YourOrganizationName", "YourAppName");

settings.setValue("lastVideoFilePath", lastVideoFilePath);

settings.setValue("lastVideoPosition", videoPlayer->position());

qDebug() << "Video position saved for file: " << lastVideoFilePath << ", Position: " << videoPlayer->position();

}

void MainWindow::loadLastVideoPosition() { QSettings settings("YourOrganizationName", "YourAppName");

QString savedFilePath = settings.value("lastVideoFilePath").toString();

if (!savedFilePath.isEmpty() && QFile::exists(savedFilePath)) {

qint64 savedPosition = settings.value("lastVideoPosition", 0).toLongLong();

QMessageBox::StandardButton reply = QMessageBox::question(

this,

"Load Last Video Position",

"Do you want to load the last video position?",

QMessageBox::Yes | QMessageBox::No

);

if (reply == QMessageBox::Yes) {

lastVideoFilePath = savedFilePath;

QFileInfo fileInfo(savedFilePath);

QString filename = fileInfo.fileName();

playlistWidget->addItem(fileInfo.path(), filename);

videoPlayer>setMedia(QUrl::fromLocalFile(savedFilePath));

videoPlayer->setPosition(savedPosition);

ui->videoLabel->setText("Now Playing: " + filename);

togglePlayPause();

qDebug() << "Video position loaded for file: " << savedFilePath << ", Position: " << savedPosition;

} else {

qDebug() << "User chose not to load the last video position.";

}

} else {

qDebug() << "No saved video position found.";

}

}

void MainWindow::savePlaylist(const QString &playlistPath, const QList<QString> &filePaths) {

QFile playlistFile(playlistPath);

if (playlistFile.open(QIODevice::WriteOnly | QIODevice::Text)) {

QTextStream stream(&playlistFile);

// Write the first two lines

stream << "#EXTM3U" << Qt::endl;

// Iterate through the file paths and write the necessary lines

for (const QString &filePath : filePaths) {

QFileInfo fileInfo(filePath);

QString fileName = fileInfo.fileName();

//qint64 durationInSeconds = 0; // Replace with actual duration retrieval logic

// Write the EXTINF line

stream << "#EXTINF:" << "," << fileName << Qt::endl;

// Write the file path (with percent encoding for spaces)

stream << QUrl::toPercentEncoding(fileName) << Qt::endl;

qDebug() << QUrl::toPercentEncoding(fileName);

}

playlistFile.close();

} else {

QMessageBox::warning(this, "Error", "Unable to save playlist.");

}

}

void MainWindow::saveCurrentPlaylist() {

QString playlistPath = QFileDialog::getSaveFileName(this, "Save Playlist", QString(), "Playlist Files (\*.m3u \*.m3u8)");

if (!playlistPath.isEmpty()) {

QList<QString> filePaths;

for (int i = 0; i < playlistWidget->count(); ++i) {

QListWidgetItem \*item = playlistWidget->item(i);

QString encodedFolderPath = item->data(Qt::UserRole).toString();

QString folderPath = QUrl::fromPercentEncoding(encodedFolderPath.toUtf8());

QString videoFileName = item->text();

QString filePath = QDir(folderPath).filePath(videoFileName);

filePaths.append(filePath);

}

savePlaylist(playlistPath, filePaths);

QMessageBox::information(this, "Playlist Saved", "Playlist has been saved successfully.");

}

}

void MainWindow::openFolderAndAddToPlaylist() {

QString folderPath = QFileDialog::getExistingDirectory(this, "Open Folder", QString());

if (!folderPath.isEmpty()) {

QDir dir(folderPath);

QStringList nameFilters;

nameFilters << "\*.mp4" << "\*.avi" << "\*.mkv";

QStringList videoFiles = dir.entryList(nameFilters, QDir::Files);

for (const QString &videoFileName : videoFiles) {

playlistWidget->addItem(folderPath, videoFileName);

}

QMessageBox::information(this, "Videos Added", "Videos from the selected folder have been added to the playlist.");

}

}

void MainWindow::showHelpMessage() {

QString helpMessage = "This is a video player application. "

"You can open a video file, control playback, adjust volume, and manage playlists.";

QMessageBox::information(this, "Справка", helpMessage);

}