Asta CMS End Application

**Overview**

This project is the end application for Asta CMS, developed using Qt for Windows. It manages the digital signage content display and integration.

**Table of Contents**

**Features**

- \*\*Digital Signage Management\*\*: Display and manage digital signage content.

- \*\*Content Integration\*\*: Integrate various types of media content.

- \*\*User Interface\*\*: User-friendly interface developed using Qt.

**Requirements**

- \*\*Operating System\*\*: Windows 10 or higher

- \*\*Qt Version\*\*: Qt 5.15.2

- \*\*Compiler\*\*: MSVC 2019

**Installation**

**Prerequisites**

1. Install [Qt](https://www.qt.io/download) (version 5.15 or higher).

2. Ensure you have a suitable compiler installed (e.g., MSVC 2019 or MinGW).

**Build Instructions**

1. Clone the repository: WILL BE PROVIDED LATER

2. Open the project in Qt Creator.

3. Configure the project using the appropriate Qt kit.

4. Build the project and click the run button to execute the application.

**Running Tests**

To ensure the application is functioning correctly for the mission 1, you can run tests for various functionalities. Below are the instructions for testing each specific functionality.

**Prerequisites**

1. Ensure the application is built and deployed correctly.

2. Have access to the necessary hardware (e.g., external storage devices) and permissions.

**Test Instructions**

1. Read Media Video

* Launch the application.
* Click on download the video button.
* Video will be downloaded in a specific path and it will be displayed in the UI.

2. Write External Storage

* Launch the application.
* Click on Play the video button.
* The downloaded video will be played in the UI.

3. Read External Storage

* Launch the application.
* Click on Write Extrenal Storage Button.
* Image will be downloaded in a specific path and it will be displayed in the UI.

4. Receive Boot Completed

* Ensure the application is set to start on boot.
* 2. Restart the device.
* 3. Verify that the application starts automatically upon booting.

5. Foreground Service

* uninstal the application.
* Verify that the application try to print the internet connection status in UI and it will uninstall the app

6. Access Network State

* Launch the application.
* Disconnect and reconnect the network.
* Verify that the application detects the network state changes and toast a message accordingly.

7. Write Settings

* Turn sounf off
* Turn sound on
* Delete Device data and restart
* Restart the app
* Turn UI off
* Turn UI on
* adjust player brightness
* adjust player sound volume
* Launch the application.
* Try to click all the write settings button listed above.
* Verify that the system setting is successfully changed.

**Code File Example (main.cpp)**

This `main.cpp` file demonstrates the initialization and configuration of the Asta CMS End Application, including logging setup, translator installation, startup configuration, and the main application execution loop.

**Function Definitions**

**`void make\_startup\_app()`**

* This function configures the application to run at startup by adding an entry to the Windows registry.

**void configureModernLogging()**

* This function sets up logging for the application using a custom log formatter and log file handler.

#### void installTranslator()

* This function installs a translator for the application to support multiple languages.

### **Main Function**

The main function initializes and runs the Asta CMS End Application.

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

{

// the below line should be the first in main to write the logs properly

configureModernLogging();

int currentExitCode = 0;

do

{

LOG\_INFO("Executing Main Function");

LOG\_INFO(QString("Command Line Argument Count %1").arg(argc));

QStringList list;

for (int a = 0; a < argc; ++a)

{

LOG\_INFO(QString("Command Line Argument Index %1").arg(a));

LOG\_INFO(QString("Command Line Argument Index's value %1").arg( QString::fromLocal8Bit(argv[a])));

}

SingleApplication astaCMSApp(*argc*, argv);

astaCMSApp.setWindowIcon(QIcon("://resources/ico.ico"));

astaCMSApp.setQuitOnLastWindowClosed(false);

make\_startup\_app();

installTranslator();

QFile SystemScreenStatusLogFile(AppSettings::defaultAppDataLocation()+"/SystemScreenStatus.txt");

if(!SystemScreenStatusLogFile.*open*(QIODevice::WriteOnly | QIODevice::Append))

{

LOG\_ERROR(0,"Failed to create logs file for writeonly " + SystemScreenStatusLogFile.*fileName*());

}

QTextStream stream(&SystemScreenStatusLogFile);

stream << "SYSTEM SCREEN ON - " << QTime::currentTime().toString("hh:mm:ss")<<endl;

SystemScreenStatusLogFile.flush();

SystemScreenStatusLogFile.*close*();

QThread::currentThread()->setObjectName("Main GUI");

LOG\_INFO("Main GUI");

AppSettings::get().initSettings();

LOG\_INFO("AstaCMS App Settings Initiated");

OnlineWatcher::startThread();

QTimer::singleShot(10000,[&]

{

LOG\_INFO("Thread is " + QThread::currentThread()->objectName());

});

MainWindow mainView;

HibernationClock hibernationClock;

ApiBroker apiBroker;

QObject::connect( &astaCMSApp, &SingleApplication::instanceStarted, [ &]()

{

LOG\_INFO(QString("%1").arg(astaCMSApp.isPrimary()));

LOG\_INFO("Thread is " + QThread::currentThread()->objectName());

// mainView.silentInfo("Application is already running!");

});

LOG\_INFO("App is about to start");

Controller controller(&mainView,&apiBroker,&hibernationClock);

controller.appStarting();

currentExitCode = astaCMSApp.exec();

OnlineWatcher::stopThread();

} while( currentExitCode == EXIT\_CODE\_REBOOT );

return currentExitCode ;

}

* Initializes SingleApplication.
* Sets the application icon and ensures the app does not quit when the last window is closed.
* Calls make\_startup\_app() and installTranslator() to set up startup and translation.
* Opens a log file to track system screen status.
* Starts the OnlineWatcher thread.
* Sets a timer to log the current thread's name after 10 seconds.
* Initializes the main application window (MainWindow), hibernation clock (HibernationClock), and API broker (ApiBroker).
* Connects the instanceStarted signal of SingleApplication to log when the application is already running.
* Starts the Controller and runs the application event loop.
* Restarts the application if currentExitCode is set to EXIT\_CODE\_REBOOT.

**Main Window UI (mainwindow.ui)**

The `mainwindow.ui` file defines the user interface for the Asta CMS End Application using Qt. This file is created with Qt Designer and outlines the structure and components of the main window, which manages digital signage content display and integration.

**Overview of UI Components**

* The main window (`QMainWindow`) contains several nested widgets and layouts. Below is a detailed description of the primary UI components and their functionalities.

**Main Window (QMainWindow)**

* + \*\*Class\*\*: MainWindow
  + \*\*Object Name\*\*: MainWindow
  + \*\*Properties\*\*:
  + \*\*geometry\*\*: Defines the size and position of the main window (width: 909, height: 615).
  + \*\*windowTitle\*\*: Sets the window title to "Asta CMS".

**Central Widget (QWidget)**

* \*\*Object Name\*\*: centralwidget
* \*\*Layout\*\*: QVBoxLayout (verticalLayout\_3)
* Manages the vertical layout of child widgets.

**Stacked Widget (QStackedWidget)**

* \*\*Object Name\*\*: stackedWidget
* \*\*Purpose\*\*: Allows switching between multiple pages or widgets.

**Main Page (QWidget)**

* \*\*Object Name\*\*: mainPage
* \*\*Layout\*\*: QHBoxLayout (horizontalLayout\_6)
* Manages the horizontal layout of the main widget and the preview widget.

**Main Widget (QWidget)**

* + **\*\*Object Name\*\*: mainwidget**
  + **\*\*Properties\*\*:**
  + **\*\*maximumSize\*\*: Limits the width to 400 pixels.**
  + **\*\*Layout\*\*: QVBoxLayout (verticalLayout)**
  + **Contains several functional widgets in a vertical layout**.

**Download Widget (QWidget)**

* + \*\*Object Name\*\*: downloadWidget
  + \*\*Layout\*\*: QHBoxLayout (horizontalLayout\_5)
  + \*\*downloadWidgetLabel\*\*: QLabel displaying "Download the video".
  + \*\*downloadWidgetButton\*\*: QPushButton labeled "Download the video".

**Play Widget (QWidget)**

* + \*\*Object Name\*\*: playWidget
  + \*\*Layout\*\*: QHBoxLayout (horizontalLayout\_4)
  + \*\*playWidgetLabel\*\*: QLabel displaying "Play the video".
  + \*\*playWidgetButton\*\*: QPushButton labeled "Play the video".

**Write External Storage Widget (QWidget)**

* + \*\*Object Name\*\*: writeExtrenalStorageWidget
  + \*\*Layout\*\*: QHBoxLayout (horizontalLayout)
  + \*\*writeWidgetLabel\*\*: QLabel displaying "Write External Storage".
  + \*\*writeWidgetButton\*\*: QPushButton labeled "Write external storage".

**Read External Storage Widget (QWidget)**

* + \*\*Object Name\*\*: readExtrenalStorageWidget
  + \*\*Layout\*\*: QHBoxLayout (horizontalLayout\_2)
  + \*\*readWidgetLabel\*\*: QLabel displaying "Read External Storage".
  + \*\*readWidgetButton\*\*: QPushButton labeled "Read external storage".

**Mute System Button (QPushButton)**

* + \*\*Object Name\*\*: muteButton
  + \*\*Properties\*\*:
  + \*\*text\*\*: "Mute system"

**Unmute System Button (QPushButton)**

* \*\*Object Name\*\*: unmuteButton
* \*\*Properties\*\*:
* \*\*text\*\*: "Unmute System"

**Delete Device Data and Restart Button (QPushButton)**

* + \*\*Object Name\*\*: deleteDeviceDataAndRestartButton
  + \*\*Properties\*\*:
  + \*\*text\*\*: "Delete Device data and Restart the App"

**Restart App Button (QPushButton)**

* + \*\*Object Name\*\*: restartAppButton
  + \*\*Properties\*\*:
  + \*\*text\*\*: "Restart the App"

**Brightness Widget (QWidget)**

* \*\*Object Name\*\*: brightnessWidget
* \*\*Layout\*\*: QHBoxLayout (horizontalLayout\_3)
* \*\*label\*\*: QLabel displaying "Brightness".
* \*\*brightnessSlider\*\*: QSlider to adjust brightness (horizontal orientation).

**Player Volume Widget (QWidget)**

* \*\*Object Name\*\*: widget
* \*\*Layout\*\*: QHBoxLayout (horizontalLayout\_9)
* \*\*label\_2\*\*: QLabel displaying "Player Volume".
* \*\*playerVolumeSlider\*\*: QSlider to adjust player volume (horizontal orientation).

**Spacer (QSpacerItem)**

* \*\*Object Name\*\*: verticalSpacer
* \*\*Properties\*\*:
* \*\*orientation\*\*: Vertical

**Preview Widget (QWidget)**

* \*\*Object Name\*\*: previewWidget
* \*\*Layout\*\*: QVBoxLayout (verticalLayout\_2)
* Purpose: Reserved area for media preview.

**Black Page (QWidget)**

* \*\*Object Name\*\*: blackPage
* \*\*Layout\*\*: QHBoxLayout (horizontalLayout\_7)
* Purpose: Placeholder for UI blackout functionality.

**Toggle UI Widget (QWidget)**

* \*\*Object Name\*\*: ToggleUIWidget
* \*\*Layout\*\*: QHBoxLayout (horizontalLayout\_8)
* \*\*turnUIOffButton\*\*: QPushButton labeled "Turn UI Off".
* \*\*turnUIOnButton\*\*: QPushButton labeled "Turn UI On".

**Summary**

The `mainwindow.ui` file defines a comprehensive and functional user interface for the Asta CMS End Application. It includes widgets and layouts for media management, system control, and user interaction. This UI can be customized and extended using Qt Designer to meet specific requirements.

### **Documentation for mainwindow.cpp**

This documentation provides an overview of each functionality implemented in the MainWindow class.

**Constructor: MainWindow::MainWindow(QWidget \*parent)**

Initializes the main window and its components.

* Sets up the UI components using the Ui::MainWindow class.
* Initializes labels, video widget, and media player.
* Adds the labels and video widget to the preview widget's layout.
* Sets up brightness and volume sliders with their respective ranges and default values.
* Connects the sliders to their corresponding slots for handling brightness and volume changes.

**Destructor: MainWindow::~MainWindow()**

Cleans up the UI components by deleting the ui pointer.

**Function: bool MainWindow::download()**

Handles the downloading of a video file from a specified URL.

* Sets up the network request and SSL configuration.
* Connects the QNetworkAccessManager signals to handle SSL errors and download completion.
* Writes the downloaded data to a file and updates the UI accordingly.

**Function: bool MainWindow::writeFile(QString filePath, QNetworkReply\* reply)**

Writes the downloaded data to a specified file path.

* Opens the file in write-only mode.
* Writes the data from the network reply to the file.
* Updates the UI to show the file save path.

**Slot: void MainWindow::on\_downloadWidgetButton\_clicked()**

Triggered when the download button is clicked.

* Disables the download button.
* Calls the download() function to start downloading the video.
* Hides the other widgets in preview layout and show only the progress bar

**Slot: void MainWindow::on\_playWidgetButton\_clicked()**

Triggered when the play button is clicked.

* Disables the play button.
* Checks if the video file exists, and if so, plays it using the media player.
* Updates the UI to show the video widget.

.

**Slot: void MainWindow::on\_readWidgetButton\_clicked()**

Triggered when the read button is clicked.

* Disables the read button.
* Checks if the image file exists, and if so, displays it in the image label.
* Updates the UI to show the image label.
* Hides the other widgets in preview layout and show only the Image widget.

**Slot: void MainWindow::on\_writeWidgetButton\_clicked()**

Triggered when the write button is clicked.

* Disables the write button.
* Downloads an image from a specified URL and saves it to a file.
* Updates the UI to show the file save path.
* Hides the other widgets in preview layout and show only the progress bar

**Slot: void MainWindow::on\_muteButton\_clicked()**

Triggered when the mute button is clicked.

* Calls the toggleSystemSounds(true) function to mute system sounds.

**Slot: void MainWindow::on\_unmuteButton\_clicked()**

Triggered when the unmute button is clicked.

* Calls the toggleSystemSounds(false) function to unmute system sounds.

**Function: void MainWindow::toggleSystemSounds(bool mute)**

Toggles system sounds on or off.

* Uses Windows API to get the default audio endpoint and set the mute state based on the mute parameter.

**Function: bool MainWindow::deleteDirectoryContents(const QString &dirPath)**

Deletes all contents within a specified directory.

* Iterates through and removes all files and subdirectories within the given directory path.

**Function: void MainWindow::restartApplication()**

Restarts the application by returning the reboot exit code

* Gets the current application path and arguments.
* Starts a new process with the same path and arguments, and quits the current application.

**Slot: void MainWindow::on\_deleteDeviceDataAndRestartButton\_clicked()**

Triggered when the delete data and restart button is clicked.

* Deletes the application directory contents.
* Restarts the application if the deletion is successful, otherwise shows a warning message.

**Slot: void MainWindow::on\_restartAppButton\_clicked()**

Triggered when the restart button is clicked.

* Calls the restartApplication() function to restart the application.

**Slot: void MainWindow::on\_turnUIOffButton\_clicked()**

Triggered when the turn UI off button is clicked.

* Sets the stacked widget to show a black page.

**Slot: void MainWindow::on\_turnUIOnButton\_clicked()**

Triggered when the turn UI on button is clicked.

* Sets the stacked widget to show the main page.

**Function: void MainWindow::setBrightness(int value)**

Sets the brightness of the system monitor.

* Uses WMI methods to get the monitor handle and set the brightness level.

**Function: void MainWindow::setVolume(int value)**

Sets the volume of the System

Checks if the media player is initialized and the video file exists.

* Sets the media player's volume to the specified value.