Distributed system for managing dermatology diagnostics

Kim Hansen

Kongens Lyngby 2012
IMM-B.Eng-2010-103

Technical University of Denmark
Institute for informatics and mathematics modeling
Asmussens Alle, building 305, DK-2800 Lyngby, Denmark
E-mail: reception@imm.dtu.dk
Phone +45 4525 3351, Fax +45 4588 2673
www.imm.dtu.dk

Summary

In this project I will create a system which will be able to handle different dermatologic diagnostics. To test one of these, I have borrowed a Matlab model provided by a fellow student named Henrik Mygind[1], which makes feature extraction on an image. The image, which is sent to the model, contains a birthmark. The feature extraction of the image happens with a view to be able to analyze the risk of melanoma.

The assignment is to write about a distributed system to handle a dermatology diagnostic based on submitted images, which on the client side calls one or more web services which is implemented with basis of calculation in Matlab on the server side. This way, the mathematical platform Matlab has to offer in conventional code, is used to its fullest. In this respect, issues in this approach will be looked on, such as maintenance, errors, advantages and disadvantages.

[1] http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6076

Resumé

I den konkrete opgave lave et system, som kan håndtere forskellige dermatologiske synspunkter. For at afprøve en af disse, har jeg lånt en medicinsk billedbehandlings model lavet i Matlab, lånt af Henrik Mygind[2], som laver feature extraction på et billede. Billedet, som bliver sendt til modellen, indeholder et modermærke. Feature extraction af billedet sker med henblik på, at kunne analysere muligheden for kræft i mordermærket.

Opgaven i denne rapport består i at skrive om et distribueret system, der håndterer dermatologidiagnosticering baseret på indsendte billeder, som på klient siden kalder en eller flere webservices, der er implementeret med beregningsgrundlag i Matlab på server siden. Således udnyttes den matematiske platform som Matlab stiller til rådighed i konventionel kode. I den henseende kigges der på problematikker i denne løsningsmodel, som f.eks. vedligeholdelse, sikkerhed, svartider, fejlscenarier, fordele og ulemper.

[2] http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6076

Preface

The report is composed as a completion to the education as Diplom-IT engineer, on the basis of the project I have been working on in cooperation with Pallas Informatik A/S. Together with my adviser at Pallas Informatik, Rune Duschek Saaby and my DTU-adviser Hubert Baumeister, I have decided to make a distributed system for managing dermatology diagnostics based on submitted images.

Acknowledgements

I would like to thank Rune Duschek Saaby and Hubert Baumeister for supervision and guidance throughout the project.

I am very grateful to Henrik Mygind for providing the Matlab model.

I would also like to thank Christian Møller, Morten Toxværd, Thomas Axel and Bo Poulsen for support throughout the project.

Table of Contents

1 Introduction

1.1 Problem statement

The main problem in this project is to create a system, with an infrastructure that can handle different problem domains, such as melanoma detection and eczema.

How can I make it easier for people to check for a possible dermatology diagnosis?

I want to create a system which enforces patient empowerment and thus makes a lot of working procedures easier. For instance, there are a lot of different working procedures in the following:

- Social and health care assistants
  - External workplaces (home care)
  - Multiple patients a day
- Ordinary men and women
  - Worrying about a birthmark on their sons arm
  - Checking a birthmark on their 80 year old mother

A lot of these scenarios could be lightened by having a system which for instance, could determine whether or not you had melanoma or specify the seriousness of eczema.

By giving an individual the opportunity to develop self-help, by automating a standard procedure like going to the doctor to get a birthmark checked for possible melanoma, I will make the life a lot easier for a lot of people, and take some of the pressure of the general practitioners (GP).

The user to this system can be of different occupation, so for instance the local GP could use the system, as well as a private person.

This will hopefully lead to a telemedicine viewpoint and a system which supports this viewpoint.

1.2 Background

To understand the nature behind dermatology and behind the working procedures connected to these, I have looked into two different cases. One I already mentioned is Melanoma.

Melanoma is the fastest growing skin- disease and cancer, and is seen on over 2000 people per year in Denmark. The disease is mostly seen among women, and people born with a lot of birthmarks. The cells in our body which produces melanin (the pigment in our skin), will in the case of melanoma start to split unrestrained. One of the main reasons for getting melanoma is the sun's ultraviolet radiation, and the risk can increase up to 2-3 times when "enjoying" sunbed when you are younger than 25 years old. The reason why melanoma is so dangerous is because of its fast growing rate and tendency for spreading to other parts of the body.

There are different symptoms of melanoma. This can be sudden growth of a birthmark, bleeding or itching or a change of color or multiple colors in one birthmark. Other signs could be irregular edges, asymmetry and a size over 6 millimeters. Since every people have birthmarks, it is only if you see a change in these, that you have to be concerned.

The public health care system recommends anyone with the above symptoms to go see a doctor. This is one of the things, which has led to the making of my system, exactly to lighten this process. The procedure of curing melanoma is by removing the birthmark which is infected. Afterwards the birthmark is thoroughly looked at, and if it is very hostile, the surrounding lymph glands will also be removed, along with the surrounding skin to make sure the tumor is gone.

Another perspective is wound treatment. Eczema or wounds from an ostomy is two very common wound types. The similarity between these two types of wounds is that they both need regular visual treatment, which my program is the perfect solution to.

In this project I have been given a Matlab model, from a fellow student and co-worker[3]. This model makes it possible to classify a birthmark as a melanoma. The model uses feature extraction, together with a neural network and several algorithms that is used for classification. The model has six different values, which together determines the classification of the birthmark.

There are two asymmetric values of the image. The first is the value of the major axis and the second is the value of the minor axis. The third value is the sharpness of the transition from melanoma to the background. Now the mean and the variance of this will be measured. Then there is an edge-to-area ratio used to show how rough the edge is. Finally there is the area size of the birthmark.

[3] Henrik Egeberg Mygind (s062406) http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6076

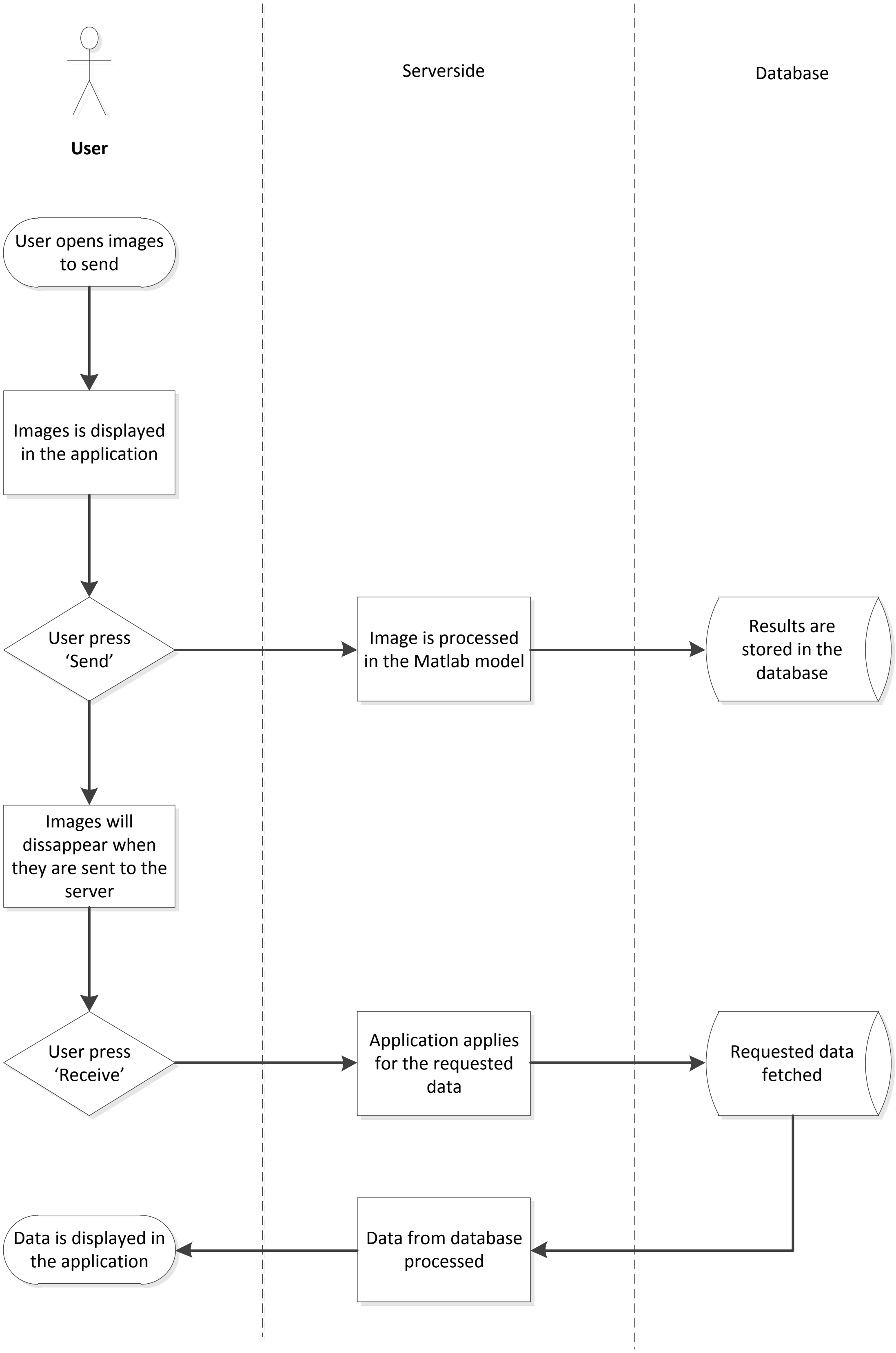
2 Specification requirement

This section contains a description of the requirements of the system together with some use cases, and some background on how a final system could work. Furthermore you will find a description of the architecture requirements.

2.1 Flow Diagram

The flow diagram below shows the typical flow in the program, of what happens when a user submits images to be send.

The user opens the application and finds the folder with the images he wants to send. The selected images will now be displayed in the application. The user can now press ‘send’, to send the images to the server, where they will be processed in the Matlab model. When the images have been successfully sent to the server, the images will disappear in the application window. After the processing is finished the data will be stored in the database. The user can now press ‘receive’ and a request will be sent to the server, which will connect to the database and fetch the data. The data will now be processed and sent back to the user and displayed in the main window.


User
Serverside
Database
User opens images to send
Images is displayed in the application
User press 'Send'
Image is processed in the Matlab model
Results are stored in the database
Images will dissappear when they are sent to the server
User press 'Receive'
Application applies for the requested data
Requested data fetched
Data is displayed in the application
Data from database processed

2.2 Functional & non-functional requirements

The functional requirements of the system are defined in the green section and the non-functional requirements are defined in the blue section. In addition I will provide an example solution to some of the requirements.

Area of concern	**Requirements**	**Comments/Example solutions**
1. User interface (UI)		
1.1 Open	It shall be possible to open 1 or more images. After selecting the folder with the desired image(s), it will be displayed in the application window.	I have chosen a folder solution, because it is intended to a WiFi camera, which uploads the pictures to a specified folder. For further development this solution is cleverer.
1.2 Send	It shall be possible to send 1 or more images to the server. Before sending the images to the server, you will have to enter a unique ID, to identify the data of the images when you want to receive these.	
1.3 Receive	By entering your ID, you will receive the data belonging to that ID's images. The data will be displayed in the application window	
2. System interface		
2.3 Error handling		
	If the connection to the server is lost a reestablishment will be tried, and if it isn't successful, the user will have to send the images again.	

Below you will see two use case descriptions of the main procedures in the program.

Name of use case	Send image
Actors	User
Brief description	The user wants to send an image. He opens the directory, and selects the folder with the image(s) to be sent. After the selection the user presses 'send'.
Basic flow of events	1. The user press 'open'. 2. The user selects a folder containing images. 3. The image will be displayed in the application 4. The user press 'send' 5. The image will be sent to the server.
Alternate flow of events	1. 1-4 is repeated. 2. The server returns an error, that the image cannot be received. 3. The user will receive a note to try again in a few minutes.
Entry conditions	The user has an image to send.
Exit conditions	The image will be successfully sent.

Name of use case	Receive image
Actors	User
Brief description	The user wants to receive an image. He will now have to enter the email connected to the image. The user received the data connected to the image.
Flow of events	1. The user press 'receive'. 2. The user enters email. 3. The data connected to the images email will be displayed in the main window.
Alternate flow of events	1. 1+2 is repeated. 2. The email is incorrect, so the user will be prompted to try again.
Entry conditions	The user wants to receive an image.
Exit conditions	Data from the image is successfully received.

2.3 Concrete architecture requirements

Figure 2.3.1 shows the concrete architecture of the system. I have connected two C#-applications, which communicate through a WCF-service hosted on an Internet Information Service (IIS). On the IIS, the C#-application connects to the Matlab model through Matlab Compiler Runtime (MCR)[4]. Furthermore the C#-application has a connection to an MSSQL database, which is on the server.

There is no special reason why I have chosen this particular architecture; this is a proof of concept. Implementation of the final system will take time and money, and acceptance from several health care divisions and the Danish Melanoma Group (DMG)[5].

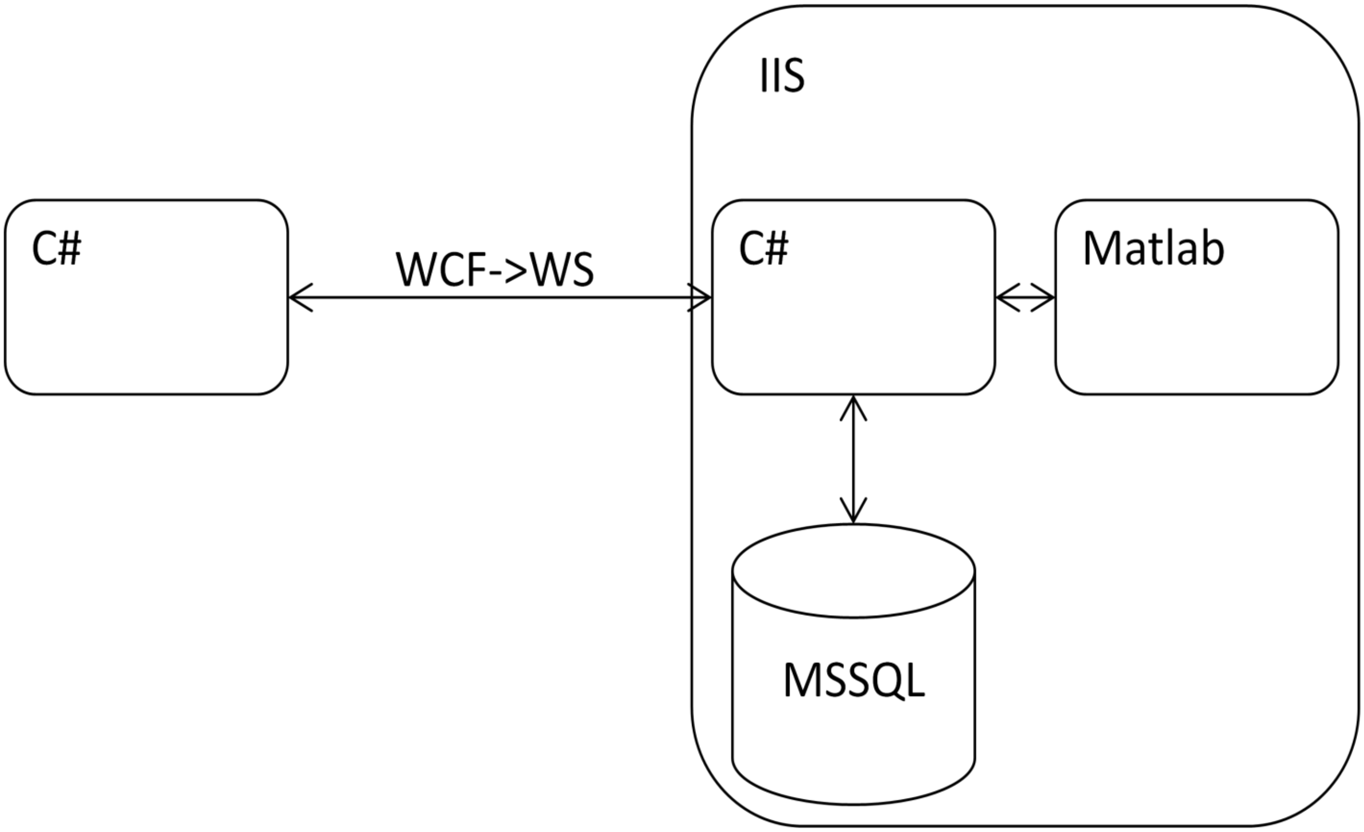


Figure 2.3.1

[4] See section 3.1.2.1 for more info on the MCR.

[5] See section 4.1 for more info about DMG.

3 Technology

The development of the system is in C# .NET (Client/server side), with a WCF service, and Matlab (Server side). The server is a windows server 2008, with a role as a Web Server. The database is on this server, and it is an MSSQL database. In this section you will find a description of the different technologies I have used.

3.1 Matlab

To make the Matlab model work with .NET, I have used two additional deployment programs from Mathworks; Matlab Compiler and Matlab NE Builder. With these it is possible to compile the model to a usable .NET component (.dll).

3.1.1 Matlab Builder NE

Mathworks provides the possibility to create .NET and COM components, from your own Matlab program, by using the Matlab Builder NE. These components can now be used in your own .NET application, independently from Matlab. By the use of the Matlab Compiler, the Matlab Builder NE encrypts the Matlab program, and generates a .NET wrapper around it. Now I can use it like a native .NET component in my C# application.

3.1.2 Matlab Compiler

The Matlab Compiler makes it possible to share a Matlab application. This makes it possible to run your Matlab application outside the Matlab environment, together with you .NET application. It is also possible to make shared libraries for other platforms than .NET, such as Java or even excel add-ins. To run your application you will need a runtime engine called the Matlab Compiler Runtime (MCR), which is provided with the Matlab Compiler.

3.1.2.1 Matlab Compiler Runtime

MCR is a runtime engine, which helps you execute your Matlab application outside the Matlab environment. It is very important to add the MCR.exe to your Matlab program when you compile it to a .NET component. Otherwise you won't obtain the correct version, and your Matlab application won't work in other environments. The MCR has a lot of dependencies, which plays a big role when you want to develop a distributed system; because it has to be installed on every platform where you access your Matlab program. At first when I used the MCR, I didn't realize this, so it took a lot of time to get all the different components to work together.

A very important component MCR has to offer is the MWArray, which contains all the data types in Matlab. These data types can be converted to native C# data types and I will explain how this works in section 5.

MCR is a fairly new feature, so it still have some very annoying bugs. As I mentioned earlier, I had to make sure that the MCR corresponded to the right Matlab version and the right MWArray.dll version.

A strange thing suddenly happened when I was testing my program; it was looking for another .dll. This wasn't mentioned anywhere on Mathworks homepage, so this took a lot of time to figure out. It turned out to be a .dll needed in the project, by the MCR and Matlab, which could be found in the MCR folder[6].

3.2 WCF

Windows Communication Foundation (WCF) is a communication service integrated in the .NET framework, which allows service oriented applications to communicate across the web. The reason I am using WCF and not another form of web service, e.g. ASMX service, is because of all the opportunities and complexity in WCF. For instance WCF is capable of sending complex types and it is the latest communication service. The way you communicate with a WCF service, is through an interface which has an operation contract. This means that any class that implements the interface can be a service. Another class which uses the same interface could also be implemented. This offers is a loose coupling between server and client, which makes it easy to change. Through various configurations, it is possible the expose the service as wanted. To learn all the configurations is a very complex process, and it takes time, thus I have not had the time to study WCF as much as I would have liked.

If you look at performance, WCF has improved performance over ASMX, this is without using any security. Performance is needed in my program, due to the possibility of large image files. WCF uses the basic WCF binding, BasicHttpBinding, and ASMX uses the ASP.NET standard.

[6] The version control associations can now be found here: http://matlab.wikia.com/wiki/FAQ

4 Design

This section contains information about the design of my system. You will find the most common architecture of the system and design of the database.

4.1 Architectures

Figure 4.1.1 shows the architecture for a normal user, who takes a photo with a digital camera or a smart phone, which either has a WiFi card or the user can upload the photo manually.

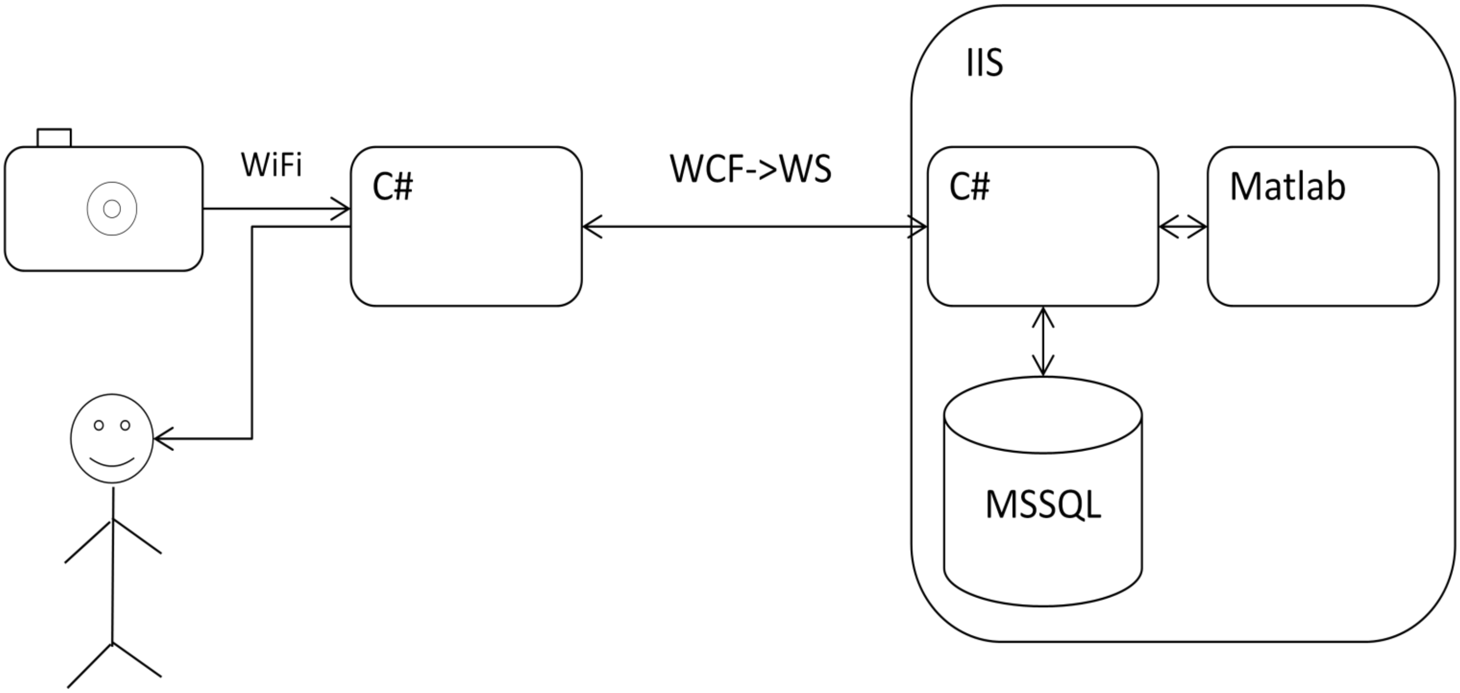


Figure 4.1.1

The idea here is to make it easier for the user to upload images. I have for no particular reason chosen C#, since this is a proof of concept. C# was a natural choice for me and it also is easier to debug, and thus less time consuming, than a straight browser solution. It is also a lot easier to configure a target for the Wi-Fi card. When you take a picture with a camera with a Wi-Fi card, the picture will be transferred immediately to a given folder. In the final solution, I will have a program running, and when one or more pictures are received in the folder, the program will load the images immedietly and wait for the user to accept the loaded images, and then they will be sent to the Matlab model to be processed.

As you can see below, I have provided a Dataflow diagram, which shows how the flow of the final system that is aimed for could be (see figure 4.1.2).

In the final version there are several components, which I am not implementing in my version, but I will explain the process below.

The user shall be able to fetch a template from the web, to use when taking an image of a birthmark. This template will make a more complex Matlab model able to analyze an image with a birthmark more thoroughly, and thus will make it even more accurate than in the model I am using. The template contains different colors to compare with the picture taken, a grid to make it easier to measure the size of the birthmark and a QR-code, which contains answers from a questionnaire filled out by the user before the template is printed[7]. Furthermore, there will be several notes returning to the user, with messages about the ongoing process of the image. Dependent on the result of the diagnosing the user will receive a final answer to the investigation.

If no melanoma is detected, the user will receive a note explaining there is no risk in the processed birthmark, but to be certain the process must be repeated in approximately four weeks.

Whether the classification detects suspicion of melanoma or not, the data will be transferred to and stored in an Image Archiving and Communication System (PACS). In Denmark, a group called "Dansk Melanom Gruppe" (Danish Melanoma Group – DMG) has provided several guidelines for those with positive melanoma[8]. This group have also formed several teams called "Multidisciplinært team" (Multi-Disciplinary Team" – MDT), which consists of several healthcare professionals, around the country. These teams handle the patient, after a melanoma turns positive. So if Melanoma is suspected, MDT will receive a note with a message about new potential melanoma, and then the patient will enter into a procedure called "Pakkeforløb for modermærkekræft", which is a series of steps for all with melanoma to follow[9].

The human followup will prevent a false positive, that will scare the user, or a positive false, which will leave the user with a false relief.

The system will also prevent a lot of unnecessary visits to the doctor, because he will only be notified if there is a possible uncertainty. In an economic point of view, this would save a lot of money to the state.

[7] In appendix A I have provided an example of the template, which I have contributed to create in April 2011.

[8] You will find the guidelines on www.melanoma.dk

[9] See http://www.sst.dk/~/media/Planlaegning%20og%20kvalitet/Kraeftbehandling/Pakkeforloebsbeskrivelser/Pakkeforloeb%20for%20modermaerkekraeft.ashx

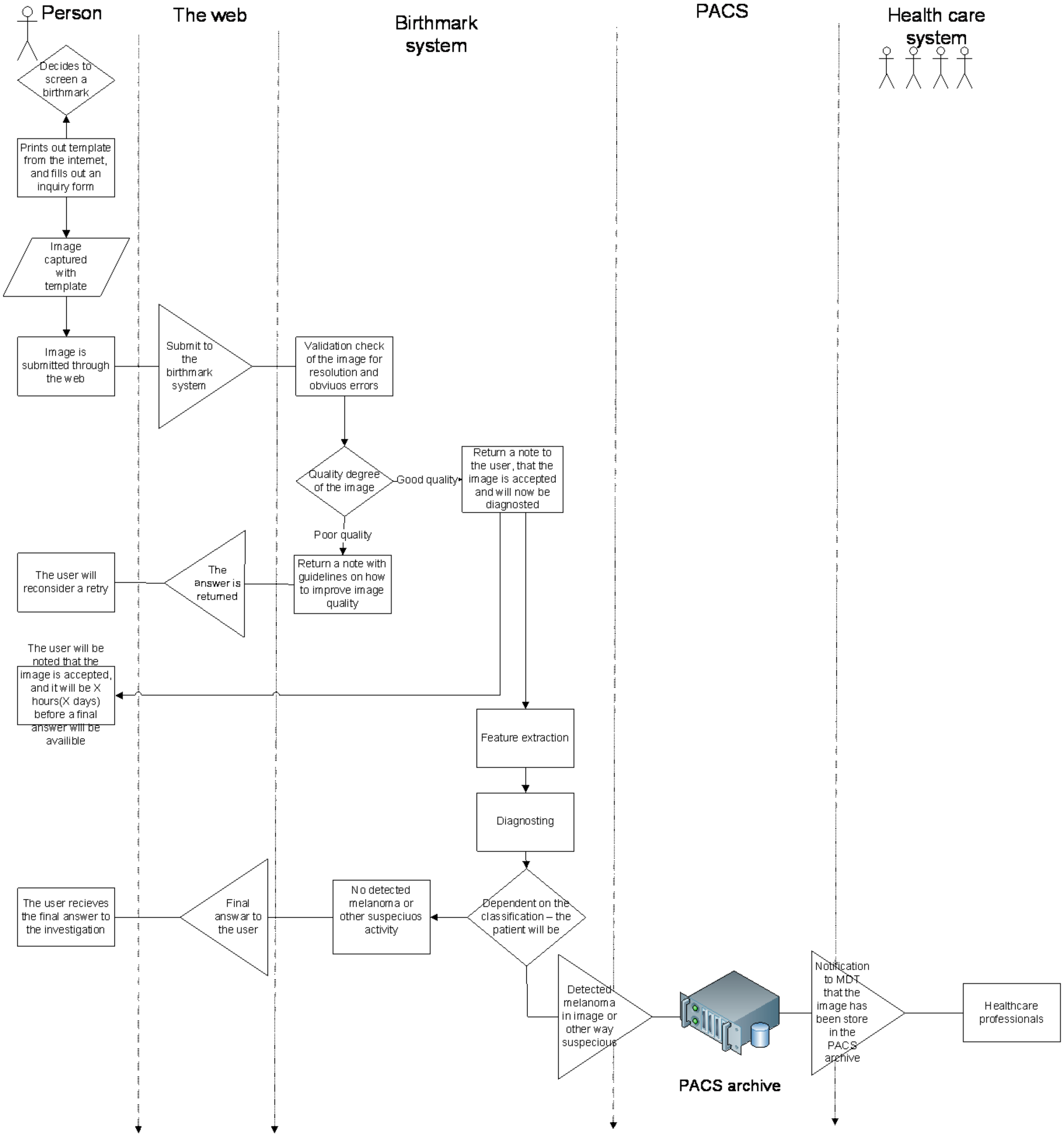
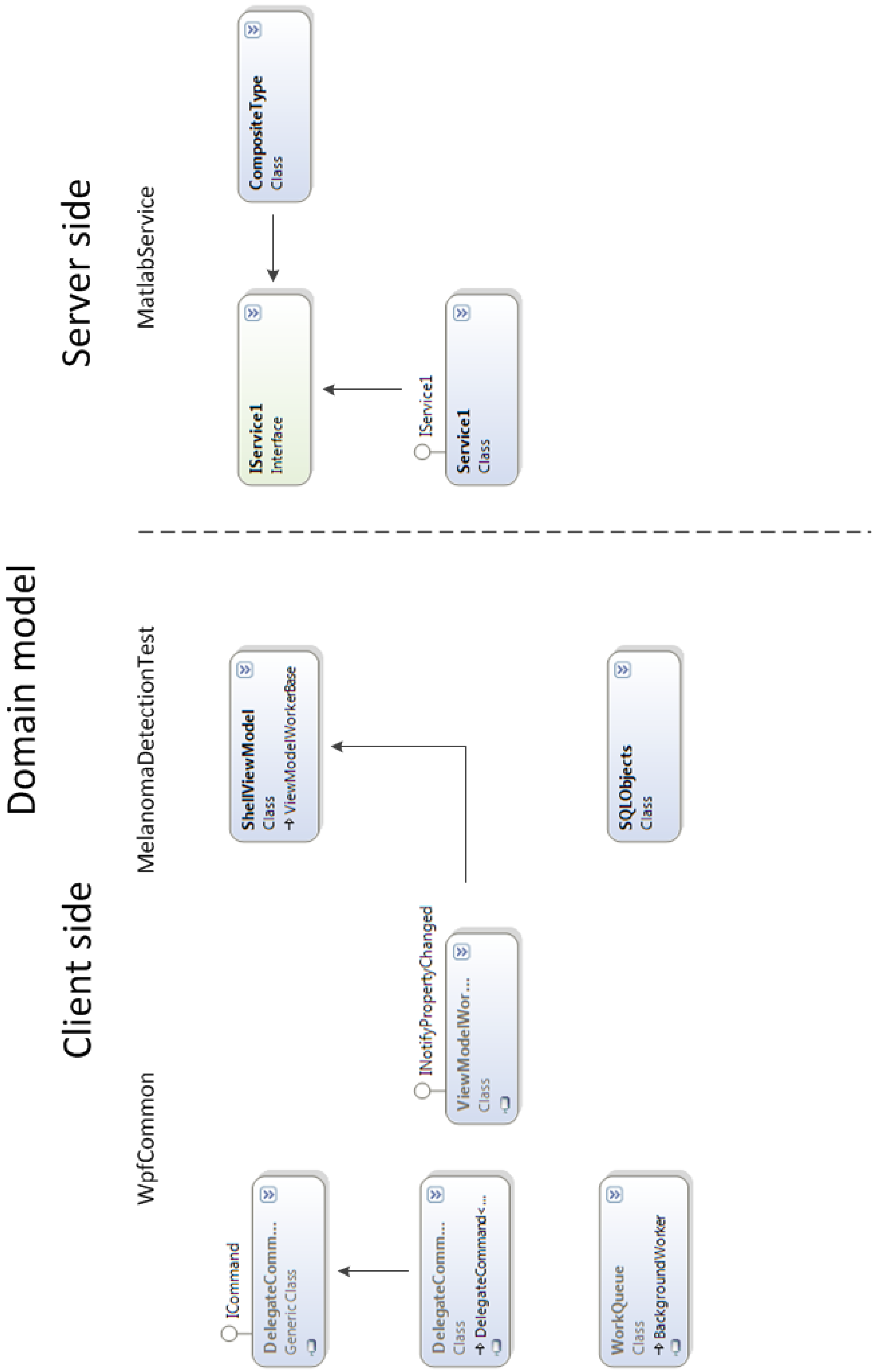

Person
The web
Birthmark system
PACS
Health care system
Decides to screen a birthmark
Prints out template from the internet, and fills out an inquiry form
Image captured with template
Image is submitted through the web
Submit to the birthmark system
Validation check of the image for resolution and obviuos errors
Quality degree of the image
Good quality
Return a note to the user, that the image is accepted and will now be diagnosted
Poor quality
Return a note with guidelines on how to improve image quality
The answer is returned
The user will reconsider a retry
The user will be noted that the image is accepted, and it will be X hours(X days) before a final answer will be availible
Feature extraction
Diagnosting
Dependent on the classification – the patient will be
No detected melanoma or other suspeciuos activity
Final answar to the user
The user recieves the final answer to the investigation
Detected melanoma in image or other way suspecious
PACS archive
Notification to MDT that the image has been store in the PACS archive
Healthcare professionals

Figure 4.1.2

10



[10] A class diagram with attributes and methods can be found in appendix B

4.2 MSSQL

Below you see my SQL class. It is now very complex, but at the moment it is all I need. I will briefly explain the attributes so the names will make more sense. 'Asym1' and 'Asym2' is the asymmetric values of the image, 'Asym1' is the value of the major axis and 'Asym2' is the value of the minor axis. 'EA' is the sharpness of the transition from melanoma to the background. 'MeanEA' is the mean of EA. 'VarEA' is the variance of EA. 'edge2area' is a ratio used to show how rough the edge is. 'areasize' is the area size of the birthmark. For a more detailed description, go to section 1.3. Furthermore there is an Email, which is provided by the user, and a unique ID which is auto incremented in the database.

ImageData	
PK	**ID : Int**
	Email : Varchar Asym1 : Varchar Asym2 : Varchar MeanEA : Varchar VarEA : Varchar edge2area : Varchar areasize : Varchar

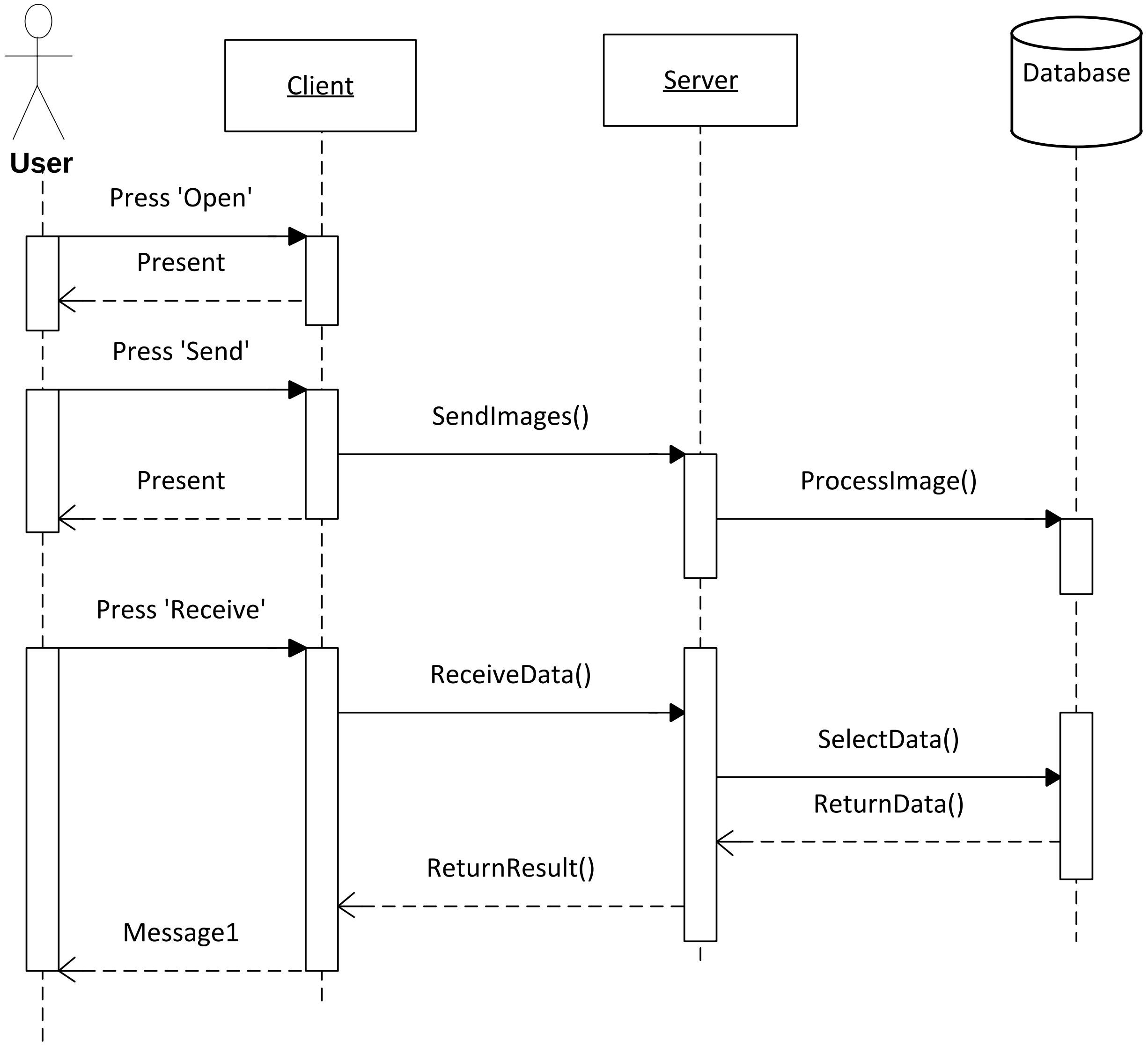
5 Implementation

In this section you will find information about the implementation, such as the systems functionality and some relevant series of code.

5.1 Functionality

Below you see an overview of how the different parts of the system interact. Further on you will find a more specific overview of the client and the server part.

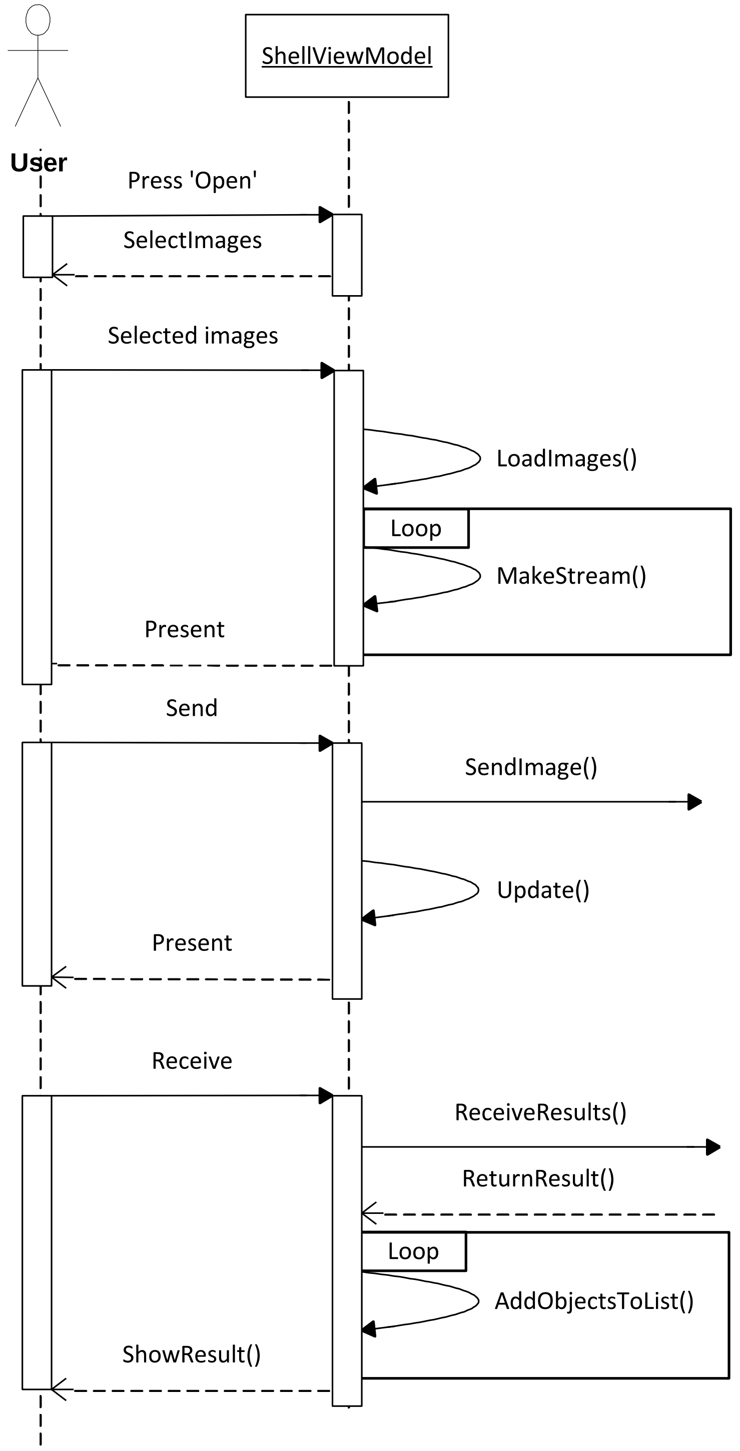
System overview



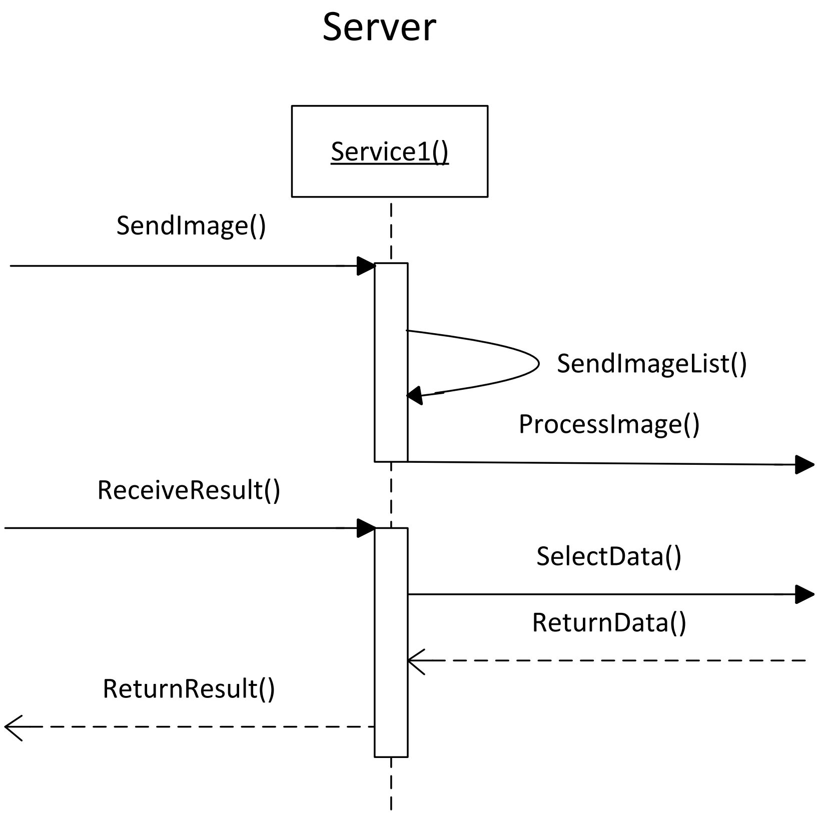
In the initial window[11] the user has 3 different options. When the user press ‘Open’, a folder browser dialog appears, where the user is to choose the folder containing the images he wants to send. The images will now be displayed in the window. Now the user can choose to send the picture to the server for processing. When pressing ‘Send’ the user has to give an email, which will be connected to the data extracted from the images. The images will now be sent, and the window will be updated, showing no images. After the images have been processed, the data will be saved in the database, and the user can receive them by pressing ‘Receive’. An email is needed to retrieve the data and when such is given the data will be displayed in the window.

[11] See section 6 for figures of the different windows

Client


ShellViewModel
User
Press 'Open'
SelectImages
Selected images
LoadImages()
Loop
MakeStream()
Present
Send
SendImage()
Update()
Present
Receive
ReceiveResults()
ReturnResult()
Loop
AddObjectsToList()
ShowResult()

On the server I have multiple threads running for up to 3 pictures at a time. I use "Task Factory" for this, because it makes it very easy to create new threads, compared to the use of the "Threading" class. The method *SendImageList()* creates a new thread with the task to execute *ProcessImage()*. I have added a TaskScheduler, which handles number of threads running, so I don't get an overflow of threads. By using the methoed ".ContinueWith()", I can add a new action to be executed, when the initial task is finished. I have used this to create a recursive method, so all the images will be processed.



5.2 Matlab

The system is created around the given Matlab model. To integrate the Matlab functionality, I have generated a .NET component[12] from Matlab and added it as a reference to my service. Together with the given MWArray component[13], I can now convert the data types from Matlab, to native C# data types. Below is an example from the code of how this is done.

```
//Accessing the Matlab class and receives a Matlab data type. Here an MWArray.
var values = m.ProcessImage(firstFilePath);
```

[12] See section 3.1 about Matlab.

[13] From the MCR. See section 3.1.2.1.

```
//The MWArray contains a data type, MWStructArray, which is like a dictionary in C#. I now extract the values.
var asymmetri = ((MWStructArray)values)["asymmetry"];
var meanEA = ((MWStructArray)values)["meanEA"];
var varEA = ((MWStructArray)values)["varEA"];
var edge2area = ((MWStructArray)values)["edge2area"];
var areaSize = ((MWStructArray)values)["areaSize"];
```

Here I extract the values in the MWArray by casting it to the right data type, in this case MWStructArray. Now I have the values processed in the Matlab program and I know store them in the database.

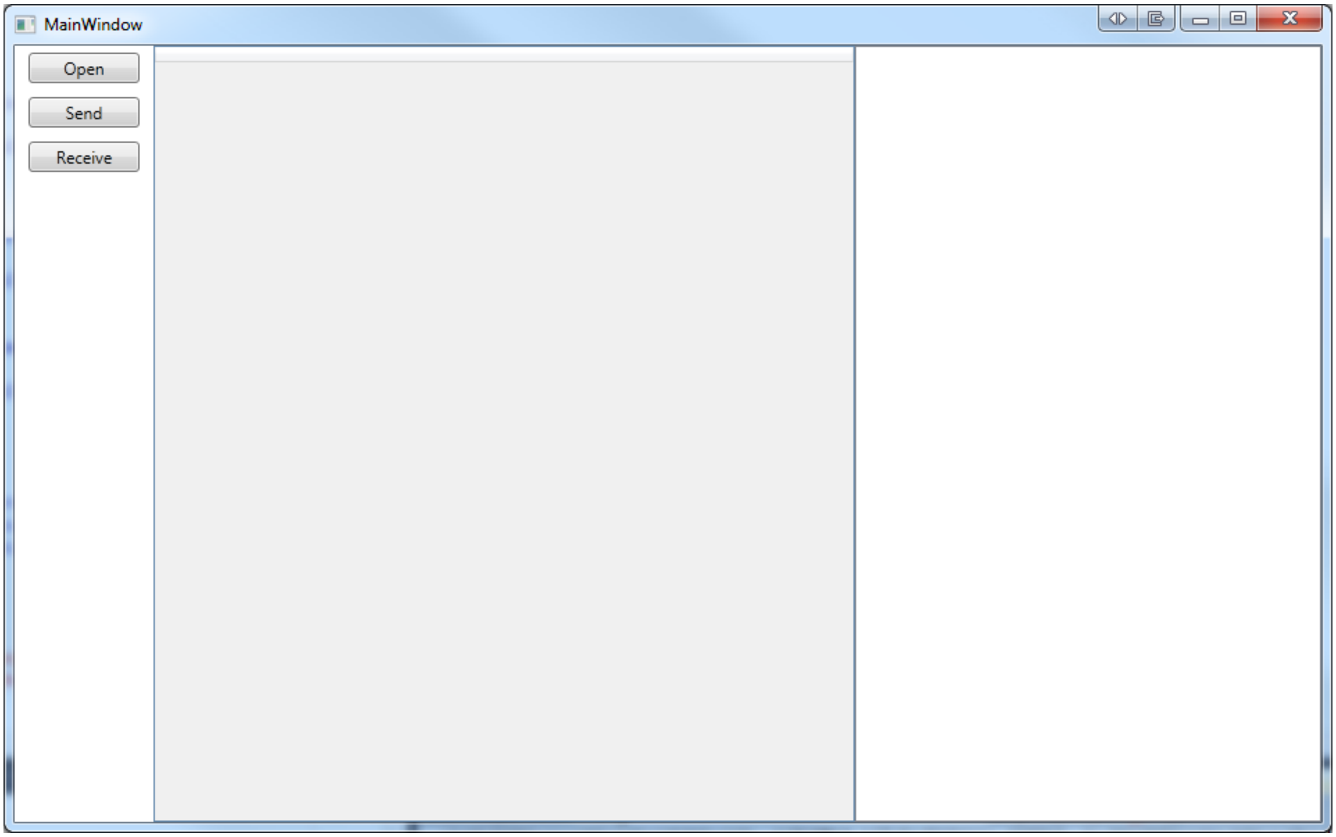
```
var sqlCmd = new SqlCommand(@"INSERT INTO TestTable (Email, Asym1, Asym2, meanEA, varEA,
edge2area, areasize) VALUES (@Email, @Asym1, @Asym2, @meanEA, @varEA, @edge2area, @areasize)",
sql);
```

The data from the image will now be stored in the database and will be available when the user is ready to receive it.

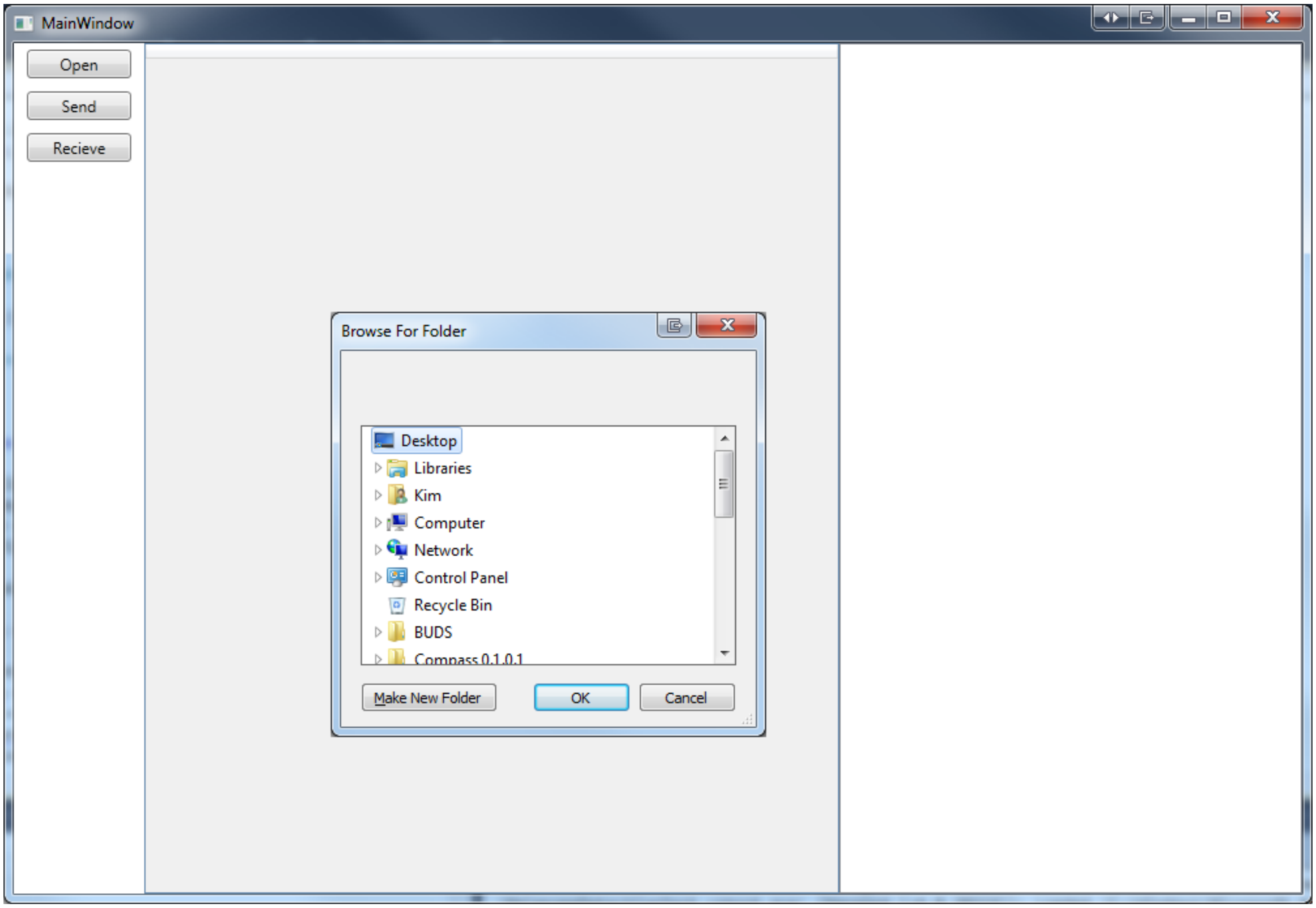
6 Test

I have tested the application and below you will see the steps I have followed, and images belonging to this.

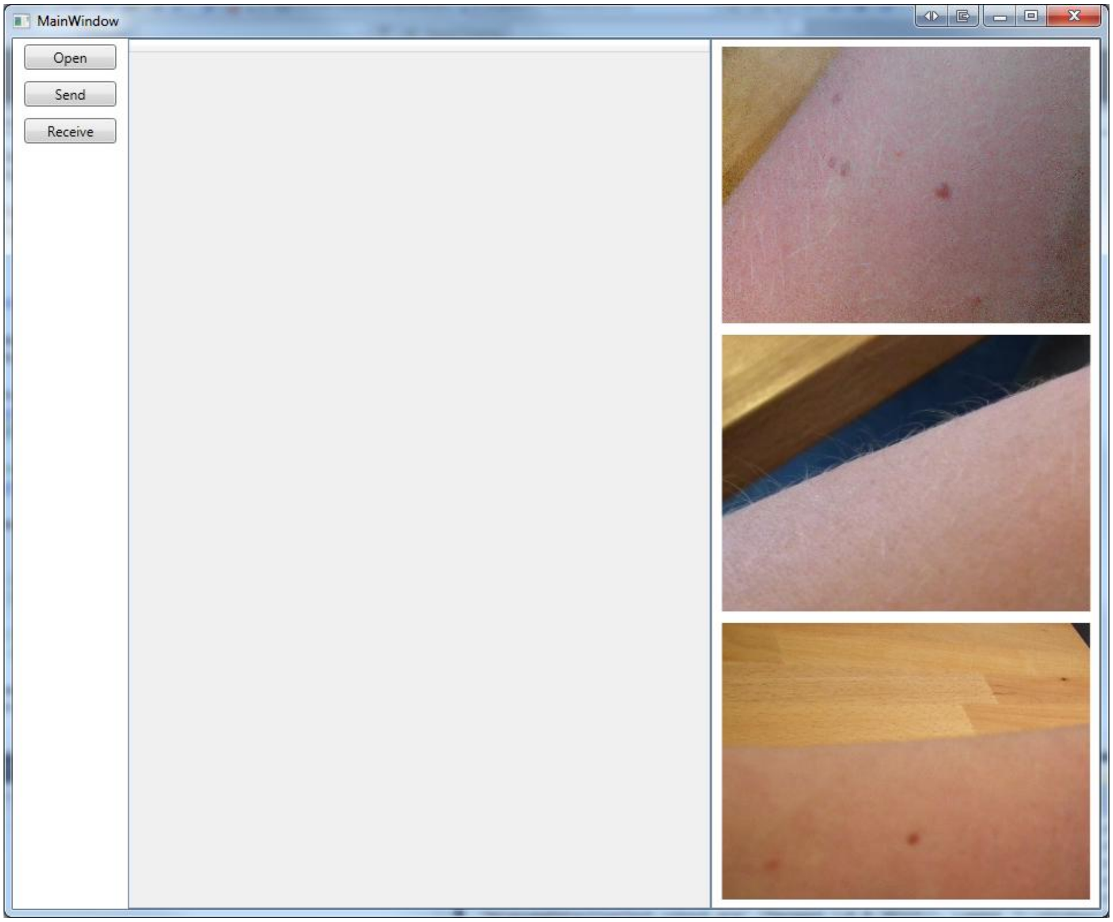
When I run the program, the main window will dispay:



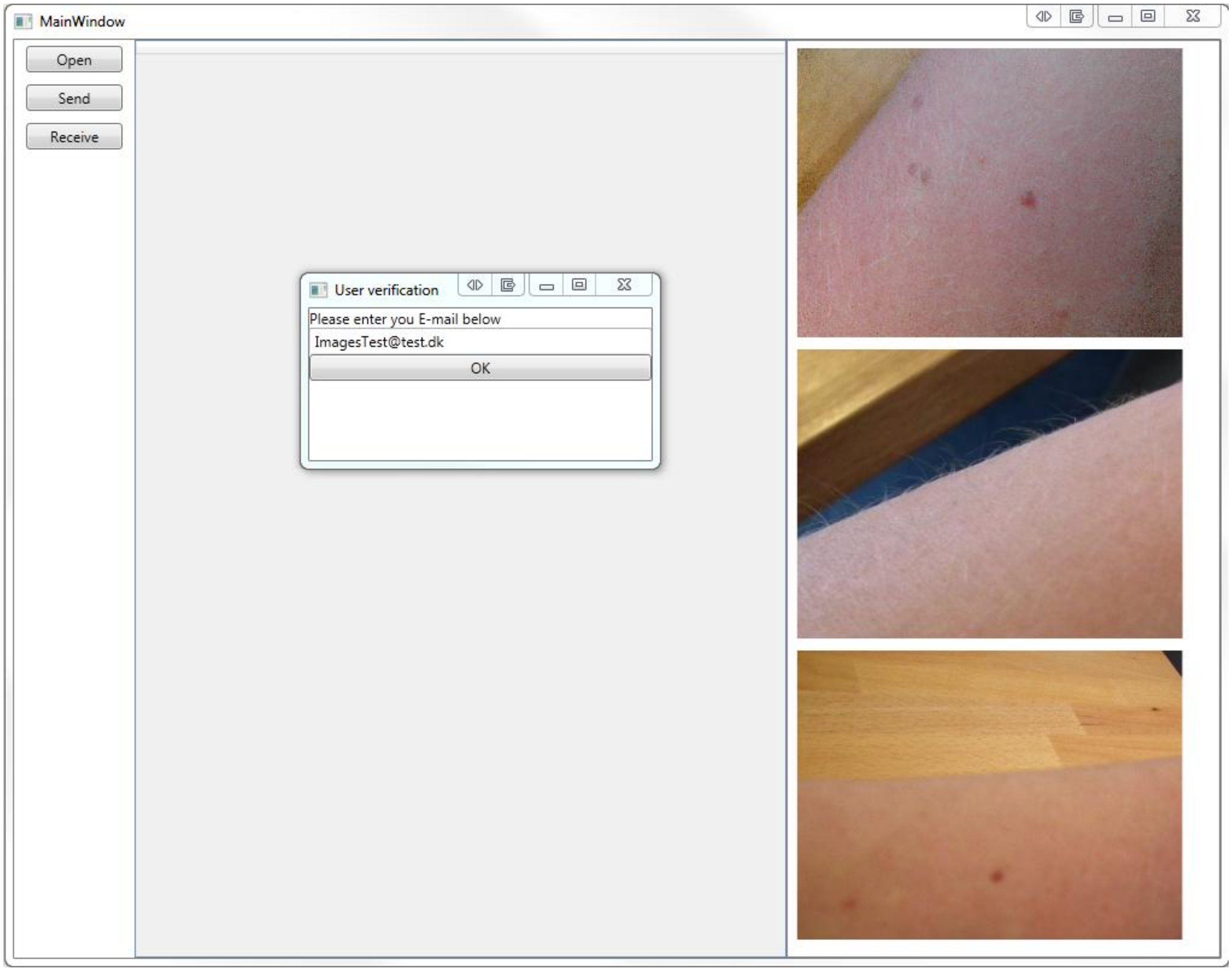
Here you have 3 options: “Open”, “Send” and “Receive”. Now I am pressing “Open”, to select where I want to load my images from. I now get a folder browser, so I can choose the right path.



After navigating to the right path, I press “OK”, and the images will be displayed in the main windows, in the right sidebar.



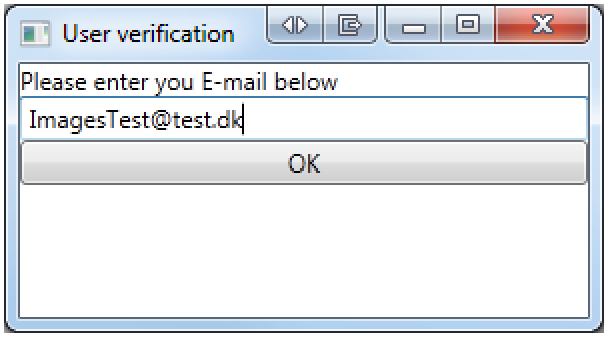
Now I press “Send” and the following dialog displays where I am asked to enter my E-mail.



Please enter you E-mail below

ImagesTest@test.dk

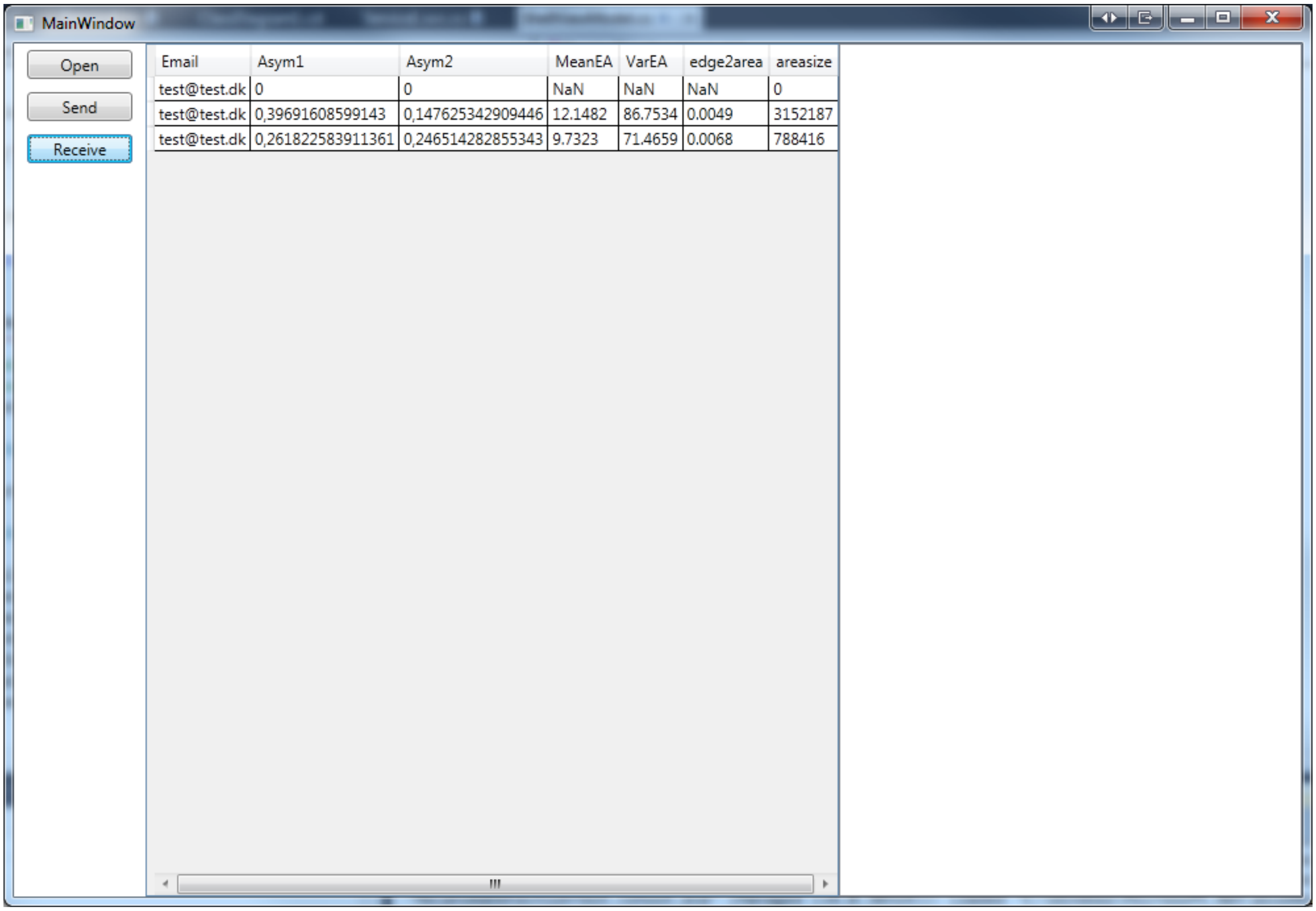
Now the images disappear, and I am back at the initial main window. I can now choose the last option, "Receive" to receive the data from the images I just sent. When I press "Receive" I get another user verification dialog, where I shall enter the email linked to the images I want to receive data on.



Please enter you E-mail below

ImagesTest@test.dk

After pressing “OK”, I will see the data displayed in a data grid in the main window.



Email	Asym1	Asym2	MeanEA	VarEA	edge2area	areasize
test@test.dk	0	0	NaN	NaN	NaN	0
test@test.dk	0,39691608599143	0,147625342909446	12.1482	86.7534	0.0049	3152187
test@test.dk	0,261822583911361	0,246514282855343	9.7323	71.4659	0.0068	788416

7 Further development

For further development of this system, a clever choice would be to net enable the application by making a cloud solution. This way anyone with an internet connection could access the application anywhere. Furthermore I would create an App for several platforms, such as the iOS and Android. Windows Phones can run C# code, so this would only take a little optimization.

I would also work on the database and save the images here too and not only the data. This way you can always go back and look on the images in case of an error or if they make a new discovery in science about melanoma, some pictures may need to be rechecked. Furthermore they could be used in a study of melanoma, birthmarks or other dermatology point of views.

8 Conclusion

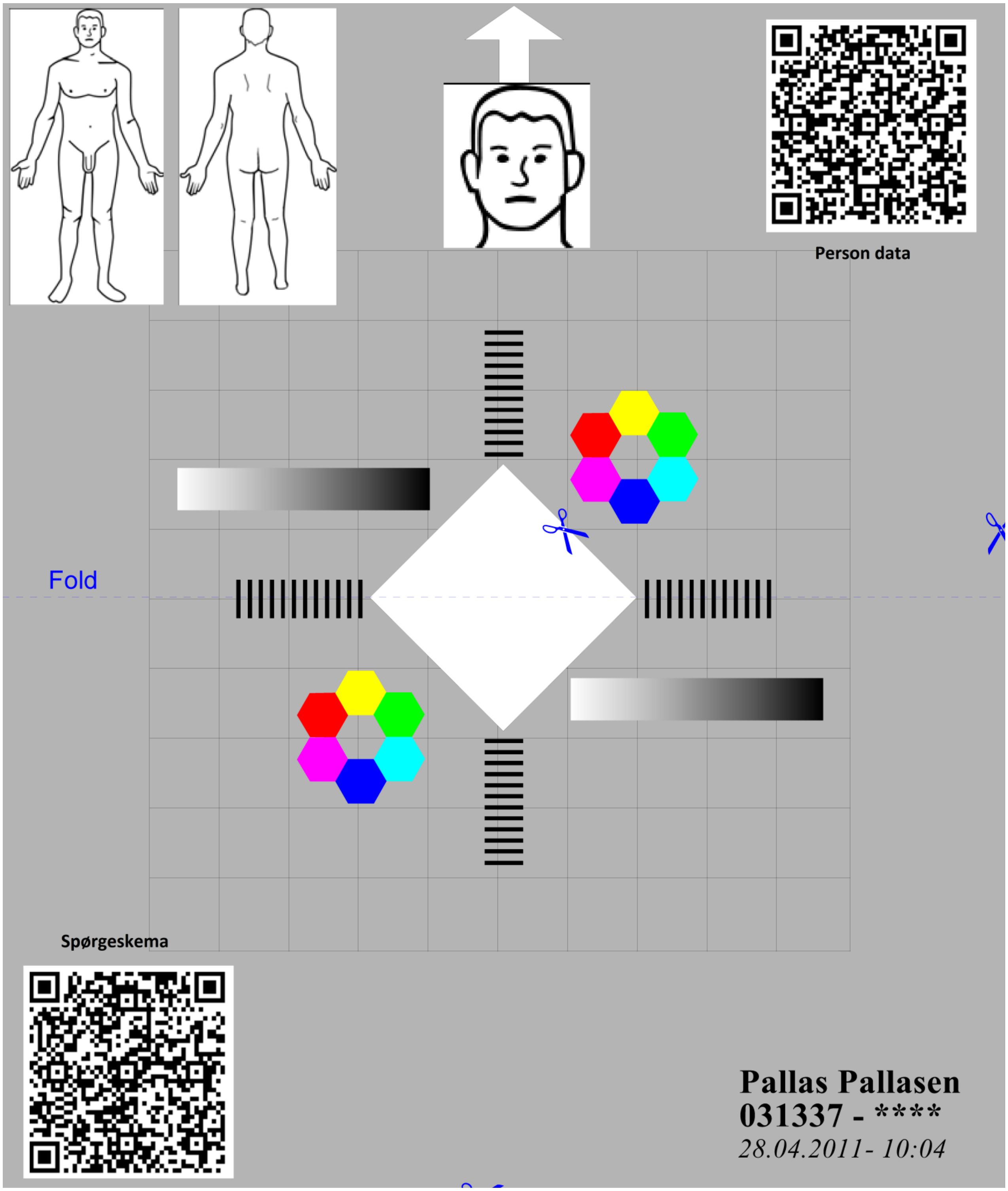
It has been a very exciting and a very learning rich experience to make this program. My final goal for this program is a bit further ahead, but due to some (big) bumps on the road with the MCR and another personal incident, I know I have managed to make the program as well functioning as possible. There is still a lot of work that needs to be done, to make this program worth implementing in the health care system, but I definitely think I have showed the way.

9 References

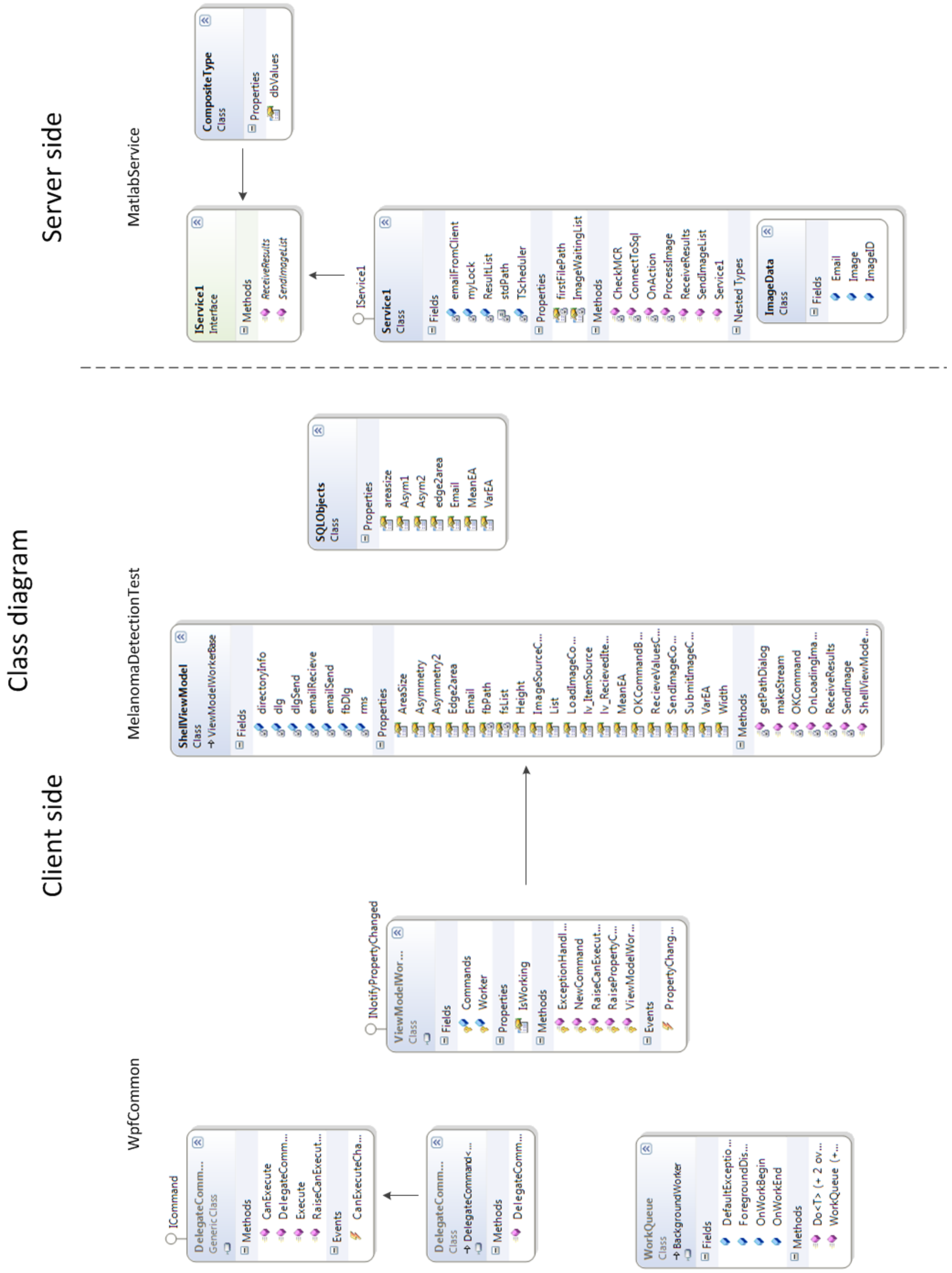
1. http://www.mathworks.se/products/
2. http://msdn.microsoft.com
3. http://msdn.microsoft.com/en-us/library/bb310550.aspx
4. http://www.netdoktor.dk/sygdomme/fakta/hudkraeftmaligntmelanom.htm
5. http://www.cancer.dk/Hjaelp+viden/kraeftformer/kraeftsygdomme/modermaerke/
6. http://www.sundhedsguiden.dk/da/temaer/alle-temaer/hudkraeft/malignt-melanom-modermaerkekraeft/
7. http://www.bispebjerghospital.dk/menu/Afdelinger/Kliniske+afdelinger/Dermato-Venerologisk+Afdeling+og+Videncenter+for+Saarheling/Videncenter+for+Saarheling/For+sundhedsfaglige/Generel+saarbehandling/
8. http://www.sundhedsmagasinet.dk/aid/1058/Telemedicin-en-oversigt.htm
9. http://matlab.wikia.com/wiki/FAQ

10 Appendix

10.1 Appendix A: Template



10.2 Appendix B: Class diagram


Class diagram
Server side
MatlabService
CompositeType
Class
Properties
dbValues
IService1
Interface
Methods
ReceiveResults
SendImageList
IService1
Service1
Class
Fields
emailFromClient
myLock
ResultList
stdPath
TScheduler
Properties
firstFilePath
ImageWaitingList
Methods
CheckMCR
ConnectToSql
OnAction
ProcessImage
ReceiveResults
SendImageList
Service1
Nested Types
ImageData
Class
Fields
Email
Image
ImageID
Client side
MelanomaDetectionTest
SQLObjects
Class
Properties
areasize
Asym1
Asym2
edge2area
Email
MeanEA
VarEA
ShellViewModel
Class
→ ViewModelWorkerBase
Fields
directoryInfo
dlg
dlgSend
emailRecieve
emailSend
fbDlg
rms
Properties
AreaSize
Asymmetry
Asymmetry2
Edge2area
Email
fbPath
fsList
Height
ImageSourceC...
List
LoadImageCo...
lv_ItemSource
lv_Recievedite...
MeanEA
OKCommandB...
RecieveValuesC...
SendImageCo...
SubmitImageC...
VarEA
Width
Methods
getPathDialog
makeStream
OKCommand
OnLoadingIma...
ReceiveResults
SendImage
ShellViewMode...
INotifyPropertyChanged
ViewModelWor...
Class
Fields
Commands
Worker
Properties
IsWorking
Methods
ExceptionHandl...
NewCommand
RaiseCanExecut...
RaisePropertyC...
ViewModelWor...
Events
PropertyChang...
WpfCommon
ICommand
DelegateComm...
Generic Class
Methods
CanExecute
DelegateComm...
Execute
RaiseCanExecut...
Events
CanExecuteCha...
DelegateComm...
Class
→ DelegateCommand<...
Methods
DelegateComm...
WorkQueue
Class
→ BackgroundWorker
Fields
DefaultExceptio...
ForegroundDis...
OnWorkBegin
OnWorkEnd
Methods
Do<T> (+ 2 ov...
WorkQueue (+...