A Windows Application for Secure Message Communication using Image Steganography

DT265
Higher Diploma in Computing

David Flynn B.Eng (Electronics)

Project Supervisor
Catherine Mulwa

School of Computing
Dublin Institute of Technology

<Date: 11th February 2012>

Abstract

My initial aims with this project was to develop an application which would have a user interface and a backend Database with a tier of .Net Classes in between which would handle all the processing plus both enrich and challenge me in writing such classes. I wanted also to learn how to research, develop an application utilising a user interface and a back end Database. The next challenge was to choose an area of research or subject which would present a sufficient challenge, enhance my knowledge, boost my job skills and also peak my interest plus hopefully that of others whom would use this application. I choose steganography as I felt this would tick all boxes; I decided to focus on image steganography, i.e. embedding of messages within images.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

____

David Flynn

<Date>

Acknowledgements

I would like to take this opportunity to thank the staff and students of the Higher Diploma in Computing in DIT for their values assistance and support in this endeavour. I would particularly like to thank my Project Supervisor Catherine Mulwa for her excellent guidance and support in this project. I would also like to thank Professor Jonathan Blackledge, Stokes Professor of Digital Signal Processing, School of electrical Engineering, DIT whose lectures on Steganography and Cryptography which I attended in 2011 particularly influential in my choice of subject for this Project and Application.

Table of Contents

1. Introduction

This Project is an Application which allows users to embed a message inside image along with an authentication key. The same application will also allow the recipient to retrieve the message from the same image provided they have the key. The application will be a downloadable executable file from this website: http:// members.upc.ie/david.flynn8. Users will half to register initially and login each time to use the application. For this purpose the application connects to a Database hosted online at stem.arvixe.com. When a user embeds a message within a image they can choose to arrange transport of the image to the recipient via email, usb etc. or use the Database to store the image for the End User. Plus the Database will also facilitate storage of the authentication key if required. This is a Windows Application(except xp at the moment). I choose to develop a windows application as I was most familiar with using .Net technologies. The target group of users are those whom need to send secure or hidden messages.

1.1Dissertation Question statement

"Can we effectively send embedded messages securely using image steganography?"

1.2 Aims and Objectives

- Investigate existing algorithms developed using c# that implement secure communication through image processing.
- Specify, design and develop an algorithm that allows users to embed messages within an image and then to facilitate retrieval of such a message.
- To test the developed algorithm by developing a simple user interface whereby users can actually use plus interact with the application.
- Implement a backend or database to the application using Mssql or similar technology to provide a list of instances in which the end application was used detailing the message sent, image used plus maybe the detail of the both the front and end user.

1.3 Methodology

The Methodology that I will be using for the development of this application is RUP or *Rational* ***U****nified* ***P****rocess*, a software development methodology. Based on UML, RUP organizes the development of software into four phases, each consisting of one or more executable iterations of the software at that stage of development.

- **Inception --** In this stage, the projects business case is stated and the team decides if the project is worth doing or if it is even possible. It is important to the process to first formulate the scope of the project and also determine what resources will be needed.
- **Elaboration --** In this stage, the developers take a closer look at the project to determine its architecture foundation and to evaluate the architecture in relation to the project. This stage is important to the RUP because it is here that developers analyse

the risks associated with changing the scope of the project or adding new technologies along the way.

- **Construction --** In this stage, the development of the project is completed. The application design is finished and the source code is written. It is in this stage that the software is tested to determine if the project has met its goal laid out in the inception phase.
- **Transition --** In this stage, any fine-tuning is performed. Any final adjustments can be based on user feedback, usability or installation issues.

RUP is similar in concept to Extreme Programming in that only what is useful and required is produced and the development plan is updated throughout the process. Both methods seek to develop a system of best practices in software development.

1.4 Dissertation Structure

The Dissertation is broken down into a number of different sections. **Chapter Two** is a literature review which details the following: Background; Investigation of existing systems; Challenges of developing this application; Benefits of this application. **Chapter Three** covers Architecture: Architectural Design, Technical Design and Process flow Diagrams. **Chapter Four** covers Implementation which is the different phases of the RUP process. **Chapter Fiv**e presents the results on testing and evaluation. In **Chapter Six** the author reflects on the overall process of the project. Finally **Chapter Seven** presents a conclusion and recommends future work.

2 Literature Review

2.1 Background

Secure communications has always been a priority for individuals, groups plus various public, private and state players/nations for thousands of years. History books are generally full of wars but generally omit the intelligence/information war both proceeding and after the conflicts in [1] question.

Cryptography has been documented since classical times. In ancient Babylon couriers would write a message on their heads and then the recipient would have the courier's head shaved in order to retrieve message. Caesar famously used his own cypher to encode his messages on the battlefield both to other units and indeed to Rome itself.

With the advent of modern technology, communications has moved from physical media such as paper etc. to digital communications. Information is usually passed over the air, copper or fiber using various electrical or optical signalling systems. At a higher or computer/user-interface level people and even systems use email or other systems to relay messages or information. This information if considered critical is usually encrypted at some level.

Encryption or Cryptography is the process of concealing or rearranging a message so it can be only interpreted by the intended end user or system. RSA and AES are two such algorithms employed in Cryptography software which will encode a message using a specific code or key which can only be decoded using that software and key.

Cryptography has one weakness in that when a message is encrypted it stands out in that someone will take notice of a message encrypted in the first place and dedicate resources to decipher that message. Steganography is another approach whereby the message is not encrypted but embedded or hidden within another plain message such as a photograph. This has the added advantage in that the fact that a critical message is been sent in the first plus the hidden message is slightly encrypted in that the message can only retrieved by a certain process.

2.2 Investigation of Existing Systems

There are quite a few such applications available both commercial and freeware which will utilise steganography to pass hidden information in media files such as images. I will take a look at some of the examples listed in the following table:

Date	Program	Price	Method
16-09-02	Camouflage	Freeware	Fuse
18-09-02	JpegX	Freeware	Fuse
21-09-02	InPlainView	$10	LSB
23-09-02	InThePicture	$25	LSB
29-09-02	Invisible Secrets 2002	$35	LSB
04-12-03	Safe&Quick Hide Files 2002	$20	Fuse
06-12-03	ImageHide	Freeware	LSB
03-01-04	Steganography 1.50 and 1.60	$25	Fuse
18-02-04	JSteg	Open Source	LSB
24-02-04	Cloak and DataStealth	Both $35	Fuse
24-02-04	FortKnox	$45	LSB
27-02-04	Data Stash	$20	Fuse

I will start with the most recent according to this table of research which is located at this [2] resource which also provided detailed analysis of each piece of software examined and I summarise forthwith.

"Data Stash" is an application which does not perform well under scrutiny as it appears to zip the hidden data and append this to the end of the carrier file which in the above research was an mp3 file. It claims to use "blowfish" encryption but this appears incorrect.

"FortKnox" is the next application to be analysed. The first error to be encountered was that if one wants to encode a sequence of zeros i.e. a byte which has "00"; it will not be able to do this! This company claims to be a market leader plus to be able to embed any file type within another which is a challenging claim as each file type is constructed differently and hence has to be accessed differently. Unfortunately it claims it can take any file as a carrier but treats them all as 24 Bmp which causes difficulties. Without defining the file type the program will modify the actual header data as opposed to the content thereby destroying the file in the process. In conclusion a negative experience.

"Cloak and DataStealth" is the next candidate on my list. The company claim steganography is a very advanced technology and our product is very advanced; however this program was broken in under an hour. It uses a "Fuse" method whereby the data to be sent is simply appended at the end of the file but thankfully does not corrupt the said carrier file. The data stream though is encoded in MD5 which I liked and is better than just plain sequential data encoding.

"JSteg" is an open source steganography suite which will actually encode information in a jpeg file using LSB instead of actually appending it through the fuse method. A good program which does exactly what it says on the tin.

"Steganography 1.50 and 1.60" uses the fuse method again. By simply looking for "FF D9" which is the EOL character in jpeg in a hexadecimal editor, one can see straight away the hidden data appears, e.g.:

```
00004A80   34 BA 2C BC F0 4C 42 78  40 3E 5C 55 5B DF F1 37   4¦,+_LBx@>\U[_±7

00004A90   F3 AB 6F FF 00 6E 7E B5  45 38 87 FF D9 50 4B 03   _½o_.n~¦E8ç_+PK.

00004AA0   04 14 00 02 00 08 00 6A  4C 78 2F C1 66 CC 56 FE   .¶.....jLx/-f¦V_

00004AB0   5D 00 00 81 63 00 00 0C  00 11 00 6A 65 6E 6E 69   ]..üc......jenni

00004AC0   66 65 72 2E 6A 70 67 55  54 0D 00 07 58 60 C1 3F   fer.jpgUT...X`-?

00004AD0   00 AF C5 3F 7D 3E C6 3F  ED BB 65 54 5C 4D B7 2E   .»+?}>¦?_+eT\M+.
```

Another feature claimed in the software is the use of encryption but further analysis this is simply the password stored in hashed format with the data not been encrypted at all.

"Jsteg" appears to be best tested here and my own application performs well against it.

2.3 Challenges of "A Windows Application for Secure Message Communication using Image Steganography"

The main challenges as I saw them were as follows:

- What technologies and platforms to use?
- What file formats to implement?

I answered the first question by deciding to use c# and .net Bitmap Class (hence Bitmap format) to implement the main coding and forms for the user interface.

- I also decided to implement the database as MsSql. The database would be connected via the Entity network utilising linq statements.
- Finally the website which would host the downloadable Application would be a simple website written in html and css.

2.4 Potential Benefits of this Image Processing Application.

This application has many potential benefits both in itself and compared to other applications.

- Firstly it uses the LSB method which I and many others consider superior to the fuse method.

True that the fuse method facilitates more file formats but I do not consider it even a proper option as it simply tags the file or message to the end of the carrier file.

- One aspect in my approach which is different to others is that all users must login to the application to a centralised database or either the user must register.

- Also my applications also allow the user to store the carrier files in the database for retrieval which will cancel the need of another transport medium for the carrier files, such as email if the user so wishes.

- Also the key for the message can be stored in the database as well as a record of all the messages in recent history.

This facility could become important as one has to be registered to use RSA and the same could become through for a lot of cyber security software particularly if a new cryptography algorithm was introduced at a later stage to upgrade this application.

3. Architecture

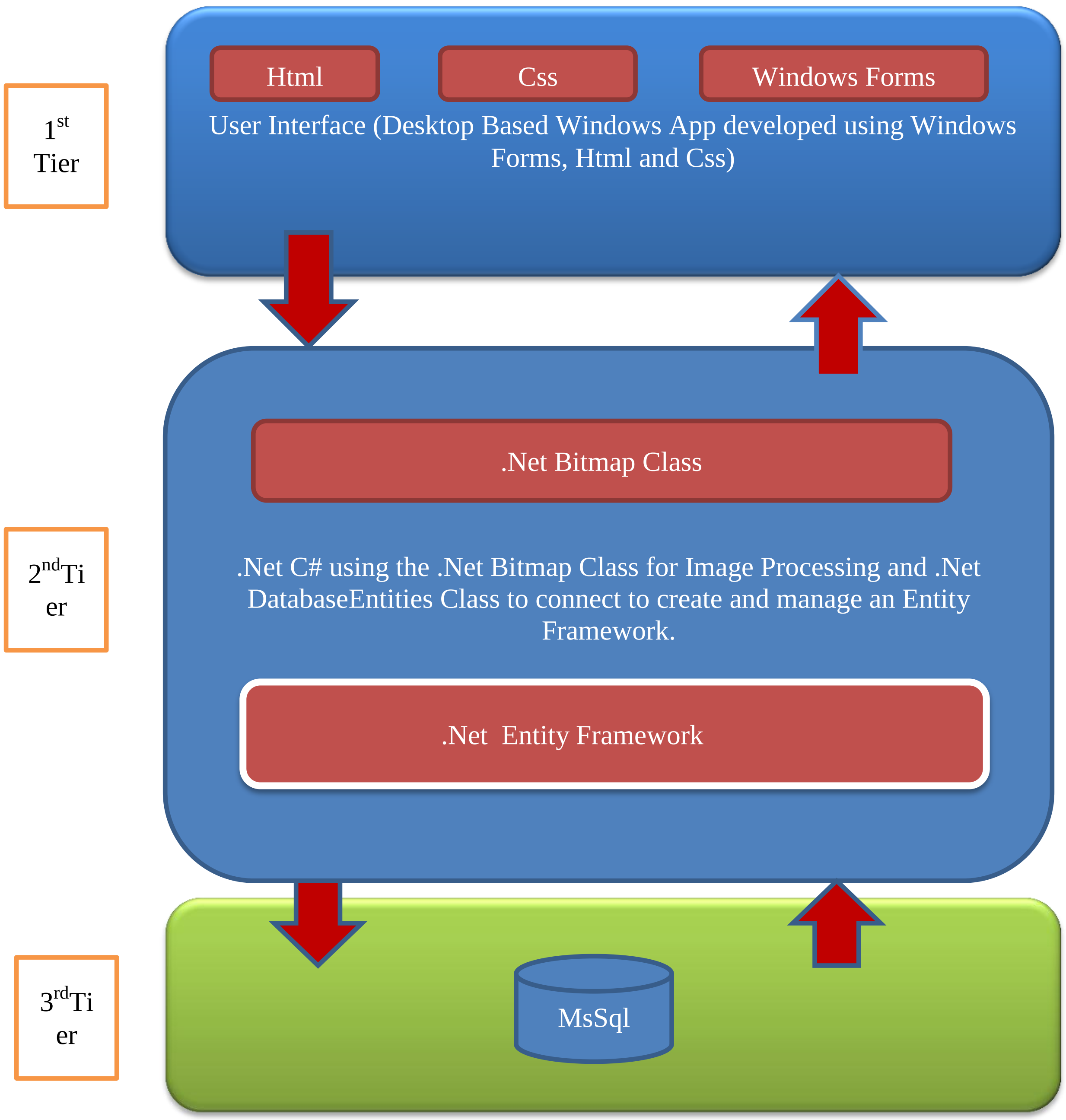
3.1 Architecture Design

This Chapter will attempt and hopefully succeed in showing the architecture of the project in a series of diagrams. The first diagram to show is the Architectural Design Diagram. It is broken down into the standard three tiers. I chose to use the three tier structure for the following reasons:

- Use of the 3 tier architecture make the application more understandable and easy to maintain and easy to modifications.

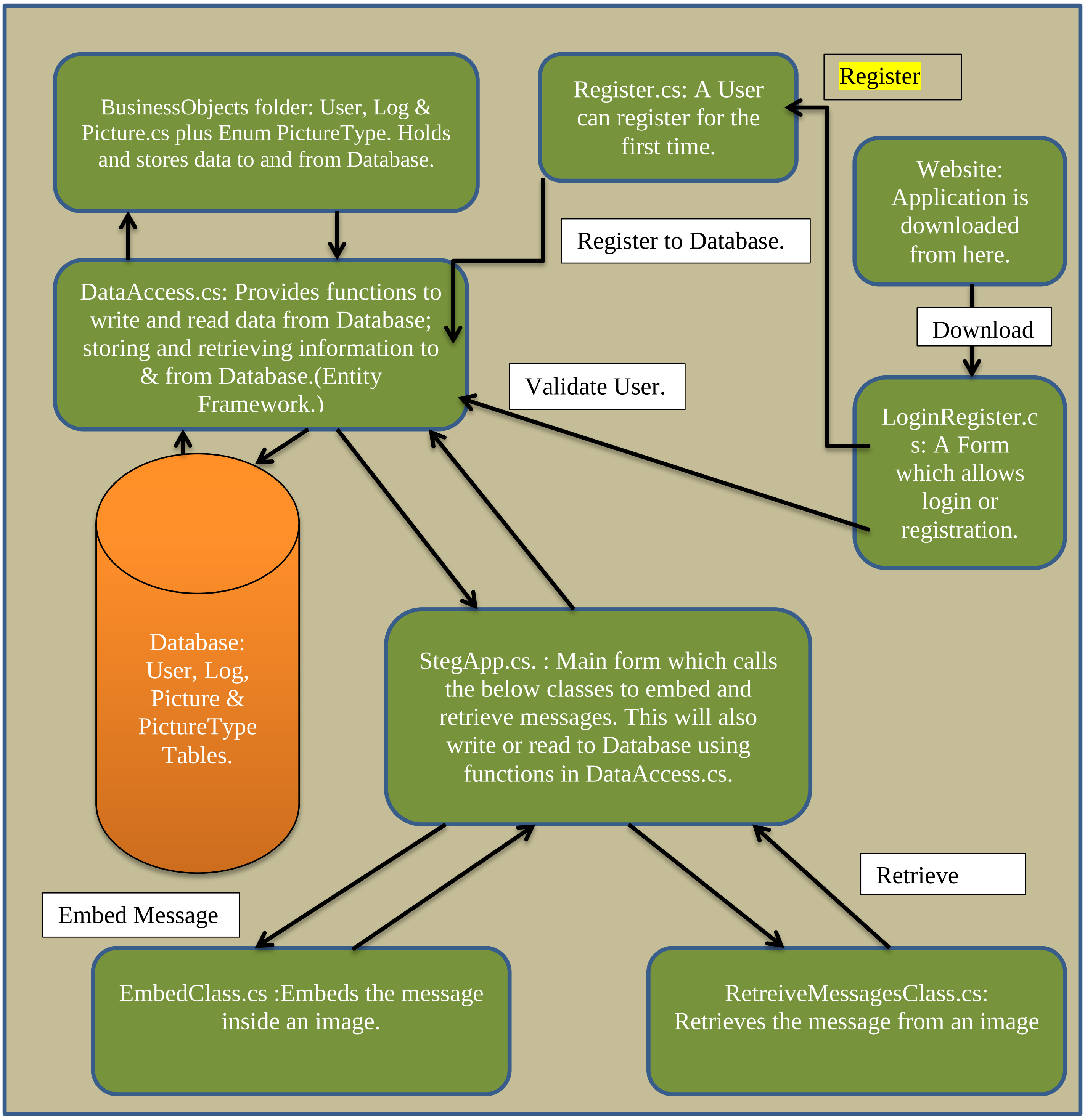
- Three tier architecture is secured, easily manageable, and highly understandable.

The above points explain themselves really. By splitting up the design in this way it is accommodating to changes and organizing how the code is written. The User interface will primarily display information or retrieve it - tier 1; tier-2 will process the data while tier-3 deals with storing the data or retrieving the data from a sorted location, in this case, the database. The first tier shows Html, Css and Forms used to construct the User interface. Tier One consists of a Website and three Windows Forms: A login form, a Registration form and finally the main form for the application: StepApp.cs'.Net' is used to construct the second tier. This Tier uses .Net c# Classes. While MsSql is used for the third tier. For tier three I use a MsSql Database and this connects to the .Net entity Framework objects in tier Two.


1st Tier
Html
Css
Windows Forms
User Interface (Desktop Based Windows App developed using Windows Forms, Html and Css)
2ndTier
.Net Bitmap Class
.Net C# using the .Net Bitmap Class for Image Processing and .Net DatabaseEntities Class to connect to create and manage an Entity Framework.
.Net Entity Framework
3rdTier
MsSql

3.2 Technical Design

The below technical architecture diagram shows how the various technologies are implemented in a number of c# files, windows forms, a website and an MsSql Database.


BusinessObjects folder: User, Log & Picture.cs plus Enum PictureType. Holds and stores data to and from Database.
Register.cs: A User can register for the first time.
Register
Website: Application is downloaded from here.
Register to Database.
Download
DataAccess.cs: Provides functions to write and read data from Database; storing and retrieving information to & from Database.(Entity Framework.)
Validate User.
LoginRegister.cs: A Form which allows login or registration.
Database: User, Log, Picture & PictureType Tables.
StegApp.cs. : Main form which calls the below classes to embed and retrieve messages. This will also write or read to Database using functions in DataAccess.cs.
Retrieve
Embed Message
EmbedClass.cs :Embeds the message inside an image.
RetreiveMessagesClass.cs: Retrieves the message from an image

This section will deal with the technical design of the project. Let me start with a brief introduction to how images are defined in the bitmap format but which can also relate to other image formats in general. Plus I will also introduce how to manipulate the least significant bit of each byte to encode data.

An image can be looked at as a series of pixels. Consider a 4x4 pixel image:

```
x x x x
x x x x
x x x x
x x x x
```

Number these pixels from 1 to 16:

```
01 02 03 04
05 06 07 08
09 10 11 12
13 14 15 16
```

Each pixel that is numbered above has a red component, a green component and a blue component. Each of those components are 1 byte each and so each component can be looked at as a value of 0 to 255. (24-bit = 8bits for red, 8 bits for green, 8 bits for blue). So each of the above numbers has 3 sets of values from 0 to 255.

So in the above example with a 4x4 image you have a total of 16pixels*3color_components = 48 bytes of data in your image. Typically what you will do is use only the least significant bit of each colour component to encode your image. In which case you would have 48 bits of data available for you = 6 bytes available to you to encode any 6 byte message you want. To make this easier though let's just look at encoding a simple 3 bit message into a single pixel. And let's assume we are only using 1 bit per colour component. Let's say we want to encode the 3 bit message: 111. Here is an example of the value **pixel 1** above has before you encode the data:

R: 10101011
G: 11111010
B: 00011010

What you do is change only the least significant bit to the new data:

R: 1010101 **1**
G: 1111101 **1**
B: 0001101 **1**

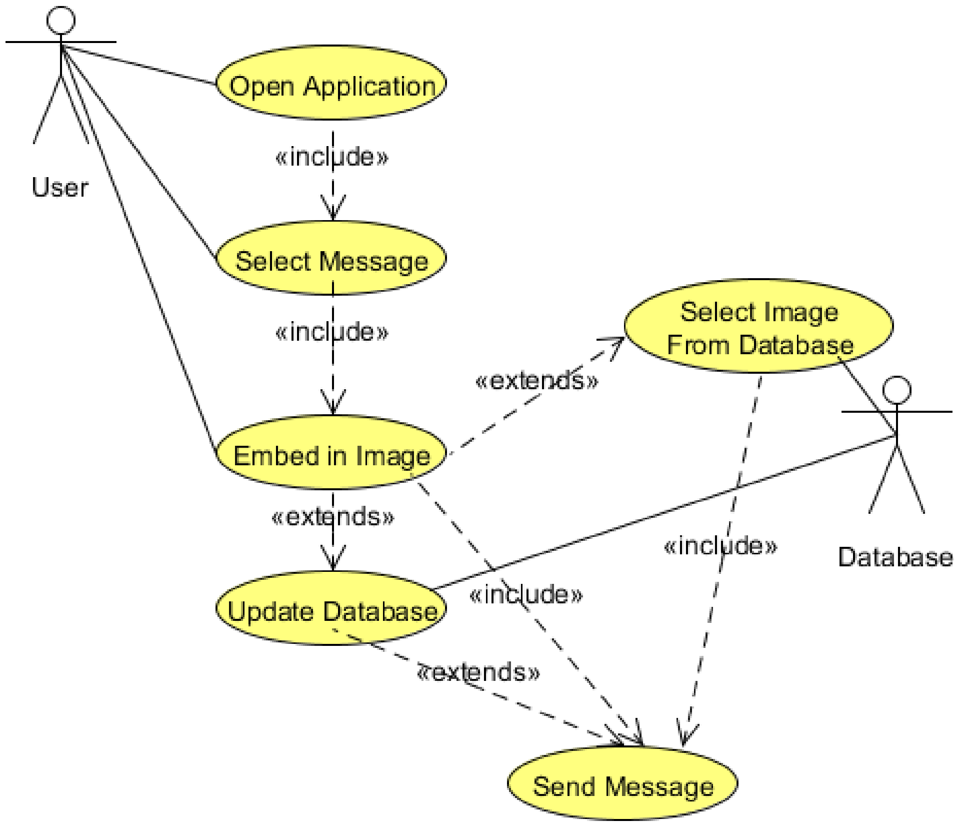
The pixel will look the same to the human eye, but now you are using the least significant bit to represent the data you wanted to encode. My program will only take 24 bitmap formats as input.

3.3 Process Flow

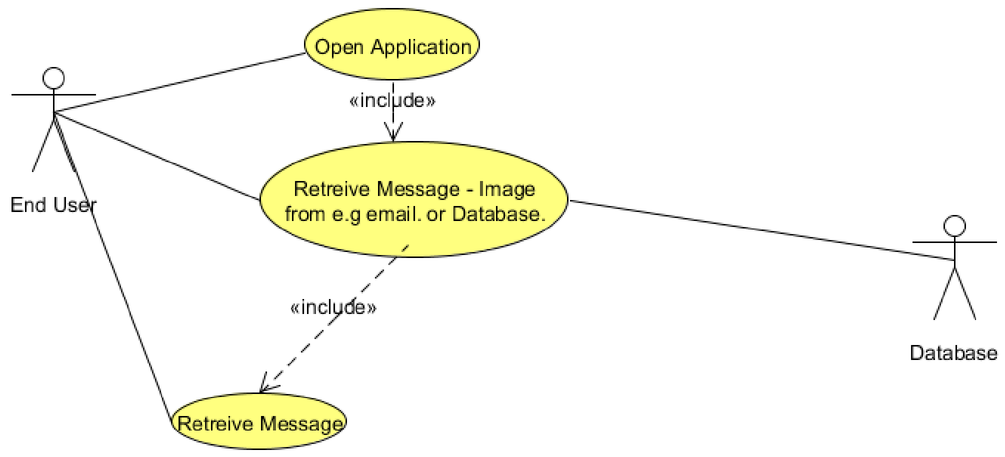
This section contains four sub sections: Case, Sequence & Activity Diagrams plus a class diagram.

3.3.1 Case Diagrams

User Case diagram:

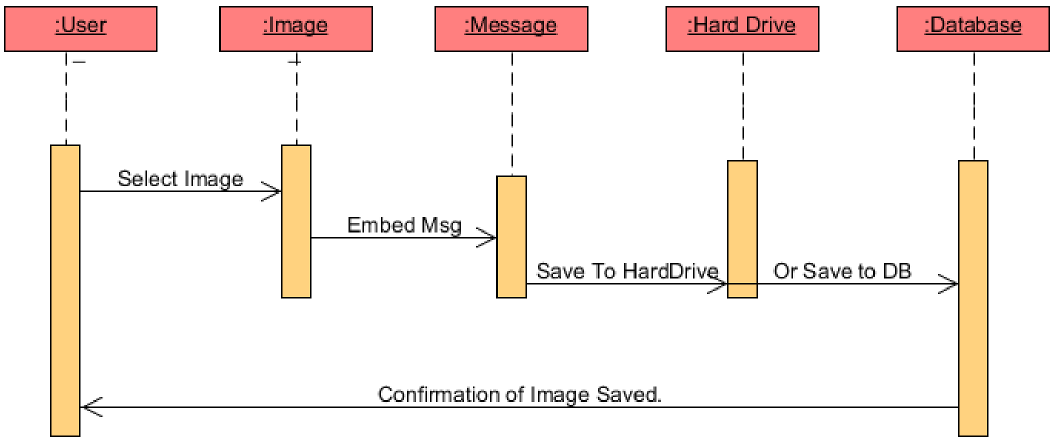


End User Case Diagram:

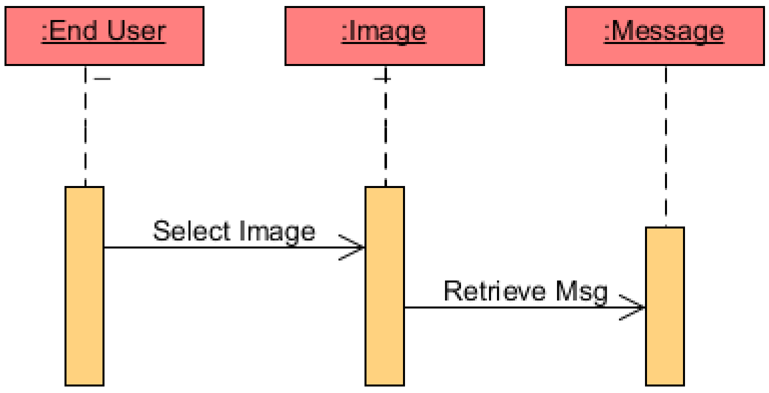


3.3.2 Sequence Diagrams

A sequence diagram for a ‘User’ sending a message:

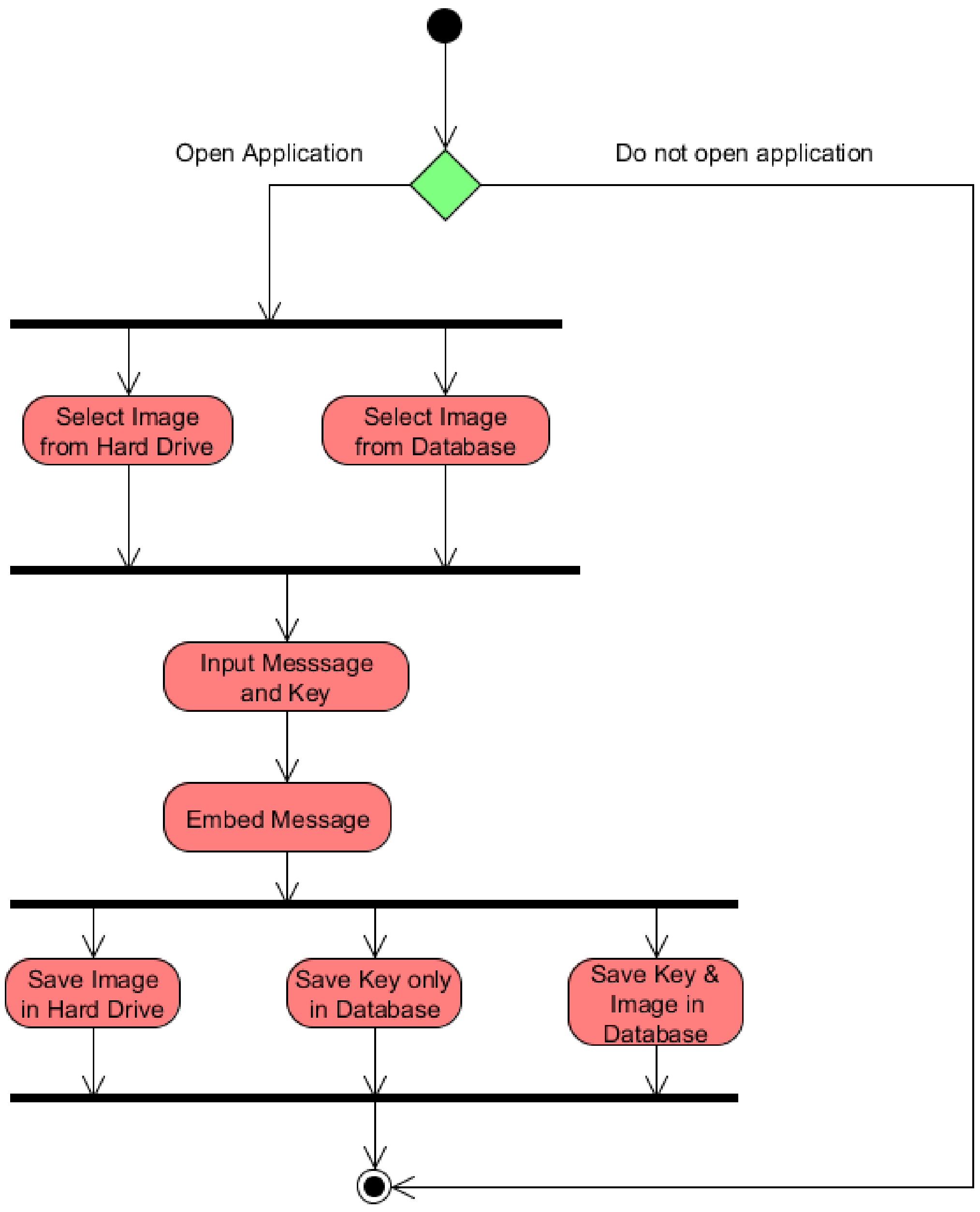


A sequence diagram for a ‘End User’ retrieving a message:

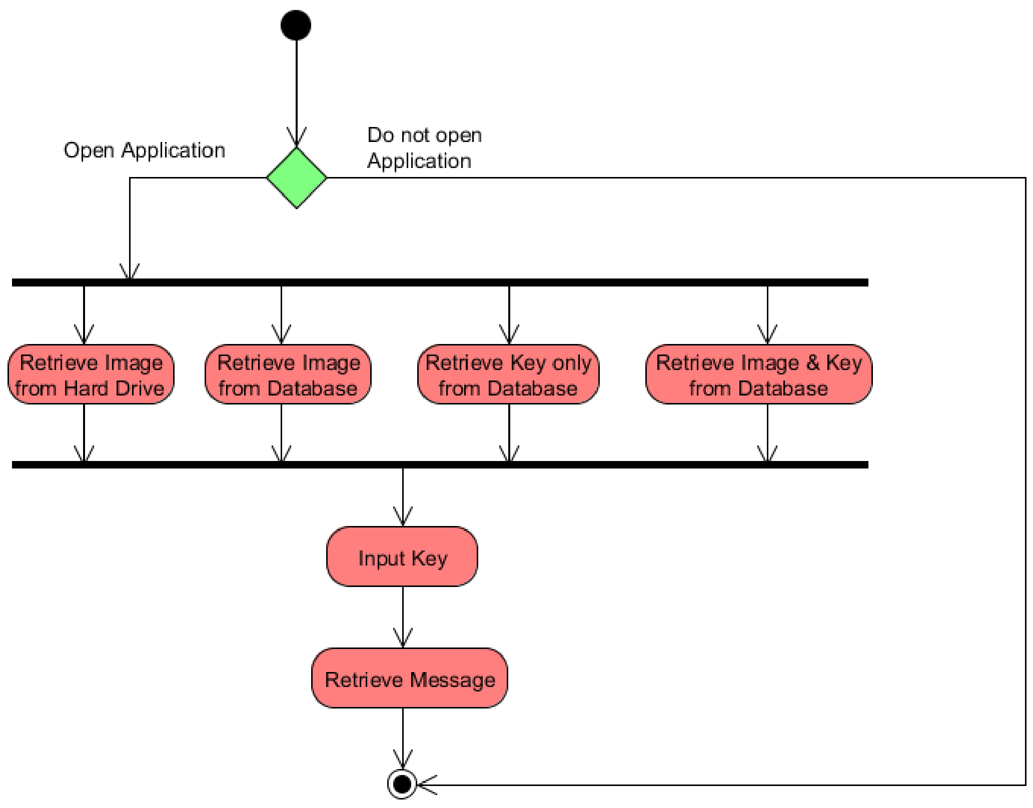


3.3.3 Activity Diagrams

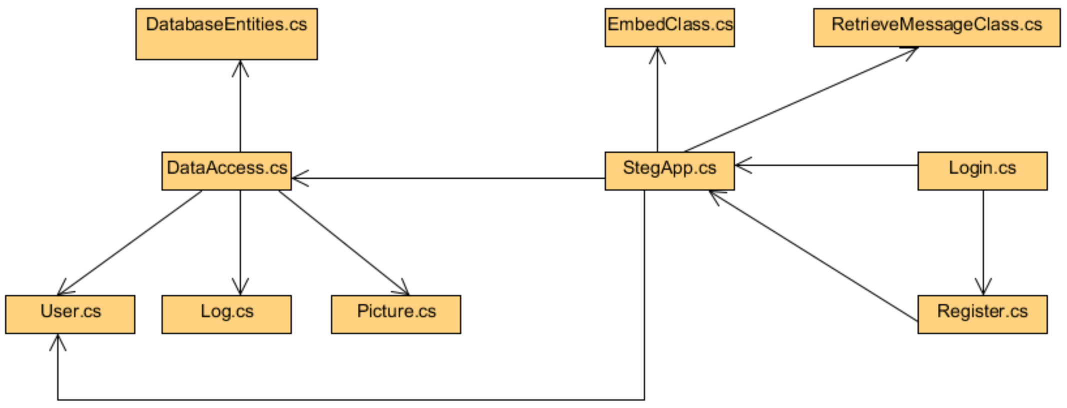
The Activity diagram for User sending a message:


Open Application
Do not open application
Select Image from Hard Drive
Select Image from Database
Input Messsage and Key
Embed Message
Save Image in Hard Drive
Save Key only in Database
Save Key & Image in Database

The Activity diagram for a ‘End User’ retrieving a message:


Open Application
Do not open Application
Retrieve Image from Hard Drive
Retrieve Image from Database
Retrieve Key only from Database
Retrieve Image & Key from Database
Input Key
Retrieve Message

3.3.4 Class Diagram



4.0 Implementation using Rational Unified Process Approach

This section describes the different phases in the implementation of this project: inception, elaboration, construction and transition:

Inception Phase: This phase required a lot of research into various areas. First, there was the business case for the project or application. My aims was to implement a project which would produce application which implement a front end, a back end with challenging and rewarding coding in the business Logic. This application achieved all these objectives. I would have a good user interface in the form of three windows forms. A working database implemented using MsSql technologies. Finally the subject of steganography and image processing proved a fulfilling experience in producing good classes using c# and .Net in the Visual Studio Development suite. Other significant points at this stage was the choice of algorithm. 'Least Significant Bit' proved the right choice in terms of technical competence necessary and timescale considerations.

Elaboration Phase: At this phase of the process I had to consider various architectural considerations. The first point to consider was to how to connect the Database to the Business layer. I chose Entity framework as the platform that which would mange the connection between my application and the Database. The next consideration was to consider the type of user interface. I chose forms over web forms because I was already familiar with this type of user Interface. I also made a decision that I would only implement the project for Bitmap images only and not investigate other types of images such as jpeg at a later stage if time permitted.

Construction Phase: There were two main challenges encountered in this phase. Unit testing and changes in architecture at the Business layer this necessitated. As shown in the Technical Design Diagram previously; StegApp.cs handles a lot of the processing for writing and retrieving to and from both the Users' Hard Drive and the Database. In order to do proper unit tests on some of the events I had to do the following. Simply but I made sure each event on the user interface called a separate function and execution was passed to these functions. I was then able to successfully unit tests a lot of these functions. Another point worth considering is that I use the real database for testing instead of the a dummy database or mocking network so some of the unit tests could be classified as integration tests because the tests involve the actual Database. But I believe this to be a minor issue and due to time constraints I would have implemented a dummy database or mocking framework.

Transition Phase: During this phase I analysed the results of the evaluation survey plus I had to rewrite some of the tier 2 logic. I created two classes: EmbedClass and RetrieveMessagesClass. These two classes provide better structure to the Business logic layer whose functionality was previously part of StegApp.cs. As stated in section five i had an evaluation of the application whose results were generally positive.

4.1 Phases in the Implementation

4.1.1 The Bitmap Class in .Net

In .Net there is a Bitmap class which I have utilised for this project. It takes a bitmap image as input.

```
image1 = new Bitmap(@"C:\Documents and Settings\All Users\"
         + @"Documents\My Music\music.bmp", true);
```

The above is a declaration of a bitmap object. ‘image1.Height’ will gave the number of rows of pixels and ‘image1.width’ will gave the number of columns, hence each pixel has an x and y coordinates.

```
try
   {
      // Retrieve the image.
      image1 = new Bitmap(@"C:\Documents and Settings\All Users\"
         + @"Documents\My Music\music.bmp", true);

      int x, y;

      // Loop through the images pixels to reset color.
      for(x=0; x<image1.Width; x++)
      {
         for(y=0; y<image1.Height; y++)
         {
            Color pixelColor = image1.GetPixel(x, y);
                          // Get the current pixel
            Color newColor = Color.FromArgb(pixelColor.R, 0, 0);
            image1.SetPixel(x, y, newColor);
         }
      }

      // Set the PictureBox to display the image.
      PictureBox1.Image = image1;

      // Display the pixel format in Label1.
      Label1.Text = "Pixel format: "+image1.PixelFormat.ToString();

   }
catch(ArgumentException)
   {
      MessageBox.Show("There was an error." +
         "Check the path to the image file.");
   }
```

Above is an example of code which utilises a bitmap object generated using an image. I have a nested for loop which will go through each pixel in turn via their respective x & y coordinates. I will now introduce another type or struct known as ‘Color’ which is used to represent an ARGB (alpha, red, green, blue) colour which is exactly what it does. The red, green and blue components are each represented by a number between 0 and 255.

As illustrated in the above code image1.Getpixel(x,y) will get that specific pixel depending on the values of x or y given. Likewise 'pixelColor.R' will return the red component.

```
Color newColor = Color.FromArgb(pixelColor.R, 0, 0);
            image1.SetPixel(x, y, newColor);
```

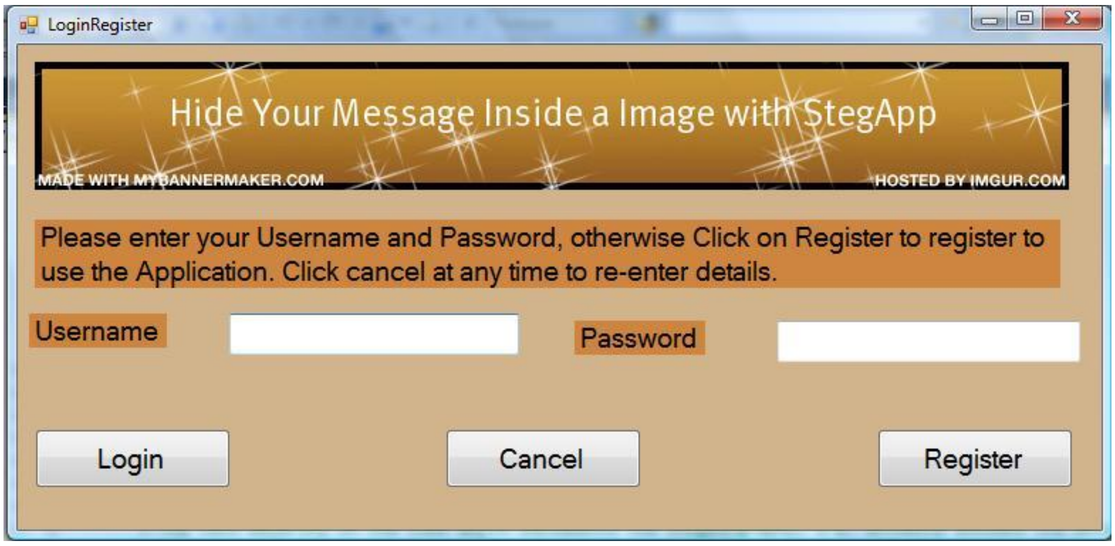
The above piece of code will create a new color struct but with the green and blue components set to zero. Finally the property SetPixel will change the pixel at a given coordinate with the new value supplied.

4.1.2 Windows Forms

The main user interfaces for my application are windows forms. In fact the whole project is a windows form with an 'Entity' object attached as well as other classes for specific purposes.

There are three forms in this application. A login screen will appear when the application is initially run. The user will login or they must click on the register button if this is the first time to use the application. The register screen is another form whereby the user can enter his details plus choose a username and password. At this point when registration is finished or login is verified; the user is directed to another form; the actual application.

Login Screen:



LoginRegister

Hide Your Message Inside a Image with StegApp

MADE WITH MYBANNERMAKER.COM HOSTED BY IMGUR.COM

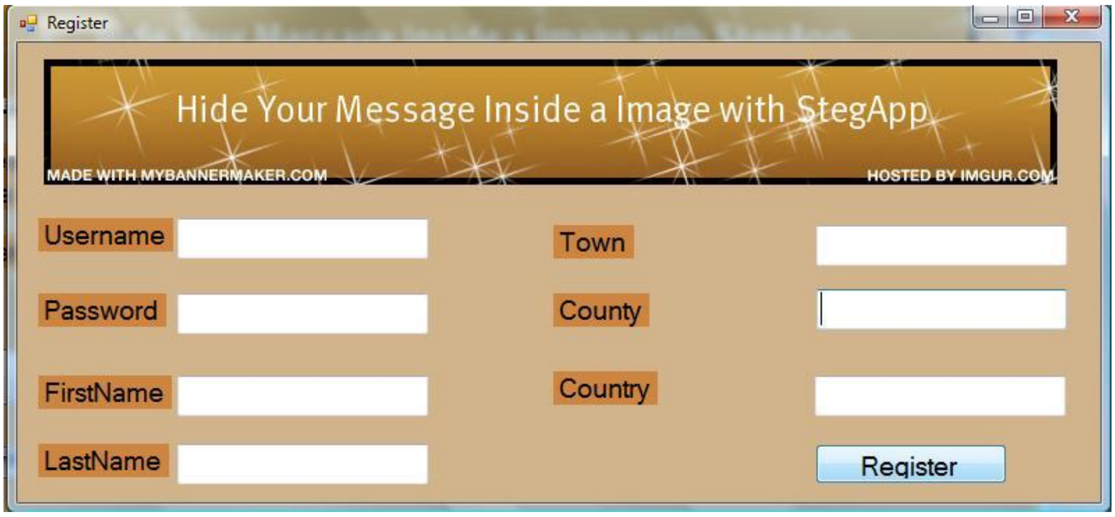
Please enter your Username and Password, otherwise Click on Register to register to use the Application. Click cancel at any time to re-enter details.

Username

Password

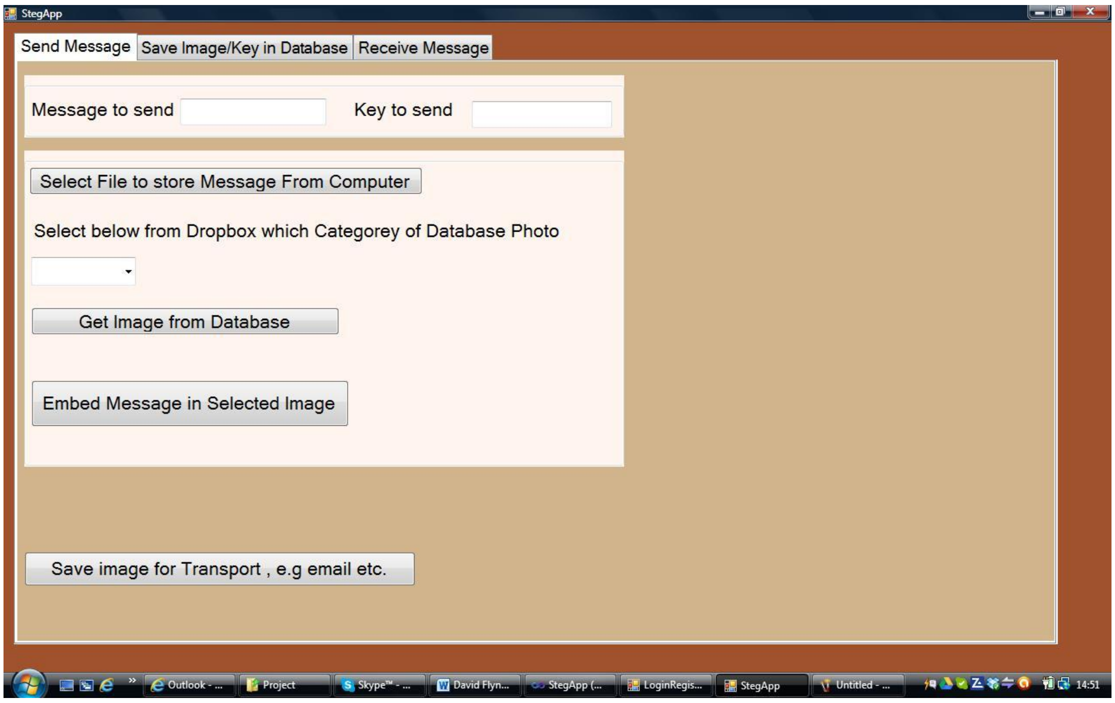
Login Cancel Register

Registration screen:

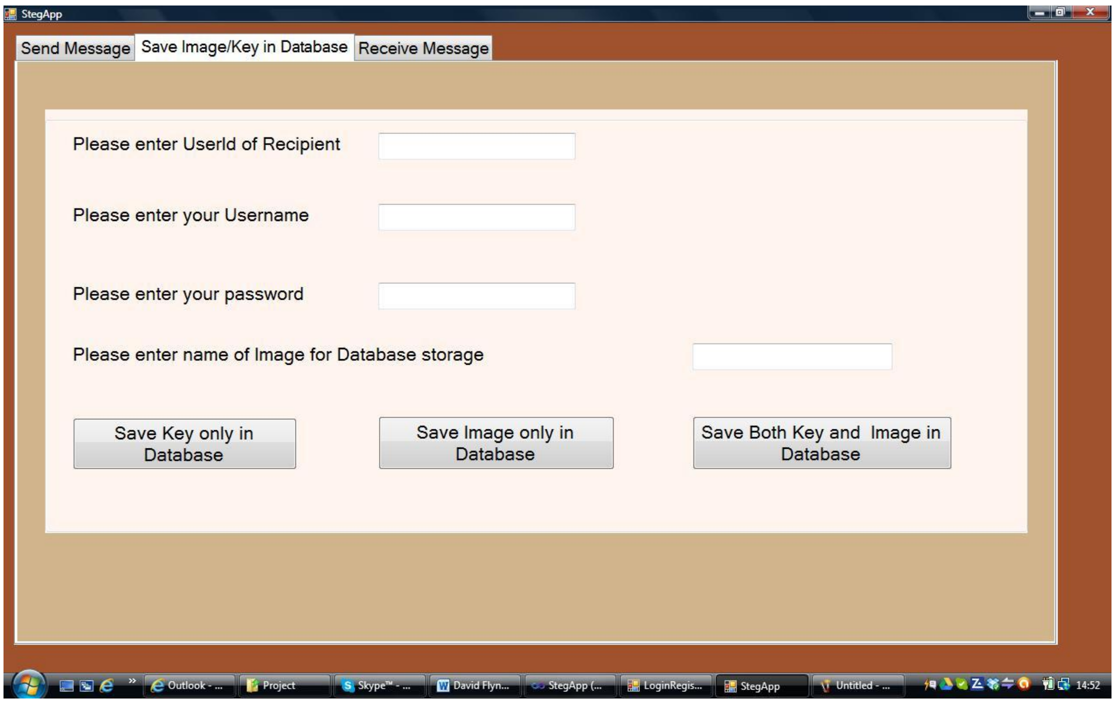


Screenshots of main form StegApp.cs:

Send Message screen:


StegApp
Send Message
Save Image/Key in Database
Receive Message
Message to send
Key to send
Select File to store Message From Computer
Select below from Dropbox which Categorey of Database Photo
Get Image from Database
Embed Message in Selected Image
Save image for Transport , e.g email etc.

Screenshot of ‘Save Image/key in Database’:


StegApp

Send Message
Save Image/Key in Database
Receive Message

Please enter UserId of Recipient

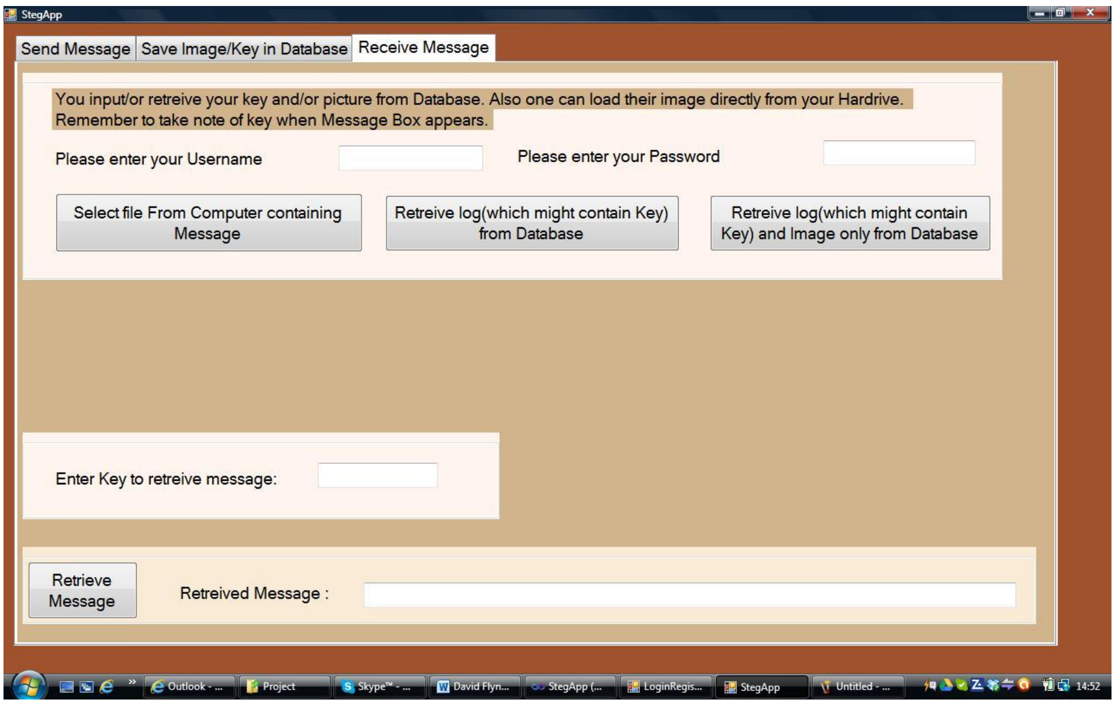
Please enter your Username

Please enter your password

Please enter name of Image for Database storage

Save Key only in Database
Save Image only in Database
Save Both Key and Image in Database

Screenshot of ‘Receive Message’:


StegApp

Send Message
Save Image/Key in Database
Receive Message

You input/or retreive your key and/or picture from Database. Also one can load their image directly from your Hardrive. Remember to take note of key when Message Box appears.

Please enter your Username

Please enter your Password

Select file From Computer containing Message
Retreive log(which might contain Key) from Database
Retreive log(which might contain Key) and Image only from Database

Enter Key to retreive message:

Retreived Message :

This form has three tabs; one for sending a message, one for saving image and or key in Database plus one for receiving a message. For sending a message, the user enters the message and key in separate text boxes. Then the actual image is selected either form the user’s hard drive or from the database. The user now embeds the message inside the 24bitmap image. Then the user has the choice of saving the image to the hard drive; save image in database, save only key in database or save both key and image in database. To use the Database option one must select the before mentioned tab.

To receive a message one must click on the receive message tab. If the user knows that a key is stored in the database; then the user clicks on the relevant button to retrieve the relevant log from the database provided they enter their username and password. The user can also load the image from the database if it is stored there. The user can also load the image from their own hard drive if necessary. Finally at this point the user presses the retrieve message button to retrieve the message which is shown in a text box.

4.1.3 Steganography Code

The vast majority of the code which implements the steganography, i.e. actually encodes the message inside the image and retrieves the message from a carrier file is code associated with the events of pressing two buttons: 'Embed Message' in the send screen and 'Retrieve Message' in the 'Receive Message' tab. The code is commented so I will try not to repeat myself but I hope what I write here will complement the comments in the code. 'StegApp.cs' is the file which is the before mentioned code.

I start off by declaring variables which will be used in both methods within both classes. These include string builders, strings, and ints' and byte arrays. Some of these are declared within the StepApp.cs or within EmbedClass.cs or RetreiveMessageClass respectively. 'button2_Click' is the function which implements the embed functionality. It declares an object of the EmbedClass and accesses the method Embed which is explained in the next section: 3.2.3.1.

The code behind the 'Retrieve Message' button is implemented in the function button4_Click(object sender, EventArgs e). It declares an object of RetrieveMessageClass and calls the Retrieve Method within RetrieveMessagesClass. This code and class is explained in 3.2.3.2.

EmbedClass.cs: The EmbedClass.cs is a class I created specifically for embedding a message inside a image. Within this class I have a public method:

`public int Embed(string Embedkey, string EmbedMessage, Bitmap image3)`

Which takes as parameters the Key, the message and the image as a bitmap object.

Firstly, there is a while loop which forces the user to enter a key if one is not present plus makes sure it is no longer than six characters. I then create a four character string(temp5) which contains the length of the message in decimal format, e.g. if the message is say 20 characters, then temp5 would be '0020'. Finally I concatenate the key, the message length and the message into one string temp4. Now I must convert this string to a byte array b1. I then have a for loop which iterates through each byte of the b1 array: Each element of the array 'b1' contains the decimal ascii of each character in temp4. The while loop will convert each element of the byte array to its equivalent binary string and store it in the string:' tmp'. Each value of tmp is sequentially stored in the string array tmp2.

The next for loop concatenates each element in tmp2 to tmp3. What follows is redundant piece of code which appends either '00' or '01' to start of the binary string tmp3 depending on whether a key is used or not. However I have changed the piece of code to only accept a key so '00' is just appended at the beginning of tmp3 anyway. 'tmp2' is then appended to 'sb' the string builder which is initialised as zero. 'stringlenght' is the length of 'sb'. I have two masks, 'Mask0' which is set to 254 and 'Mask1' which is set 1. I will explain these shortly.

What follows next is a nested for loop whereby the first for loop counts the number of columns and the next for loop counts the number of rows. These for loops will naturally stop when the each pixel has been accessed in the image and also when the nested loop has iterated the same number of times as the length of the string. The loop will determine whether the current bit in sb is either a zero or one. If zero then the relevant pixel in sequence is and-ed with Mask0 and when it is a one; then the relevant pixel is or-ed with Mask1. The pixels are accessed in sequence by red, green and blue. This method returns a int to its call which determines if a message box must be displayed indicating the key is of the wrong length or the message is too long.

RetrieveMessageClass.cs : This class has one method: Retrieve which takes the arguments: the key entered by the end user and image they are trying to retrieve the message from. This code is very similar to that which has been explained above. There are three nested for loops because the key and message length has to be retrieved along with the message itself. The conversion from binary string back to ascii is code which is unique to this method. An example is given below:

```
KeyRetreived = KeyMessage.ToString();
int count = KeyMessage.Length / 8;
var StringBytes = new byte[count];
for (int i = 0; i < count; i++)
  StringBytes[i] = Convert.ToByte(KeyRetreived.Substring(i * 8, 8), 2);
string FinalKey = enc.GetString(StringBytes);
```

In the above piece of code 'KeyRetreived' is the string builder from which the binary string representing the key is retrieved form the carrier file. It is converted to a string. 'count' is the number of eight bit strings in sequence in the said string. 'StringBytes' is a bytes array of length count. I then have a for loop which iterates through StringBytes converts each eight bits in sequence to a byte. I use the substring function to access the eight bits in sequence. I then use the encoding class in .net to turn StringBytes back to a string.

I follow the same procedure to attain the message length and the actual message. The key attained from the image is compared with what the user has entered into the text box. If they are the same proceed; otherwise try again. If the keys match; attain the message as described above and return this string to the function call in the form which displays the message.

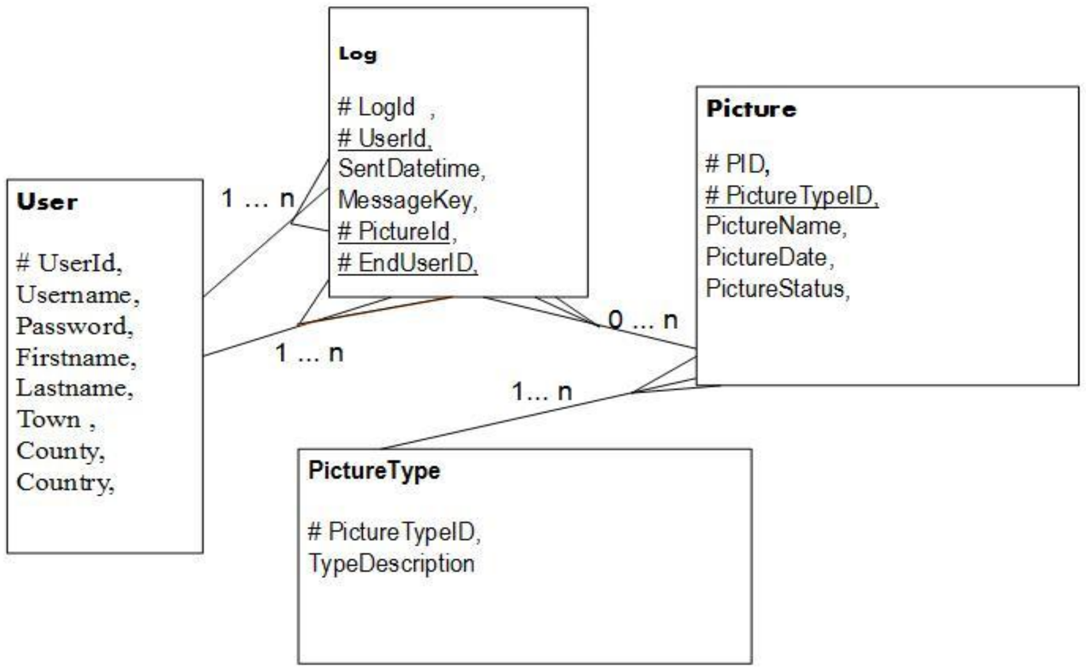
4.1.4 Image storage and Retrieval

This section deals with how the images are brought into the application and saved. I will deal with the option of storing an image inside a database in a further section. 'btnSelectFile_Click(object sender, EventArgs e)' is the function which allows the user to select a file in order to be used as the carrier for the message. I use the dialog class to open a simple user interface for the user to select a file from their hard drive and I then set this image to display in the picture box.

button5_Click(object sender, EventArgs e) is the function I use to save the carrier file with the embedded message. I use the saveFileDialog class to allow the user to save the file to disk. The event handler calls the function SaveImageDisk() which handles the dialog.

I use try catch statements in both functions in order catch any error which causes either function to fail.

4.2 Database



Above is a representation of the database which I am using for this project. It should be fairly self-explanatory. The user table contains the details of each registered user; UserId is the primary key. The log table contains details of each message sent for which the user wants to keep a record in the database. 'LogId' is the primary key. 'UserId' and 'EndUserId' are both foreign keys relating to 'UserId' in the User table. 'PictureId' is foreign key to the primary key: 'PictureId' in the Picture table. The Picture table contains the details of all pictures stored in the database.'PictureStatus' determines whether the picture is locked by another user. Finally PictureType determines what type of picture used. There are four categories of picture: Animals, Sea, Sky and Earth. All the primary keys are seeds. The Database is hosted at stem.arvixe.com for I have admin rights and I build this Database using Sql management Studio on my Machine connecting remotely to stem.arvixe.com.

4.2.1 Business Objects: User.cs, Log.cs and Picture.cs

I decided to use the Enity Network to connect my database to my project. In order to save data or retrieve data from my database I needed to create a class which would act as an intermediary in that it would mimic the said table's type for type and name for name.

So for example, let's have a look at User.cs. This contains all the variables as they appear in the User table. So when I retrieve a User's data form the database I can temporarily store the data in a Business.User object and also easily update the User's data by building a function to save data to the database which uses the User class. Likewise for the Log and Picture classes.

DataAccess.cs : This class implements all the methods for accessing, updating and saving data to the database. I have created a private property ConnectionString which houses the connection string that any object which wants to connect to the database must use. The objects that do connect to the database are instances of the 'DatabaseEntities' Class which is the class that is created by the Entity Framework to handle communication between the database, the framework and any class which wishes to access the database. Most of these methods are called from the relevant forms such StegApp, LoginRegister & Register which I will explain in a later section.

The first Method is AuthenicationCredentials which returns a Business User object and authenticates a user depending on two input arguments: username and password. I have a 'using' statement which declares a

```
var dataContext = new DatabaseEntities(ConnectionString)
```

'datacontext' is the local object which is an instance of DatabaseEntities which connects to the database using 'ConnectionString'.

```
var user = dataContext.Users.FirstOrDefault(u => u.Username.Equals(username) && u.Password.Equals(password));
```

The above line declares another variable user; datacontext.Users is the User table in the database while

```
dataContext.Users.FirstOrDefault()
```

accesses the first or default row in table according to the condition set in the brackets. In the brackets I have the following statement:

```
u => u.Username.Equals(username) && u.Password.Equals(password)
```

which specifies row u or object u returned where the username and password match what is passed to that linq statement, namely username & password. If no row is found which matches the requirements set by the linq statement; then null is returned which is what is returned to the function call. If a match is found CurrentUser(a Business.User object) is assigned the object returned by GetUser(userid). This function uses the following statement:

```
var user = dataContext.Users.FirstOrDefault(u => u.UserId == userId);
```

To search and return the entry in the table that corresponds to that user id. If one is found then a new Business.User object is returned which correlates exactly with its counterpart in the database.

'ObtainUser' is the same as 'GetUser' except identification is done by 'username'.'InsertUser' will create a new entry in the user table in the database using the arguments that it is called with. It creates a new datacontext object and a new Business User object which is assigned the values passed into the function. Then this object is added to the Users table by calling AddObject() and SaveChanges() to save it.

'UpdateUser' will insert the contents of the Business User object passed to it into the database table User again a datacontext variable and linq statements. 'InsertLog' is used to insert a log entry into the log table in the database. A new Log object is created which values are set to what has been passed to the function. Then the object is added to the database and saved. I have two different insert log functions for the case that three of four arguments are passed to the function.

The next function is RetreiveMessages(). This function will return the log object corresponding the relevant log entry in the log which corresponds to the userid passed to the function. I wanted to be able to pull the last or default entry which corresponds to that user id but DatabaseEntities objects does not have the option to use LastOrDefault even if it is listed!? So I had to use the following code :

```
var u = from p in dataContext.Logs
        where p.EndUserID == userid
        orderby p.SentDatetime descending
        select p;
  var u1 = u.FirstOrDefault();
```

to retrieve the last entry in the log table which corresponds to that user id. I get around this by sorting the relevant logs by descending date and assigning the first entry of that list to a new variable u1. Then I proceed as before and return the correct log object requested.

‘SaveImage’ will save the image passed to it in the Picture table in the database. Notice that I had a different approach in using the following:

```
var u = dataContext.Pictures.CreateObject();
```

To create the actual object and modify its elements directly and then add and save them. ‘GetPicture’ returns picture object when known Picture Id is passed to it.

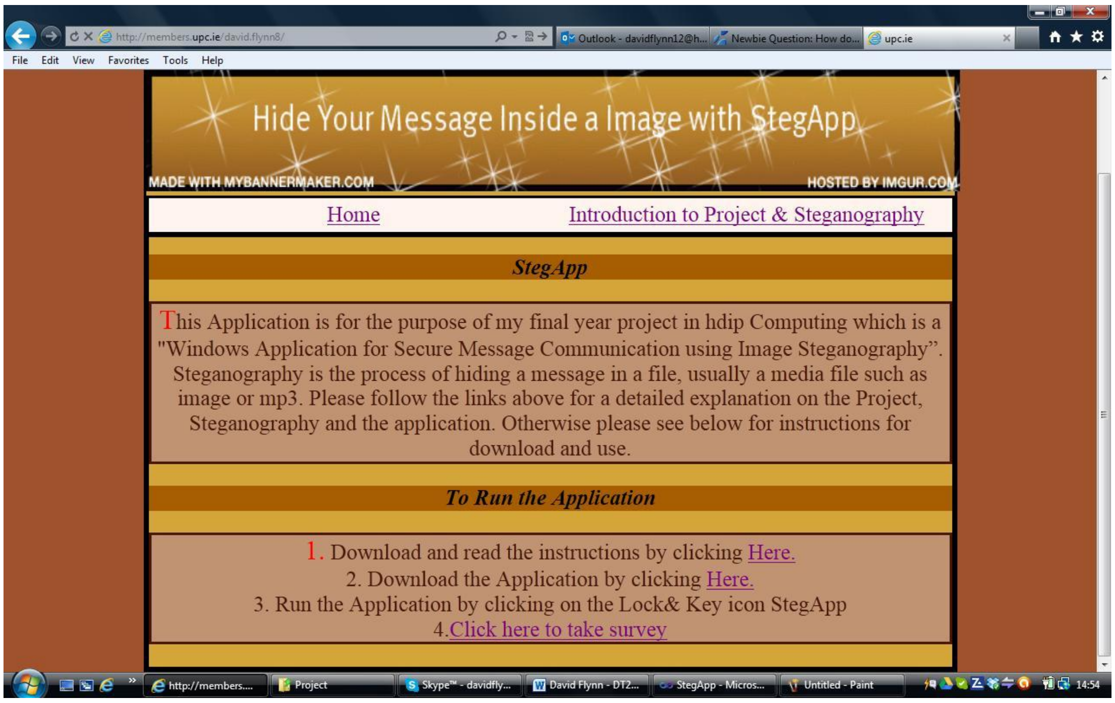
LoginRegister.cs: This class implements the functionality behind the loginRegister form which is the first form the user is presented with. The user enters his/hers username and password. When the login button is clicked I create an instance of DataAccess.cs and call the AuthenticateCredentials function passing the username and password from the text boxes. If verified I make this form invisible and open a new instance of StegApp.cs form which is the main user interface. If not verified I present a message to enter correct details or press the register button to register.

Register.cs: When the register button is clicked on the login form, a new form is open which allows a new user to enter their details. The register button calls the RegisterFn. Inside this function I declare an object of DataAccess and I call the InserUser function passing all the information in the relevant text boxes.

4.3 Implemented Website used in Downloading the Windows Application

A web-site was created which will allow the user to download the application. The website is http://members.upc.ie/david.flynn8. It consists of two web pages. The first page consists of a brief introduction to the application with instructions on how to use and download the application. The other page which details project with an introduction to steganography with a link to download a brief History of steganography.

Screenshots of Website Home page and then ‘Introduction to Steganography’:

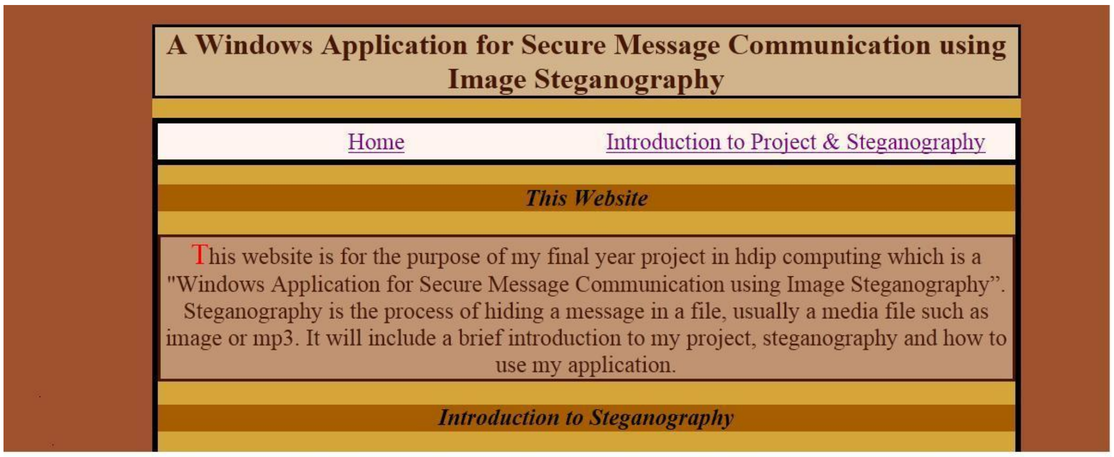

http://members.upc.ie/david.flynn8/
Outlook - davidflynn12@h...
Newbie Question: How do...
upc.ie
File Edit View Favorites Tools Help
Hide Your Message Inside a Image with StegApp
MADE WITH MYBANNERMAKER.COM
HOSTED BY IMGUR.COM
Home
Introduction to Project & Steganography

StegApp

This Application is for the purpose of my final year project in hdip Computing which is a "Windows Application for Secure Message Communication using Image Steganography". Steganography is the process of hiding a message in a file, usually a media file such as image or mp3. Please follow the links above for a detailed explanation on the Project, Steganography and the application. Otherwise please see below for instructions for download and use.

To Run the Application

1. Download and read the instructions by clicking Here.
2. Download the Application by clicking Here.
3. Run the Application by clicking on the Lock& Key icon StegApp
4.Click here to take survey

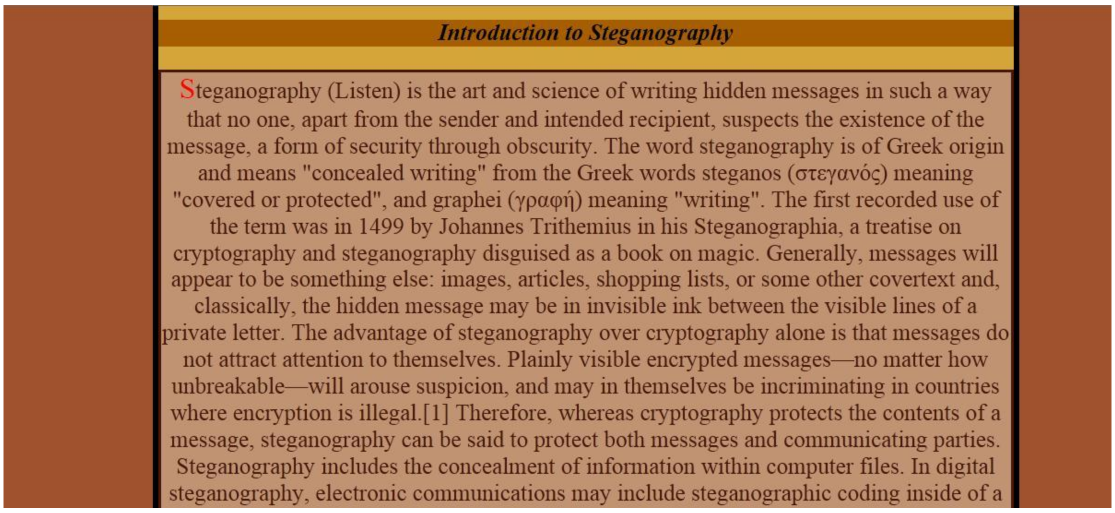

A Windows Application for Secure Message Communication using Image Steganography

Home   Introduction to Project & Steganography

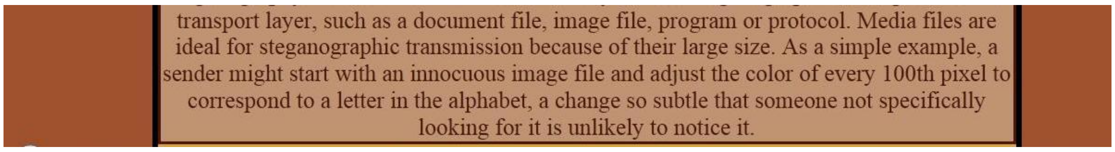
This Website

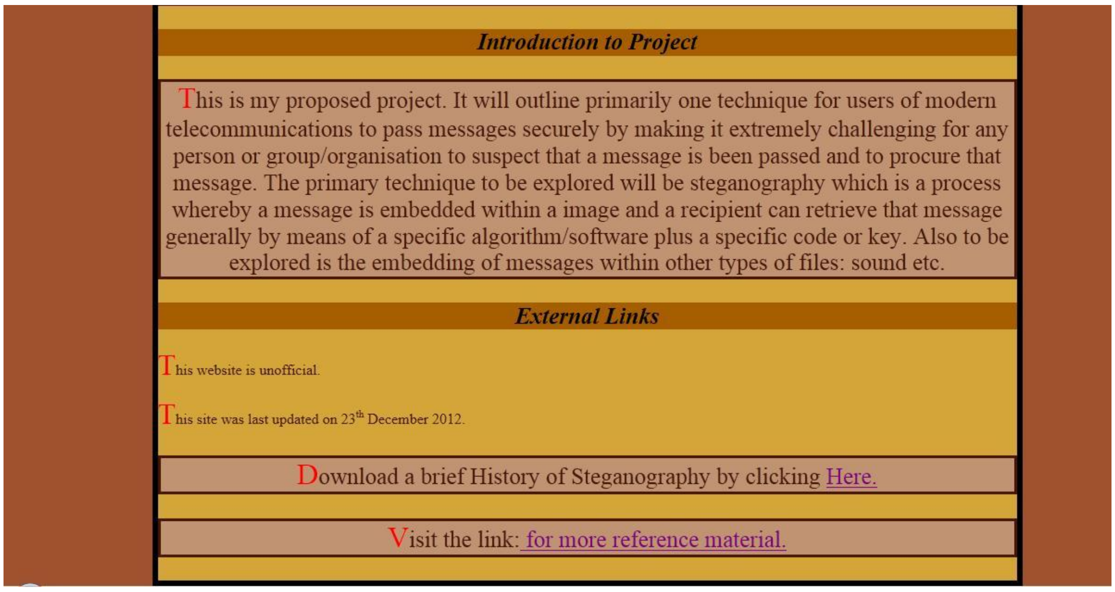
This website is for the purpose of my final year project in hdip computing which is a "Windows Application for Secure Message Communication using Image Steganography". Steganography is the process of hiding a message in a file, usually a media file such as image or mp3. It will include a brief introduction to my project, steganography and how to use my application.

Introduction to Steganography


Introduction to Steganography

Steganography (Listen) is the art and science of writing hidden messages in such a way that no one, apart from the sender and intended recipient, suspects the existence of the message, a form of security through obscurity. The word steganography is of Greek origin and means "concealed writing" from the Greek words steganos (στεγανός) meaning "covered or protected", and graphei (γραφή) meaning "writing". The first recorded use of the term was in 1499 by Johannes Trithemius in his Steganographia, a treatise on cryptography and steganography disguised as a book on magic. Generally, messages will appear to be something else: images, articles, shopping lists, or some other covertext and, classically, the hidden message may be in invisible ink between the visible lines of a private letter. The advantage of steganography over cryptography alone is that messages do not attract attention to themselves. Plainly visible encrypted messages—no matter how unbreakable—will arouse suspicion, and may in themselves be incriminating in countries where encryption is illegal.[1] Therefore, whereas cryptography protects the contents of a message, steganography can be said to protect both messages and communicating parties. Steganography includes the concealment of information within computer files. In digital steganography, electronic communications may include steganographic coding inside of a


transport layer, such as a document file, image file, program or protocol. Media files are ideal for steganographic transmission because of their large size. As a simple example, a sender might start with an innocuous image file and adjust the color of every 100th pixel to correspond to a letter in the alphabet, a change so subtle that someone not specifically looking for it is unlikely to notice it.


Introduction to Project

This is my proposed project. It will outline primarily one technique for users of modern telecommunications to pass messages securely by making it extremely challenging for any person or group/organisation to suspect that a message is been passed and to procure that message. The primary technique to be explored will be steganography which is a process whereby a message is embedded within a image and a recipient can retrieve that message generally by means of a specific algorithm/software plus a specific code or key. Also to be explored is the embedding of messages within other types of files: sound etc.

External Links

This website is unofficial.

This site was last updated on 23th December 2012.

Download a brief History of Steganography by clicking Here.

Visit the link: for more reference material.

Other Events: This section will deal with other events and the code, functions, methods and classes behind them.

The first event I want to deal with is the “Get Image from Database” button on the send tab of the StegApp.cs form. It calls the function:

```
GetPictureFromDataBase(comboBox1.Text)
```

A picture Entity Business object is declared and depending on the text of the drop down box a picture is pulled from the Database. This is done using GetPicture, a method of DataAccess.cs. As an argument it takes the picture id which is generated using the static function RandomPicture which picks an Id within the range that that particular category of photo is stored. This function will return a picture object which is assigned to the local picture Entity Object. The actual picture is a byte array stored in ‘picture.PictureData’. This is passed from memory stream to image to bitmap and finally to the Picturebox in the ‘Send’ tab.

The next event to consider is ‘Save Key in Database’. This will call the SaveKeyOnly function. The arguments it takes is the user’s username, password; the key and the UserId of the intended of the recipient. When a user decides to send a key, image or both via the database, it must be recorded somewhere for whom the actual key is for; so the UserId of the recipient must be entered in one of the text boxes shown. A Business user object p is declared and a DatabaseEntities object is declared. Using this object; AuthenicateCredentials is called to verify the user and the result is stored in the local Business User Object. If the user is valid then the InsertLog function is called; the version which takes three arguments: the user, the key and the id of the end user. If the user credentials are invalid; then a message box appears informing the user to enter the correct details.

The button ‘Save Image in Database’ calls the function:

```
SaveImageFn(UsernameTextBox.Text, PasswordtextBox1.Text,
ImageNametextBox1.Text, EndUsertextBox1.Text);
```

The function is virtually the same as the previous except it will call the four argument version of the InsertLog function which takes the pid, i.e. the picture id of the image stored in the Picture table of the database. This id is a primary in that table and a foreign key in the log table. The picture id is obtained by the following piece of code:

```
Int Pid = u.SaveImage(ImageName, BitmapToByteArray(image1, System.Drawing.Imaging.ImageFormat.Bmp), BusinessObjects.PictureType.General);
```

'Pid' is returned by calling the SaveImage function via the DatabaseEntities object u. It takes the name of the image which was passed to `SaveImageFn`. The image is converted to a byte array for storage by the function `BitmapToByteArray`. Finally the enum `BusinessObjects.PictureType.General` is passed also which is a category I gave to all images stored in the Database by Users. I simply put "Nokey" where the key is normally stored in the log.

Then we have the button "Save both Key and Image in Database". This will call the function:

```
SaveImageAndKeyFn(UsernameTextBox.Text, PasswordtextBox1.Text, ImageNametextBox1.Text, EndUsertextBox1.Text, textBox4.Text);
```

Which will store the key and image in the database. It is virtually the same as the previous function except that the key is stored in the relevant log.

In the receive tab of the form called StegApp.cs; there are two events which must be accounted for. The first such event is 'Retrieve log (which might contain key) from Database'. This event is directed to the function:

```
RetreiveKeyFn(UserNtextBox.Text, textBox1.Text);
```

Using the object Retreivelogs(a DatabseEntities object) declared outside this function I authenticate the user using AuthenicateCredentials. If this returns valid I then proceed to check if there are any logs in the database for the user Id of the validated user. The most recent log is the one attained. If no logs exist, then a message appears and informs the user of such; other the log is printed to a message box. Finally if a user is not validated; then a message box is shown for this also. The other event is the button: "Retrieve log and Image only from Database" which will retrieve the log and the associated image form the database. This event is passed to the function: RetreiveImageFn. This function is similar to RetreiveKeyFn except for the following. An extra if else statement is added which checks if there is a picture Id in the retrieved log:

```
RetreiveLogs.RetreiveMessages(p.UserId).PictureId
```

The above statement is evaluated to '0'. If it is zero, then a message box is displayed which informs the user that there is a picture associated with the most recent log. Otherwise a'using' statement is implemented that uses both the RetrieveMessages function to get the picture id and pass it to GetPicture; the returned byte array is put into a memory stream which is converted to a bitmap object and displayed in the relevant picture box.

5. Testing and Evaluation

The chapter will be broken down into two sections. The first will with evaluation and the next section is testing.

5.1 Evaluation

For evaluation I decided to use survey monkey for and this survey as an anchor on the website for the project http://members.upc.ie/david.flynn8. I then emailed to my survey group with a link to the above website informing them of how to access the survey on the website. Screenshots of the actual survey are in Appendix 1. It is hosted by surveymonkey.com. The first page details terms and conditions while the second page details the actual survey which consists of eight questions.

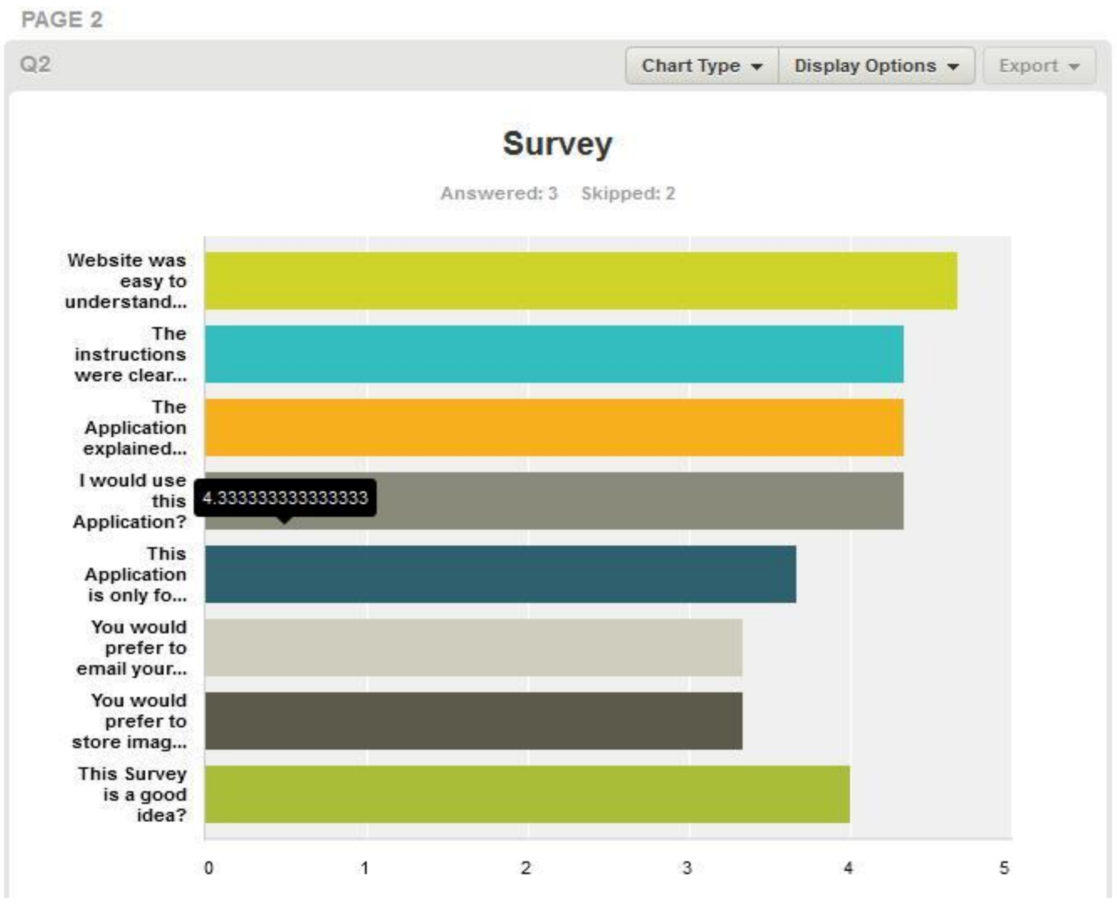
5.1.1 Survey Questions

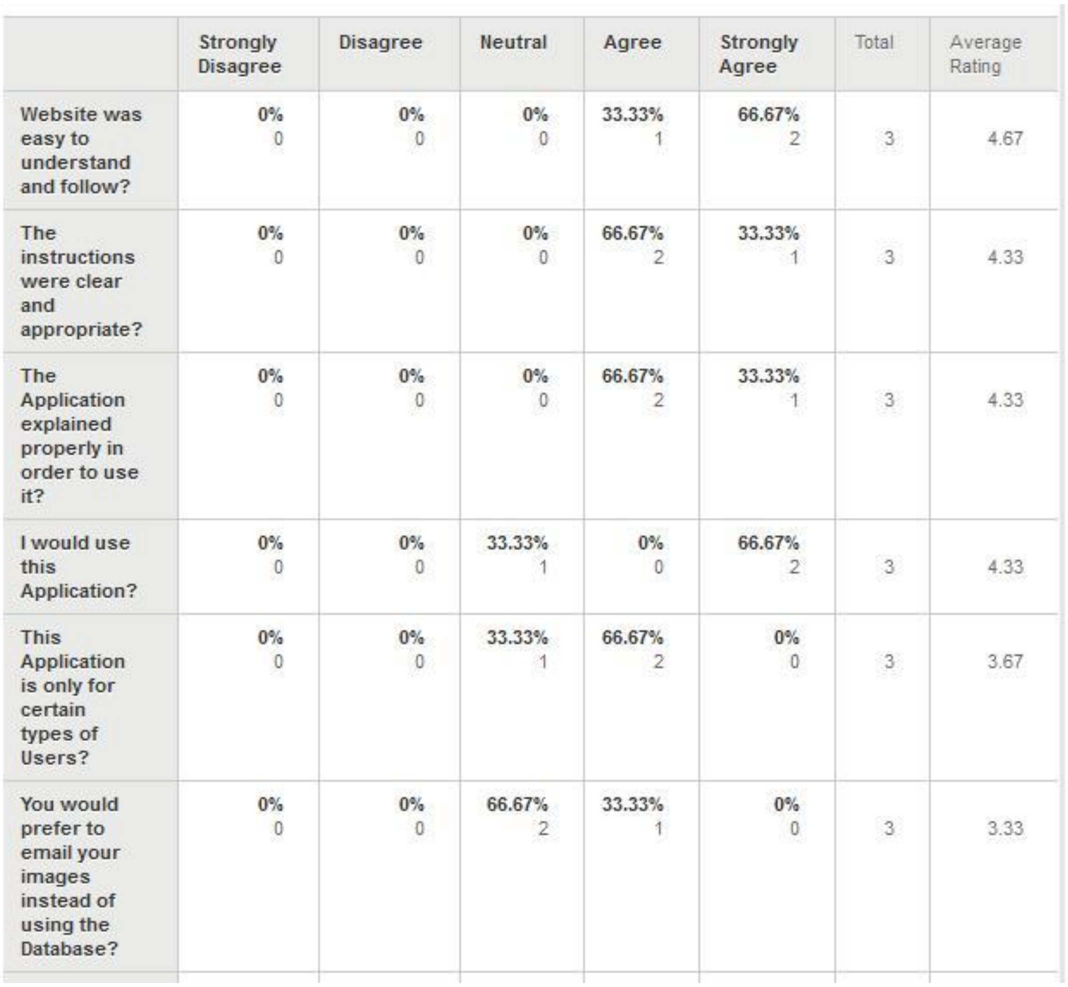
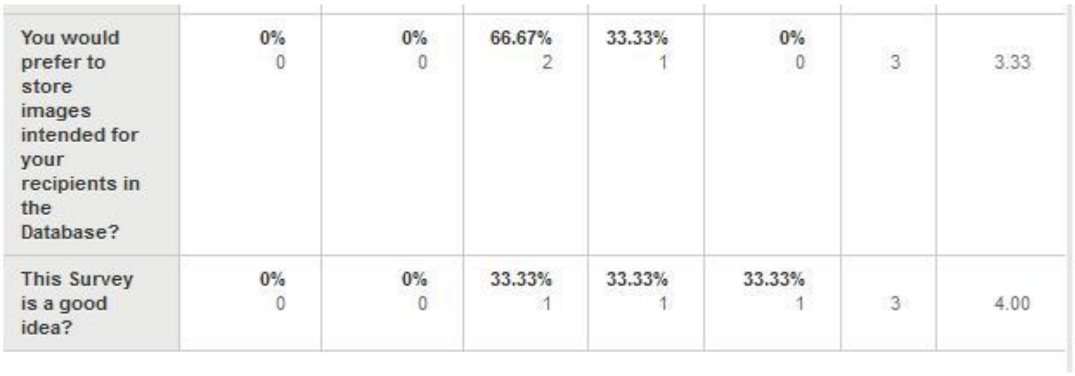
Below is a table; the column on the right details the questions asked while the column on the left gives the reason why the question was asked.

A query on how the Website was designed.	Website was easy to understand and follow?
A specific reference to the ReadMe.doc attached to the website.	The instructions were clear and appropriate?
Again I am evaluating my instruction and users learned to use the application.	The Application explained properly in order to use it?
Trying to ascertain if a user would use this Application.	I would use this Application?
Determining what type of actual person would use this Application.	This Application is only for certain types of Users?
A reference to what features in the application a user would use.	You would prefer to email your images instead of using the Database?

5.1.2 Analysis of Results of Survey

The results of the survey were generally favourable. I received positive feedback for in instructing respective users how to use the application plus generally they would use the application. About using the database; the response was generally neutral.



	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Total	Average Rating
Website was easy to understand and follow?	0% 0	0% 0	0% 0	33.33% 1	66.67% 2	3	4.67
The instructions were clear and appropriate?	0% 0	0% 0	0% 0	66.67% 2	33.33% 1	3	4.33
The Application explained properly in order to use it?	0% 0	0% 0	0% 0	66.67% 2	33.33% 1	3	4.33
I would use this Application?	0% 0	0% 0	33.33% 1	0% 0	66.67% 2	3	4.33
This Application is only for certain types of Users?	0% 0	0% 0	33.33% 1	66.67% 2	0% 0	3	3.67
You would prefer to email your images instead of using the Database?	0% 0	0% 0	66.67% 2	33.33% 1	0% 0	3	3.33
You would prefer to store images intended for your recipients in the Database?	0% 0	0% 0	66.67% 2	33.33% 1	0% 0	3	3.33
This Survey is a good idea?	0% 0	0% 0	33.33% 1	33.33% 1	33.33% 1	3	4.00

5.2 Unit testing of various Functions, Methods & Classes

For testing I decided to use the unit testing module in Visual Studio. I then proceeded to then check specific classes, functions and methods for testing. I have the commented each of the functions tested. The test project is part of the same solution in Visual studio which houses the whole solution. The name of the test project is called StegAppTest.

EmbedClassTest.cs, EmbedTest.

Within the above test file, EmbedTest unit tests the functionality of the method embed. This method embeds the message inside an image. For this test I gave predetermined values to EmbedKey, EmbedMessage and image3- the bitmap into which the message is supposed to be embedded. I then run this function and proceed to run various assert statements to test whether certain variables are within certain ranges and are of a certain type. The code coverage results achieved are 86.49%.

RetrieveMessageClassTest.cs, RetrieveTest.

This function tests the Retrieve method of RetrieveMessageClass.cs. The function to be tested is responsible for retrieving a message from an image. Again I set the key and the image to be used. I then test the range and type of a number of variables concerned. Code coverage is 94.29 %.

StegAppTest.cs, RetrieveKeyFnTest1

This test tests the RetrieveKeyFn in StegApp.cs which is responsible for retrieving the last log of the validated user from the database. I set the username and password and test the function which achieves 100% code coverage. I test also that the values entered are of a certain type.

StegAppTest.cs, SelectfilefnTest

The Selectfilefn allows a user to select a file from their hard drive. This test achieved 100% coverage. I test that the selection is not null and the image is of type Bitmap.

LoginRegisterTest.cs,LoginfnTest1.

This test works on the function Loginfn which is the function that handles the initial login event. 100% code coverage achieved.

RegisterTest.cs, RegisterFnTest1

In this test we test the code which is executed for the ‘Register’ button on the Register form. This is executed by the function RegisterFn. The unit test achieves 100% code coverage plus it passes the various assert statements to test the type of the variables.

StegAppTest.cs, SaveImageFnTest2

This is the code which tests the event to save an image with an embedded message in the Database. 100% code coverage achieved.

StegAppTest.cs, SaveImageAndKeyFnTest2

As the title of the function suggests we are testing the event to save both the image and the key on the database. I replicated the username, password, key and image and achieved 100% coverage in the results.

LogTest.cs, UserIdTest.

This test is for one of the Business objects classes: in this case log.cs. Within this class I am testing:

```
                        public int? UserId { get; set; }
```

which is a constructor. I use this class to hold a log object from the database and also to store information which will be stored in the database as a log object/row in the log table. 100% code coverage was achieved.

DataAccessTest.cs, AuthenicateCredentialsTest

This is the first of three unit tests dealing with DataAccess.cs; the class which deals with direct interaction with the database. Inserting, retrieving and validating data using log, User and Picture.cs to store and hold the data. ‘AuthenicateCredentialsTest’ validates the input data against actual data. In this case I input valid data and run the test which achieves 100% code coverage.

DataAccessTest.cs, RetrieveMessagesTest,

This function is responsible for the retrieval of logs from the database. For this test I input the expected logId to be returned for a specific user Id. The test passed with 100% coverage.

DataAccessTest.cs, InsertLogTest

This test will inadvertently test another function GetUser(). I call this function with valid data plus I declare a valid end user id and message key. 100% code coverage achieved.

6 Discussions

There were many challenges and decisions along the way of this project. The first decision along the way was what language or technologies to use for this project. I chose C# because it is a language I enjoy and familiar with plus I was confident and familiar with development package Visual Studio 2010.

The next decision to make was how this application to be developed. At the time the technology I wanted to go with was Windows forms as it was a technology we were starting to get familiar with and trained in at the time in our object oriented programming lectures. The only other technology I considered at the time was web based forms but time was of the essence; I knew forms and I needed to research possible algorithms for Steganography and how to integrate a Database into the application. Also forms are easier to create according to advice I received at the time. Thankfully there were no major challenges in using forms.

I had to research to ascertain which would be the best algorithm to implement embedding of a message inside an image. After much thought I chose to do the Least Significant Bit algorithm as I felt it was an approach I could accomplish. The main challenge faced in this part of the code was the retrieval of the embedded message from an image. The original message was converted into a binary string and once this was retrieved from the image, the challenge was to turn it back into a string; as previously described this was thankfully achieved. I had to do a lot of research into image processing and how .net processed images either generally or Bitmaps through the Bitmap Class.

Recent attempts showed my application to work for jpeg files and sometimes it did not, so I cannot verify other image formats without further research.

The next challenge to face was the integration of a Database into the application. Originally I had toyed with the idea of simply having a local database bundled with the application to keep a record of the user send and received. I then decided to go with a single database hosted online where a user could store his/hers logs of usage of the application as well as the facility to store and receive images via this database. I experimented with a number of approaches to implementing a database. I eventually decided on using the entity framework. It seems to be one of the more popular Database approaches in the .net environment plus the local class it created('DatabaseEntities' to mimic the Database) provided a lot functionality for connecting c# programs to such a Database. Allot of research had to go into this part as both linking a database to an application and Entity Framework was totally new to me.

A Strange quirk I had to deal with was the fact that the connection string to connect to the database was visible in the config file which is bundled with the .exe file. I got around this removing the connection string from the config and hard coding it directly into the application in DataAccess.cs.

Testing and evaluation presented some challenges. The main challenge arose in unit testing the functions which interacted directly or indirectly with the Database. When I created the test project for the whole project for some inexplicable reason it did not create a copy of the app.config file which is specifically needed to connect to the Database in any situation, test or otherwise. When a copy of app.config was inserted into the test project, the unit tests decided to perform as expected without any errors.

Regrettably the concept of MVC was only something I realised late in this Project and in the course itself. It means dividing the code into three general layers. Firstly we have the User Interface which will either display/output data or retrieve/input data. The Business logic will make computations on the input/output data while finally the Data layer encompasses code which will read or write to a Database. And in fact a lot of my code actually implements this structure. StegApp.cs however which is itself the main form will actually read or write to the Database via DataAccess.cs directly without going through an intermediary class. Both the events that trigger such courses of action are implemented in a function which is called from the function of the event. In fact it is very difficult to unit test properly unless the code disassociated with the actual event trigger. EmbedClass and RetrieveMessages.cs are good examples of Business logic functions/Classes in a MVC structure. They take or put data to or form the User Interface after doing some computation on that data.

Cryptography was something I would have liked to introduce as an add-on at the latter stage of this project. Unfortunately I did not this finish area of research. I did though write a Class: Crypto.cs which would have taken the string or binary string of the message with or without the key and length and encrypted it. Then this encrypted message would have been embedded in the image. I was planned to use the actual key as the first key with the second key static. Two keys would have been needed as I was planning on using the AES encryption algorithm.

7. Conclusion and Future Work

When I initially set out on this project I had a few ideas of what I wanted to do and what I wanted to achieve. My initial goals were to have an interesting project with user interface and a Database and a bit coding which would provide a challenge. I felt I achieved all these goals in this project.

As regards future work there are many possible avenues of approach. My first add on to this project would be cryptography. I did manage to do some research on this area with some classes written but regrettably I was unable to implement it in the application. Another area linked to this is to research different Algorithms which implement image Steganography entering the realm of Digital Signal Processing.

My next sub project for this application would be the transition of the project to some sort of web service. Basically the application would become some sort of a asp.net mvc4 or .Net WCF Rest Service where the actual business logic or processing would be carried out on the server. The user might still have to install a application,.exe or dll file but this would only perform User interface logic; select image, key and message and this data would be sent to the running process on the server. This approach would also have the added advantage that the actual code which implements the embedding and retrieval of a message would be hidden form the user, stored on the server impeding attempts to break it. Currently the application as it stands which is a downloadable application file could be reversed engineered.

And finally I of course there is the realm of mobile applications. I did some preliminary research into building a mobile app written in Lua using the Corona API. Should be possible but I noted that I initially did not find a class in Corona which implement something similar to the Bitmap or String builder class in .Net. Of course I could be totally incorrect as I my research was not very thorough here so maybe the Api does have classes which have similar functionality.

Bibliography

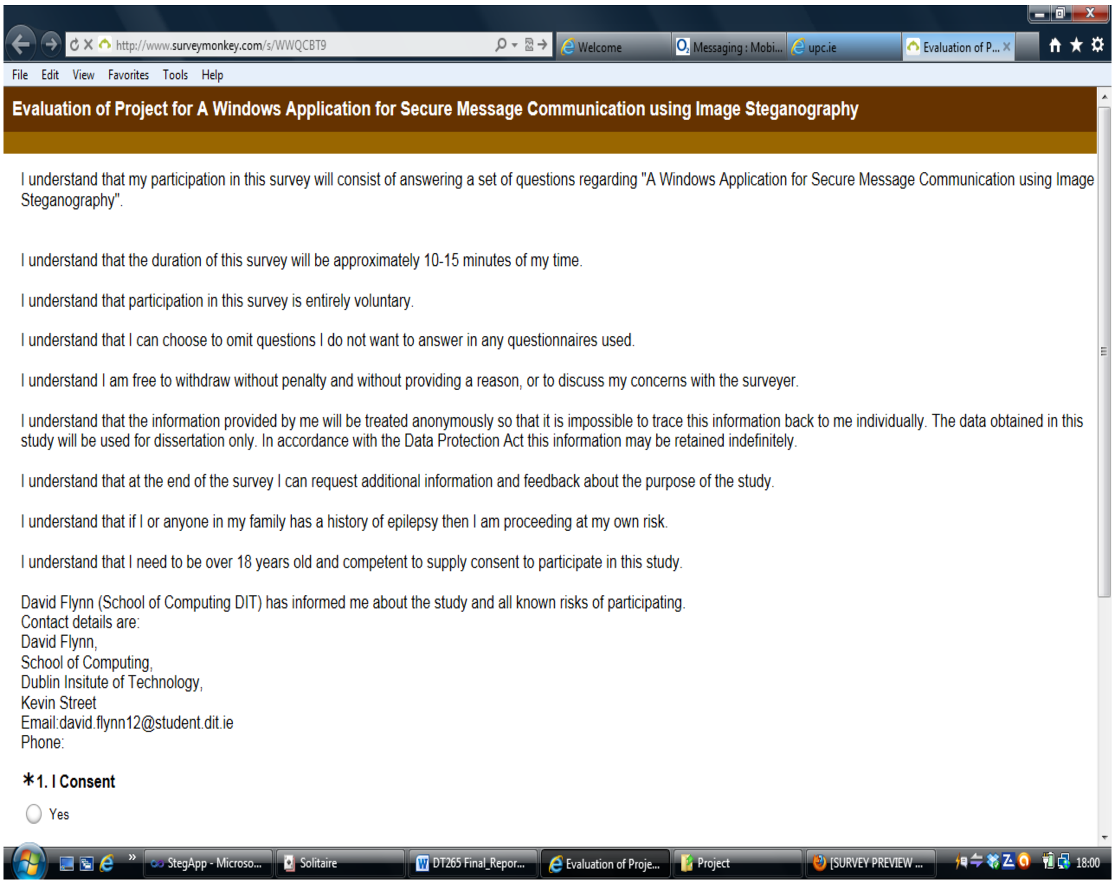
[1] *Secret Warfare: The Battle of Codes and Ciphers* by Bruce Norman.

[2] www.guillermito2.net/stegano/index.html. This outlines a detailed analysis of the software packages presented in the relevant table.

Appendices

Appendix 1 Survey

The following images show the survey followed by the results obtained, courtesy of surveymonkey.com.



Evaluation of Project for A Windows Application for Secure Message Communication using Image Steganography

I understand that my participation in this survey will consist of answering a set of questions regarding "A Windows Application for Secure Message Communication using Image Steganography".

I understand that the duration of this survey will be approximately 10-15 minutes of my time.

I understand that participation in this survey is entirely voluntary.

I understand that I can choose to omit questions I do not want to answer in any questionnaires used.

I understand I am free to withdraw without penalty and without providing a reason, or to discuss my concerns with the surveyer.

I understand that the information provided by me will be treated anonymously so that it is impossible to trace this information back to me individually. The data obtained in this study will be used for dissertation only. In accordance with the Data Protection Act this information may be retained indefinitely.

I understand that at the end of the survey I can request additional information and feedback about the purpose of the study.

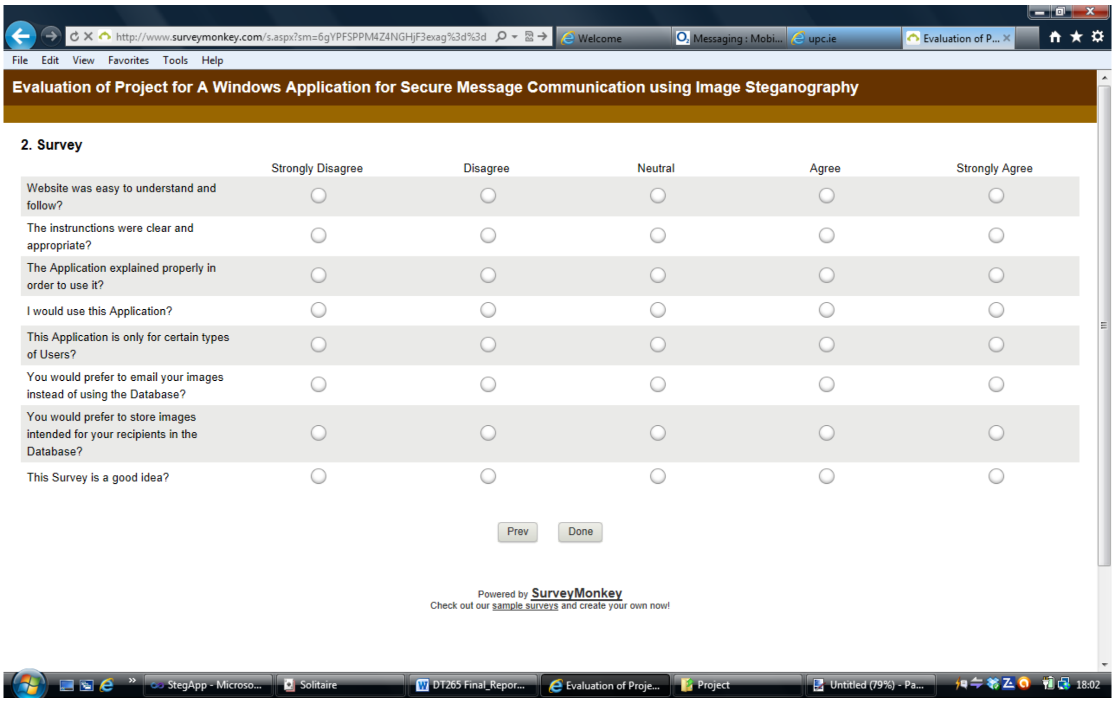
I understand that if I or anyone in my family has a history of epilepsy then I am proceeding at my own risk.

I understand that I need to be over 18 years old and competent to supply consent to participate in this study.

David Flynn (School of Computing DIT) has informed me about the study and all known risks of participating.
Contact details are:
David Flynn,
School of Computing,
Dublin Insitute of Technology,
Kevin Street
Email:david.flynn12@student.dit.ie
Phone:

*1. I Consent

○ Yes



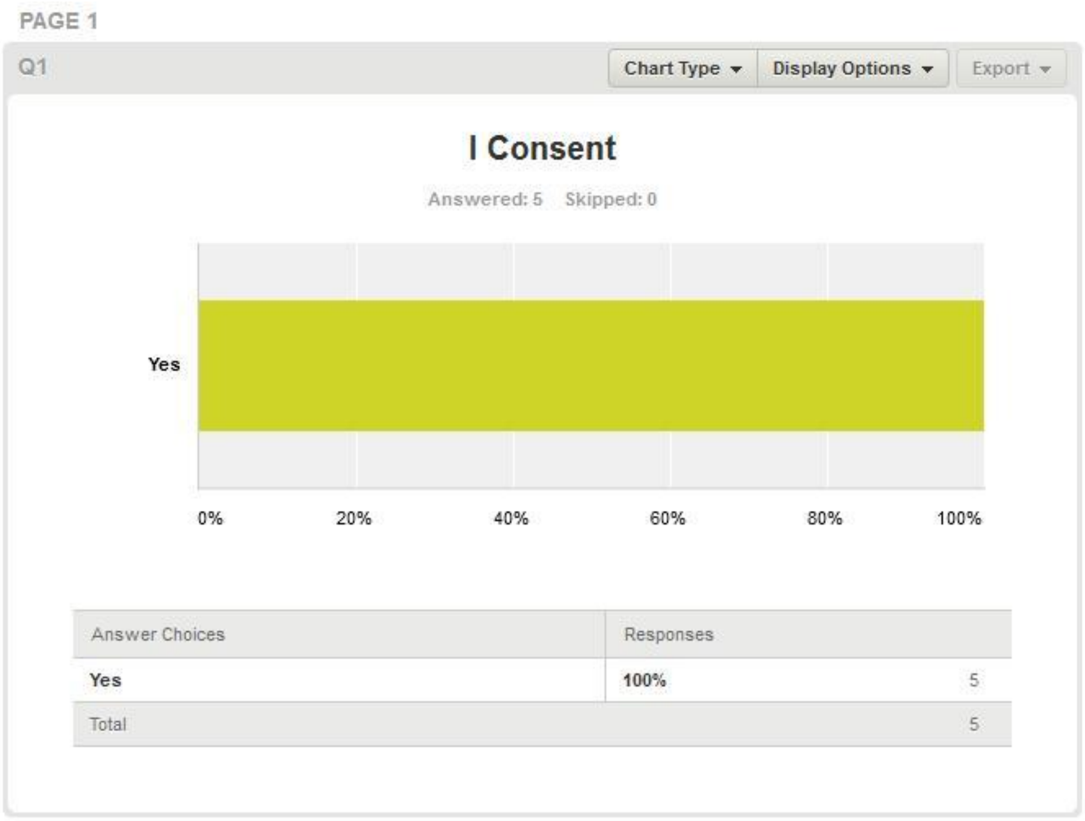
Evaluation of Project for A Windows Application for Secure Message Communication using Image Steganography

2. Survey

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Website was easy to understand and follow?	○	○	○	○	○
The instrunctions were clear and appropriate?	○	○	○	○	○
The Application explained properly in order to use it?	○	○	○	○	○
I would use this Application?	○	○	○	○	○
This Application is only for certain types of Users?	○	○	○	○	○
You would prefer to email your images instead of using the Database?	○	○	○	○	○
You would prefer to store images intended for your recipients in the Database?	○	○	○	○	○
This Survey is a good idea?	○	○	○	○	○

Prev Done

Powered by SurveyMonkey
Check out our sample surveys and create your own now!



I Consent

Answered: 5 Skipped: 0

Answer Choices	Responses	
Yes	100%	5
Total		5