Video chat application handling chat and secondary camera support

My project solves a video chat application on Windows Desktop environment (Windows 10).
The project really similar like skype:

there exists a workstation somewhere, and in this workstation a user indicates to start a video chat.
There exists a webservice on a server, which acts like a dispatcher, this will select one of the logged in user on the helpdesk, and redirect the call for that person's workstation.
If the helpdesk accepts the call, then the video chat starts and they can speak.
When the conversation is finished, then a welcome screen appears.

On the client side the program must handle a secondary camera, because if necessary, this will make a picture about papers or small objects, what the helpdesk wants to see. This "take a picture" event should be fired from the helpdesk side. The client does not need to do anything else, just press one button to call somebody on the helpdesk side (the dispatcher webservice selects the appropriate person), and talk, and if necessary, then put the small objects under the secondary camera.

The Helpdesk side application is the same. If he is logged in and set his flag online, then he shall be called by an user, and the dispatcher service will transfer the video call to him. Then it is a peer-to-peer video call.

Facts:

- This is an application to be used for video talking between a dispatcher and a client
- For both side we use the same application, but start with /CLIENT or /DISPATCHER to make difference
- On client mode we need much less functionality and buttons, so they can be hidden, and this app always full screen (kiosk mode) application
- On client side starting a call will be possible to press a button (f.e. Enter)
- Always the client initiates any video call, the dispatcher cannot start a call, just accept it
- There are many clients and many dispatchers, but one client can be connected to that dispatcher, who is associated with the client in an SQL Server database
- if the dispatcher does not answer, there is a central (so called default) dispatcher, and then the call shall be redirected to him: central dispatcher will be always available 24H
-

Progress of using the application

There is a user, who has a kiosk mode computer, where the app is running in full screen mode (without caption bar), and on this screen some text is shown:
“Please press the button for calling the dispatcher”

Then he presses the button (let say, it is an enter button on an attached keyboard), and in this case the following happens:

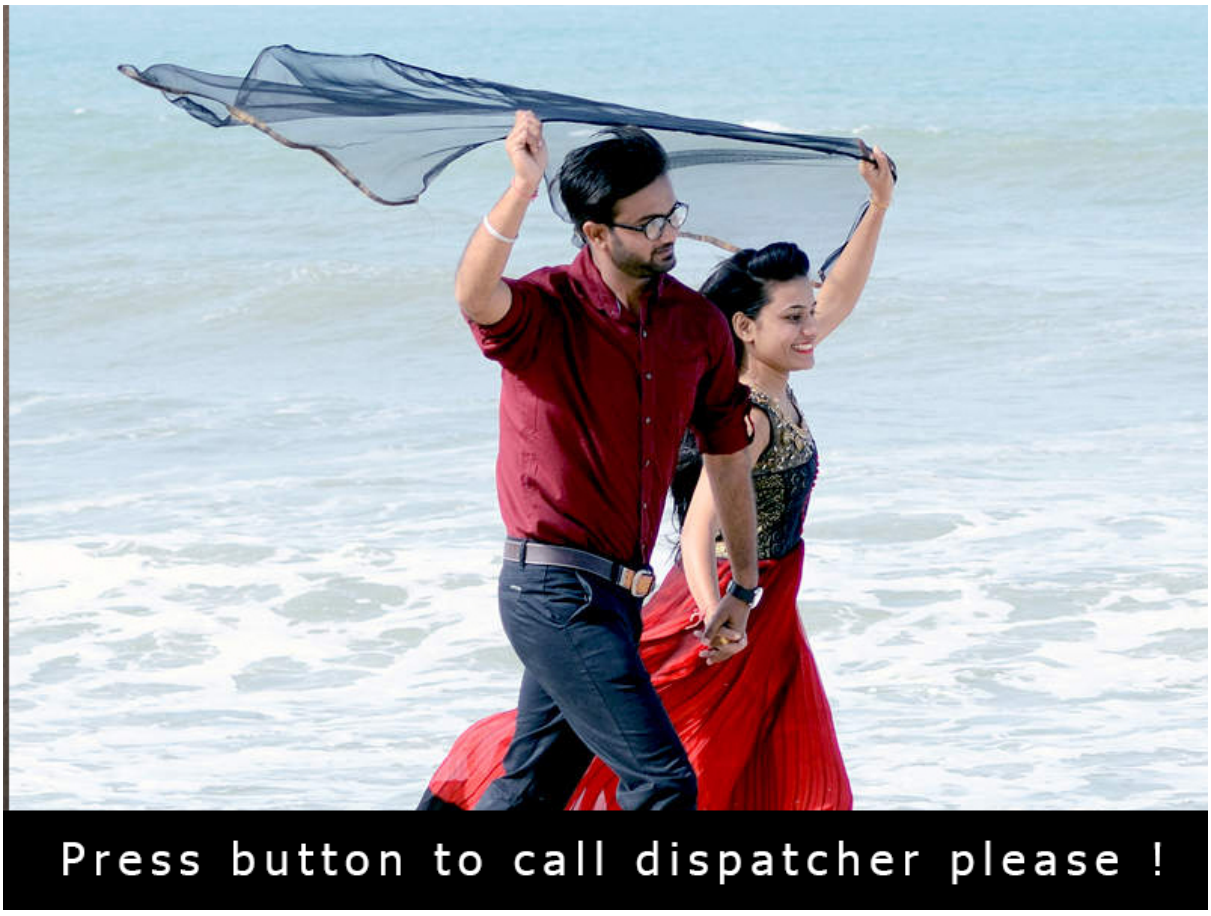
1. The server side component sees, that there is a call request.
2. There is an MS SQL Server database on the server side, where it is stored the client workstation IDs, and also, that workstation ID, where this call must be connected to. So it means, that on the client side it is not necessary to select, who is the called partner, but the server side component will select it from the database, and redirects the call to there -> to the dispatcher person
3. What happens on dispatcher side?
    a. The dispatcher person sees on his screen, there is an incoming call, picks it up and the connection is ready, and they can talk
    b. The dispatcher is not available, or he cannot pick up the headset, or anything happened ->
        i. It will ring for 60 seconds
        ii. After 60 seconds the server side component redirects the call to an other dispatcher, which is a central dispatcher
        This is the same kind of call, but the central dispatcher will be always available. If really not, then after 60 seconds the lines are broken, and the original client shall try it later.
        iii. The central dispatcher will inform (he will call by telephone or using any other media, we do not know it how, but inform that person), the client associated dispatcher, that “where the hell are you, because somebody wants you? :)”
            1. If the associated dispatcher is available -> then the central dispatcher will tell to the client: “Please call again, because the associated dispatcher will now pick up the line”
            Then the central dispatcher breaks the line, and the whole process starts again as usual
            2. If not, then the central dispatcher will say to the client to try again later
4. The conversation between the client and the dispatcher will be started.
5. It can happen, that the dispatcher asks the client, to put his ID card or any other document on the front of the secondary camera. If it is there, the dispatcher clicks on the “Get remote pic” button -> this takes a picture on the client’s machine secondary camera and sends it back to the dispatcher side, where it can be stored somewhere, and also display immediately on the dispatcher workstation.

6. The dispatcher uses the info he sees, makes some administration and keeps talking with the client. This administration is done in an external application, we do not have to care about it.
7. If the dispatcher wants to send written message to the client, then it should be displayed on the client's screen as "Message from dispatcher". Maybe in a 2.0 version we will enable also from client side, but in the frist version it is not necessary, so this functionality can be disabled on client side.
8. When the conversation is ready, the dispatcher breaks the line.
    a. On client side, we turn back to the original state *->app is running in full screen mode (without caption bar), and on this screen some text is shown: "Please press the button for calling the dispatcher"*
    b. On dispatcher side we turn back to the original app state: *waiting for an incoming call*

User interface on the client side

Default screen

When there is nothing, then on the client workstation just a message will appear:



of course instead of this picture some logo screen can be shown.

Online video chat screen

The man on the client side shall look something similar: full screen the dispatcher lady, small box the client man itself, and at the bottom a strip, which is useful for the messages to be displayed.

The call button will be associated with a keyboard button, so screen button does not needed to be displayed, since on the client side we do not plan any mouse or real keyboard, just a single enter button.

User interface on the dispatcher side

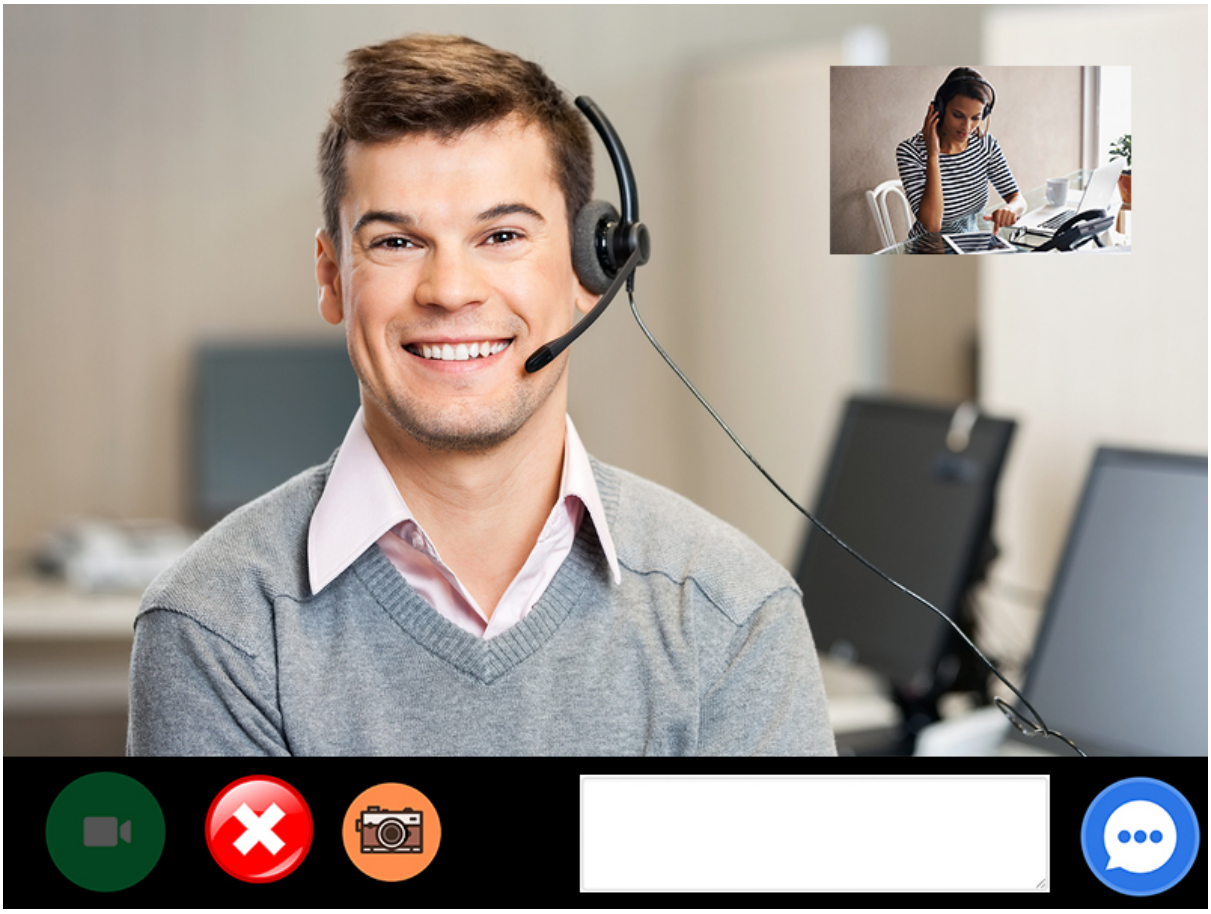
Default screen

When there is nothing, then on the client workstation just a message will appear:

When somebody calling, then it will ring on the sounders, and display an incoming call icon

Screen when talking with somebody

The lady on the dispatcher side shall look something similar: full screen the client man, small box the dispatcher lady itself.

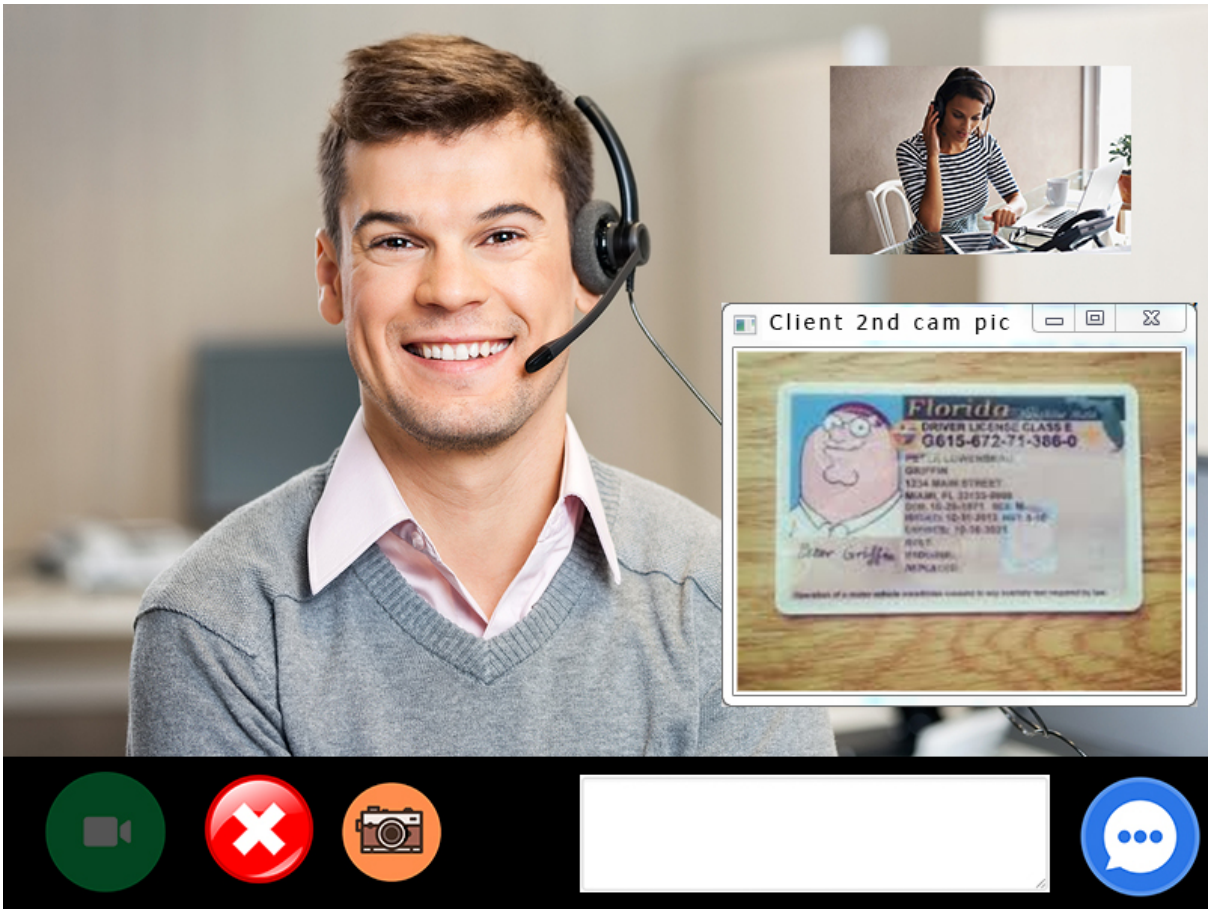


Here the dispatcher can write a message and send it with the blue button.

If he wants to take a picture of the client machine, then the orange button should be pressed.

Screen when talking with somebody and take a pic

When it is necessary, then the lady will ask via speech, to show the ID card from the client man



When the dispatcher lady asked the client man to put his ID card just to the front or under the secondary camera, then presses the orange button, and in that case:

1. Take a pic from the secondary camera on the client side
2. Save it to a directory on the dispatcher's workstation (probably "Downloads" folder, but it shall be customizable in config) with a unique name
3. Show it in a small popup window to see -> use paint.exe or any configurable built in picture viewer app to see the picture (then the dispatcher may zoom or rotate in that application if wants, or if necessary), when finished, the dispatcher will close that app window

Database plan

Since a client call can be put only to one dispatcher (or if no answer, then to the central dispatcher), it is necessary to have a database table, where we store the associations.

Tables can be the following:

Endpoints

Id - int - primary key identity number
Type - int - 0: central dispatcher, 1: dispatcher, 2: client - it is an integer flag
Name - nvarchar(128)
StationId - nvarchar(256) - this identifies the station itself (developer id)
Associated - bit - it is already associated with a workstation

CallLog

Id - int - primary key identity number
ClientEndpointId - int - Foreign key to Endpoints table
DispatcherEndpointId - int - Foreign key to Endpoints table
Start - datetime
End - datetime
PassedToCenter - bit - true if this call has been redirected to the center

Application settings

Somewhere the administrator must configure the application. In the first version it is enough to use a config file, no UI needed.

StationId

This must be the same station id, what we use in the database.

PictureFolder

The folder, where we store the pictures
Default: Downloads

PictureDisplayApp

The application, which is used to open the picture taken on the client side.
Default: C:\Windows\System32\mspaint.exe or
%SystemRoot%\System32\rundll32.exe "%ProgramFiles%\Windows Photo Viewer\PhotoViewer.dll", ImageView_Fullscreen %1

PrimaryCamera

The identifier or name of the primary camera, so the application will use this camera to show the face

SecondaryCamera

The identifier or name of the secondary camera, so the application will use this camera to take a pic

RingingTimeout

How many seconds have to wait before redirecting to the central dispatcher.
Default: 60 sec

Registration of workstations

When we install the application on any workstation, it is necessary to register it.
In the central database the administrator manually inserts all possible clients and dispatchers into the Endpoint table.
After installing the program and running that first time, the config.xml does not exist. This is how the application knows that it is first install. If we want to reconfigure, then we will delete it.

The aim is that the administrator creates the necessary record in the Endpoints table, but anybody who installs the program, must not be professional, but if he asks an Id on the phone, then this shall be enough.

Registration process

The program starts right after install, at the first time, asks in a window the followings:

1. What is the Endpoints table Id, which describes this workstation?
2. After the administrator enters the Endpoint ID integer number (relation to the Endpoints table, I will check that and inform the administrator) the program calls the webservice and asks the parameters, and sends which must be sent.
3. When the communication with the webservice was done, the webservice knows about this new workstation, it knows, if it is dispatcher or client, and the application writes all necessary information into the config.xml file
4. When it is done, then the program functionality starts, and ready to work in client or dispatcher mode. When restart, it reads the config.xml, and knows exactly, if it is client or dispatcher, what is the registered Id, and so on