Fort-Knox Encryption

MAY 20

Sekwati Thulare – 28124227
Banele Ndaba – 35482737
Given Mnisi – 34292748
Tshimbiluni Nedambale – 33574359
Njabulo Madonsela - 34729100

FK

Password Hash

We began by creating a class to make the hash function public and accessible to different forms. The hash password in its class returns a string. To access some functions of Cryptography we added the heading “using System.Security.Cryptography”. To generate the hash value of 20-byte, we utilized the Cryptography formatted hash function known as the SHA (Secure Hash Algorithm) algorithm. We then used a byte array to store our ASCII formatted password. On the completion of our password hash class and method we parsed it to the textboxes which deliver data/password hashes to the database.

Self-created encryption algorithm

The algorithm works using the following 3 steps:

1. It swaps the first letter in the string with the last letter ***(e.g. "James" ≫ "sameJ" and "Hello world" ≫ "dello worlH")***
2. It reverses then new string created in step 1 ***(e.g. "sameJ" ≫ "Jemas" and "dello worlH" ≫ "Hlrow olled")***
3. It takes the reversed string from step 2 and moves each character in the string 3 spaces up on the ASCII table. ***(e.g. "Jemas" ≫ "Mhpdv" and "Hlrow olled" ≫ "Kourz#roohg")***

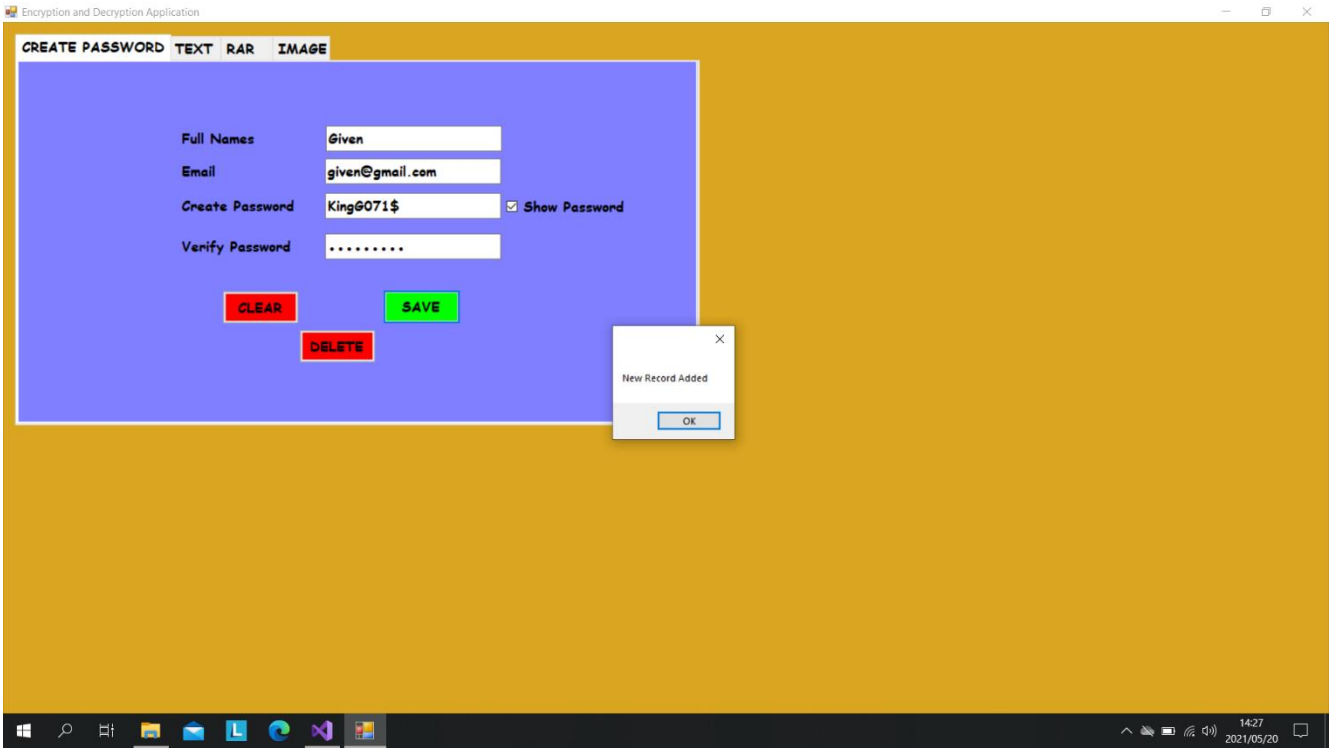
To decrypt the text the algorithm is reversed meaning step 3 of the encryption proses is step 1 of the decryption proses and so forth. We created a method for each of the steps and there are relevant comments in the code explaining how each of the steps are completed.

Extract of the program code.

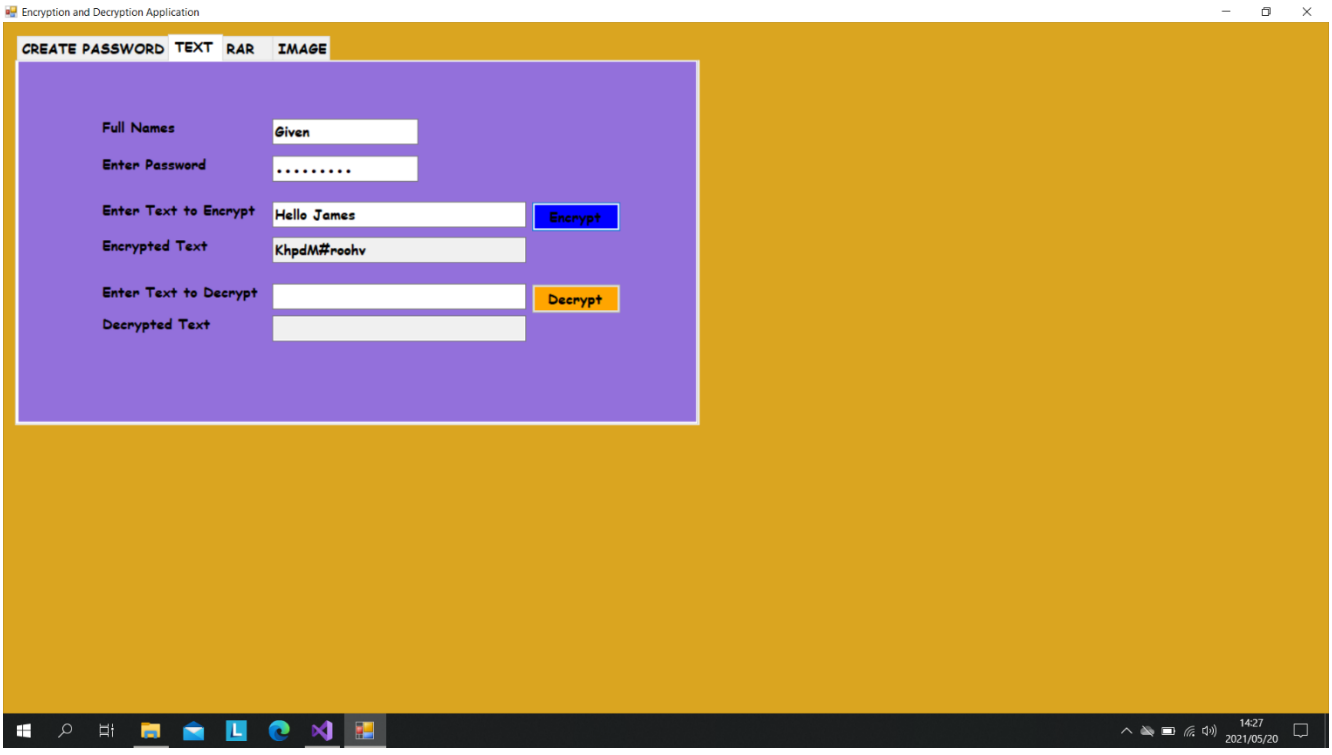
```
private static string Swap(String str) // Method that completes Step 1: Swap the first letter
with the last letter in the string and store it in newString.   || e.g. James becomes sameJ
        {
            //Variables
            int ifirstPosition, ilastPosition, Temp;
            string newStr, smidparts, sfirstPosition, slastPosition;
            ifirstPosition = 0; //Index of first Letter
            ilastPosition = str.Length - 1;  //Index of last Letter

            smidparts = str.Substring(1, ilastPosition - 1);

            //Swapping first and last letter
            Temp = ifirstPosition;
            ifirstPosition = ilastPosition;
            ilastPosition = Temp;

            //Getting first and last letter form the string
            sfirstPosition = str.Substring(ifirstPosition, 1);
            slastPosition = str.Substring(ilastPosition, 1);

            //The New String
            newStr = sfirstPosition + smidparts + slastPosition;

            return newStr;
        }

        private static string ReverseString(String input) // Method that completes Step2:
        Reverse the newString.    || e.g. sameJ becomes Jemas
        {
            string ReversedStr = "";

            for (int i = input.Length - 1; i >= 0; i--) //for to reverses string
            {
                ReversedStr += input[i];
            }
```

```
    return ReversedStr;
}

private static string EncrytedStr(string str) // Method that completes Step3: Move each
character in the newString 3 spaces up on the ASCII Table.  || e.g. Jemas becomes Mhpdv
{
      char[] arr = str.ToCharArray(); //Array that converts newString from step 2 into
      characters

    //Variables
    string newStr = ""; //Empty string initializes variable
    char CharInArray, newCharInArray;
    int ASCIIvalue, newASCIIvalue;

    for (int i = 0; i < str.Length; i++) //For loop for moving characters 3 spaces up
    on the ASCII Table and turning array into one string.
    {
        CharInArray = arr[i];

        ASCIIvalue = (char)CharInArray;

        newASCIIvalue = ASCIIvalue + 3; // Moving Characters

        newCharInArray = (char)newASCIIvalue;

        newStr += newCharInArray; // Turning array new characters into one string.
    }

    return newStr; // returns the new string.
}

private static string DecrytedStr(string str)
{
    char[] arr = str.ToCharArray();

    //Variables
    string newStr = ""; //Empty string initializes variable
    char CharInArray, newCharInArray;
    int ASCIIvalue, newASCIIvalue;

    for (int i = 0; i < str.Length; i++) //For loop for moving characters 3 spaces down
    on the ASCII Table and turning array into one string.
    {
        CharInArray = arr[i];

        ASCIIvalue = (char)CharInArray;

        newASCIIvalue = ASCIIvalue - 3; // Moving Characters

        newCharInArray = (char)newASCIIvalue;

        newStr += newCharInArray; // Turning array new characters into one string.
    }

    return newStr; // returns the new string.
}
```

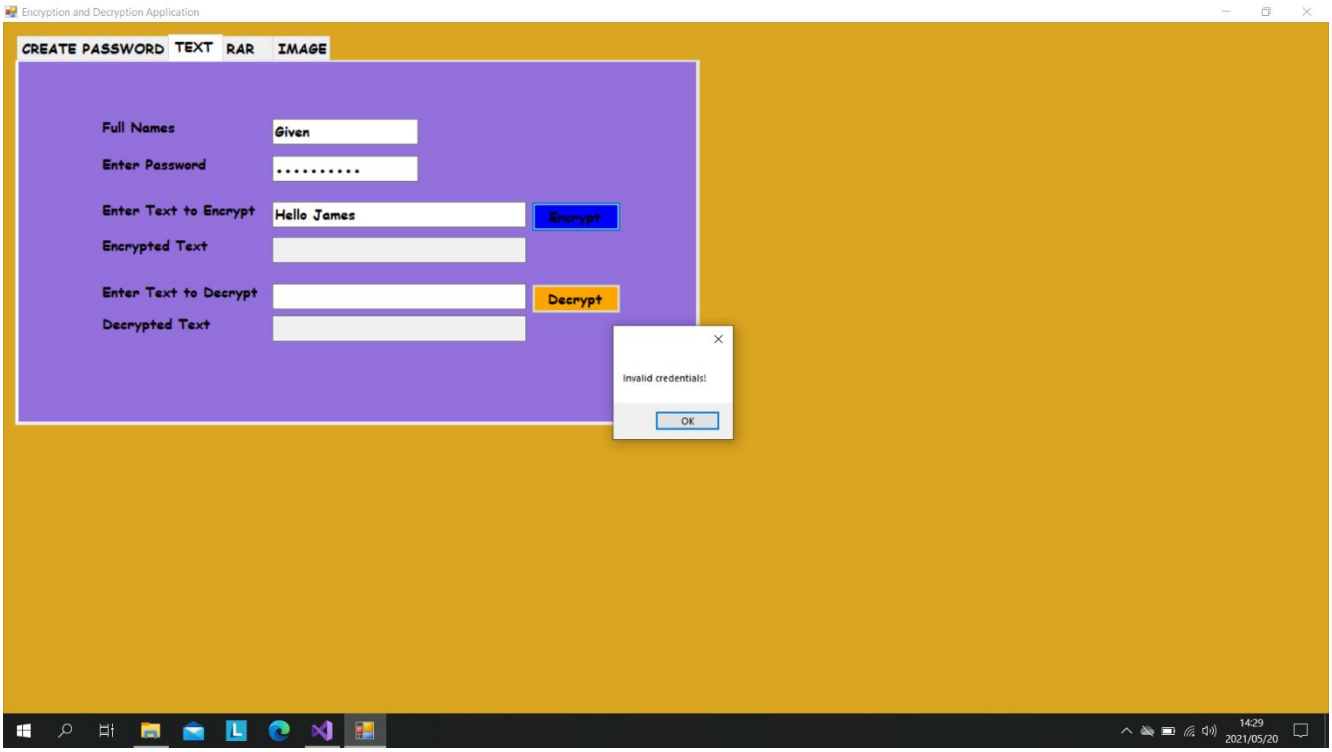
This algorithm only works for text encryption and decryption only.



The user must enter his/her name, email, and create a password. The hash of that password will be then store in a database alongside the other information.



For encrypting text the user must enter his name and password and the text they want to encrypt.



The program will only encrypt or decrypt the text if the entered password matches the one stored in the database.

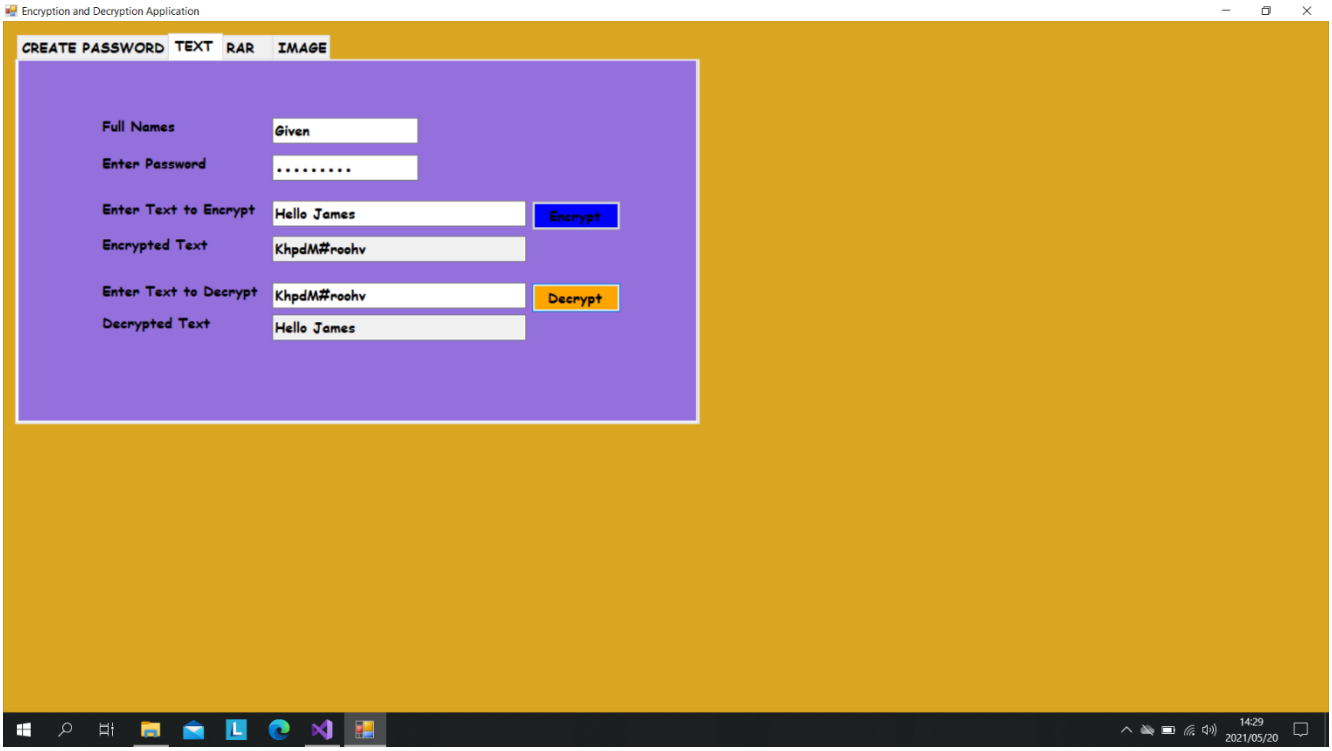


IMAGE & RAR FILE ENCRYPTION

METHODS

```
//This code is for encrypting the Image & Rar file
private void encrypt(string input, string output, string hash)
        {
            FileStream inStream, outStream;
            CryptoStream cryStream;

      //this code is for declaring the encryption from the cryptographer
            TripleDESCryptoServiceProvider tdc = new
TripleDESCryptoServiceProvider();
            MD5CryptoServiceProvider md5 = new MD5CryptoServiceProvider();

            byte[] byteHash, byteTexto;

      //this code is for uploading the file to the program for encryption
            inStream = new FileStream(input, FileMode.Open,
FileAccess.Read);
            outStream = new FileStream(output, FileMode.OpenOrCreate,
FileAccess.Write);

      //this code is for calling the method for encrypting and encrypting
the image file uploaded
            byteHash = md5.ComputeHash(ASCIIEncoding.ASCII.GetBytes(hash));
            byteTexto = File.ReadAllBytes(input);

            md5.Clear();

            tdc.Key = byteHash;
            tdc.Mode = CipherMode.ECB;

            cryStream = new CryptoStream(outStream, tdc.CreateEncryptor(),
CryptoStreamMode.Write);

            int byteRead;
```

```
            long length, position = 0;
            length = inStream.Length;
      //this code for shifting the letters of the name of the image file
            while (position < length)
            {
                byteRead = inStream.Read(byteTexto, 0, byteTexto.Length);
                position += byteRead;

                cryStream.Write(byteTexto, 0, byteRead);
            }

            inStream.Close();
            outStream.Close();
        }

      //This code is for decrypting the Image file
        private void decrypt(string input, string output, string hash)
        {
            FileStream inStream, outStream;
            CryptoStream cryStream;

      //this code is for declaring the encryption from the cryptographer
            TripleDESCryptoServiceProvider tdc = new
TripleDESCryptoServiceProvider();
            MD5CryptoServiceProvider md5 = new MD5CryptoServiceProvider();

            byte[] byteHash, byteTexto;

      //this code is for uploading the file to the program for encryption
            inStream = new FileStream(input, FileMode.Open,
FileAccess.Read);
            outStream = new FileStream(output, FileMode.OpenOrCreate,
FileAccess.Write);

      //this code is for calling the method for decrypting and decrypting
the image file uploaded
            byteHash = md5.ComputeHash(ASCIIEncoding.ASCII.GetBytes(hash));
            byteTexto = File.ReadAllBytes(input);

            md5.Clear();

            tdc.Key = byteHash;
            tdc.Mode = CipherMode.ECB;

            cryStream = new CryptoStream(outStream, tdc.CreateDecryptor(),
CryptoStreamMode.Write);
```

```
            int byteRead;
            long length, position = 0;
            length = inStream.Length;

     //this code for shifting the letters of the name of the image file
back to original name of the file
            while (position < length)
            {
                byteRead = inStream.Read(byteTexto, 0, byteTexto.Length);
                position += byteRead;

                cryStream.Write(byteTexto, 0, byteRead);
            }
            inStream.Close();
            outStream.Close();
        }
```

CODE FOR BUTTONS

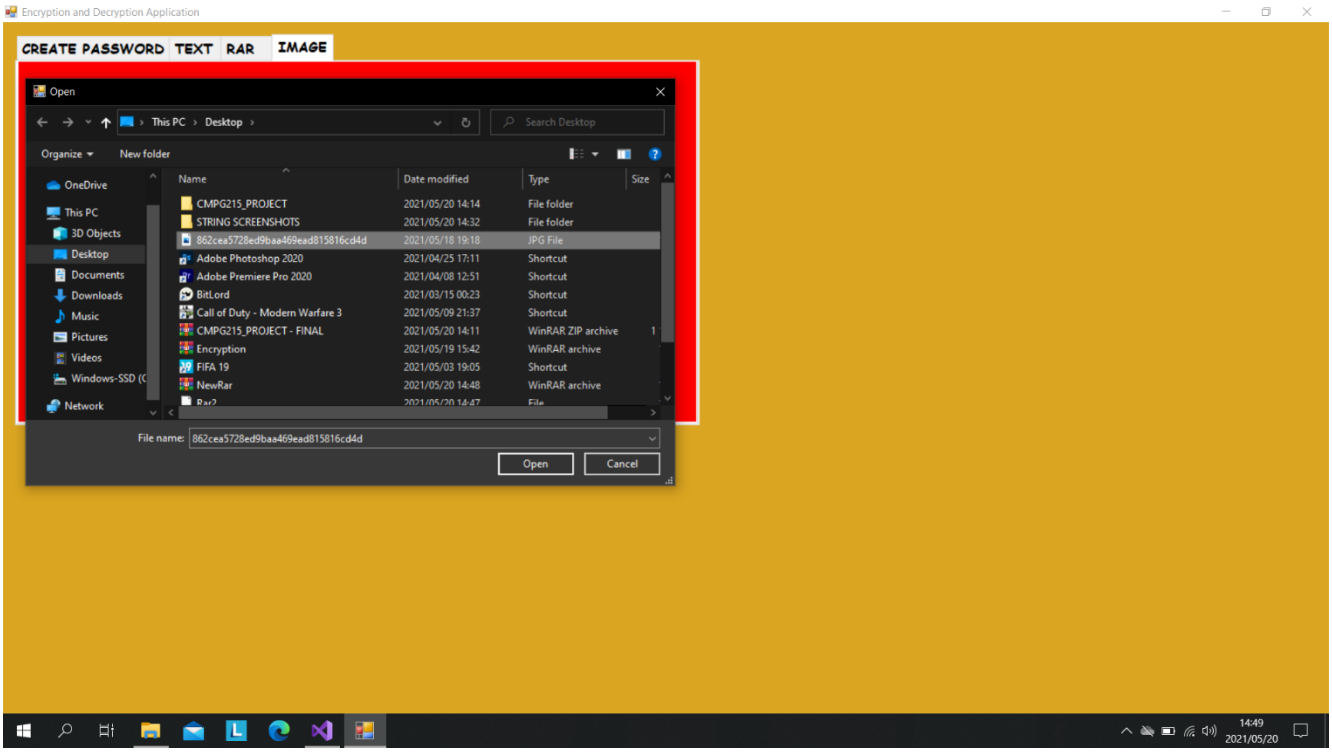
```
private void btnEncryptI_Click(object sender, EventArgs e)
        {
            string Entered_Password, Database_Password;
           //this is changing the password entered to hash password

            Entered_Password = Utilities.hashPassword(txtPasswordT.Text);
           //this code is for matching the name of the user to registered
user on the database
            string sql = "SELECT * FROM InfoTB WHERE Name Like'%" +
txtNameI.Text + "%'";
            Command = new SqlCommand(sql, Connection);
            Connection.Open();
           //this code if for verifying the password entered with the one
in the database
            using (SqlDataReader Reader = Command.ExecuteReader())
            {
                if (Reader.Read())
                {
                    Database_Password = Reader["Password"].ToString();

                    if (Database_Password == Entered_Password)
                    {
                //this code is uploading the image file to encryption
method
                        OpenFileDialog Open = new OpenFileDialog();
                        Open.ShowDialog();

                        txtImageE.Text = Open.FileName;
```

```
                    SaveFileDialog Save = new SaveFileDialog();
                    Save.ShowDialog();

                    txtEncryptedI.Text = Save.FileName;

                    encrypt(txtImageE.Text, txtEncryptedI.Text, key);
                    MessageBox.Show("Image Successfully Encrypted");
                }
                else
                {
                    MessageBox.Show("Passwords don't match");
                }
            }
        }
        Connection.Close();
    }

    private void btnDecryptI_Click(object sender, EventArgs e)
    {
         string Entered_Password, Database_Password;

       //this is changing the password entered to hash password
        Entered_Password = Utilities.hashPassword(txtPasswordT.Text);

       //this code is for matching the name of the user to registered
user on the database
        string sql = "SELECT * FROM InfoTB WHERE Name Like'%" +
txtNameI.Text + "%'";
       //this code if for verifying the password entered with the one
in the database
        Command = new SqlCommand(sql, Connection);
        Connection.Open();

        using (SqlDataReader Reader = Command.ExecuteReader())
        {
            if (Reader.Read())
            {
                Database_Password = Reader["Password"].ToString();

                if (Database_Password == Entered_Password)
                {
             //this code is uploading the image file to decryption
method
                    OpenFileDialog Open = new OpenFileDialog();
                    Open.ShowDialog();
```

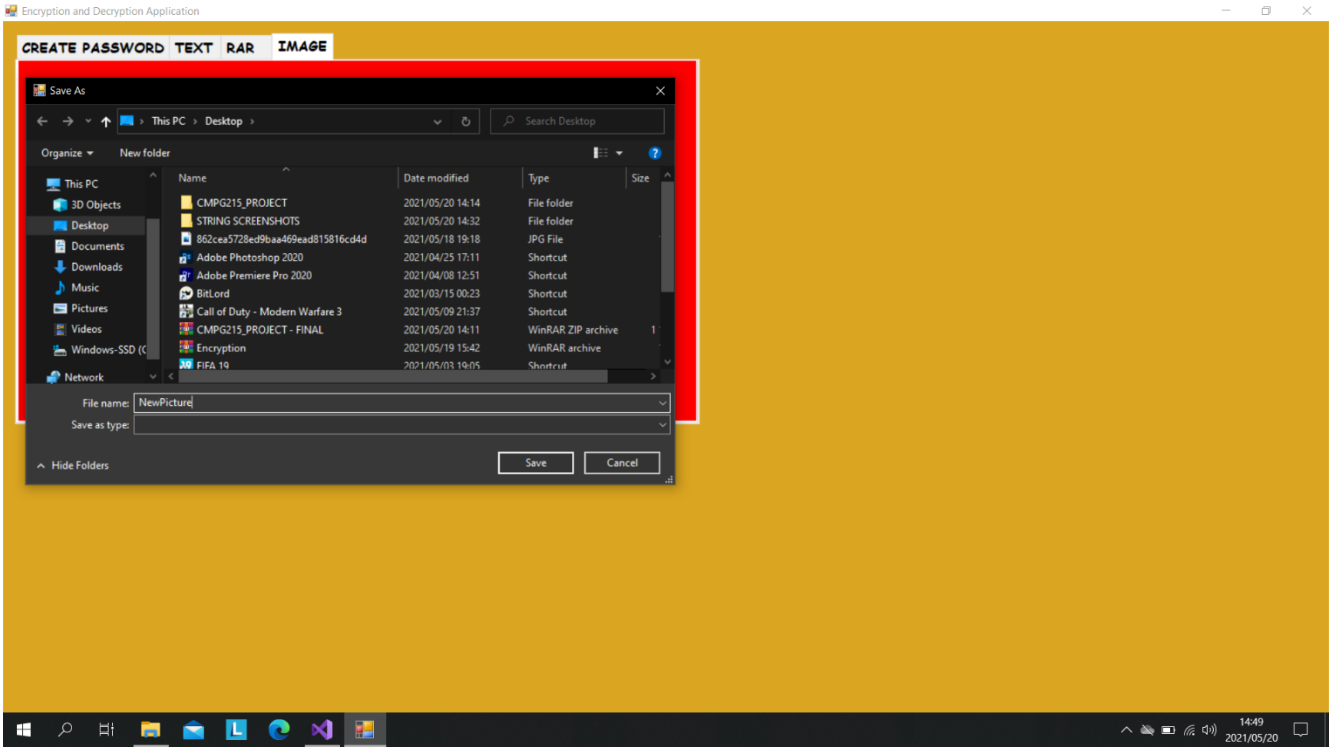
```
                    txtImageD.Text = Open.FileName;

                    SaveFileDialog Save = new SaveFileDialog();
                    Save.ShowDialog();

                    txtDecryptedI.Text = Save.FileName;

                    decrypt(txtImageD.Text, txtDecryptedI.Text, key);
                    MessageBox.Show("Image Successfully Decrypted");
                }
                else
                {
                    MessageBox.Show("Passwords don't match");
                }
            }
        }
        Connection.Close();
    }
```

NOTE!!

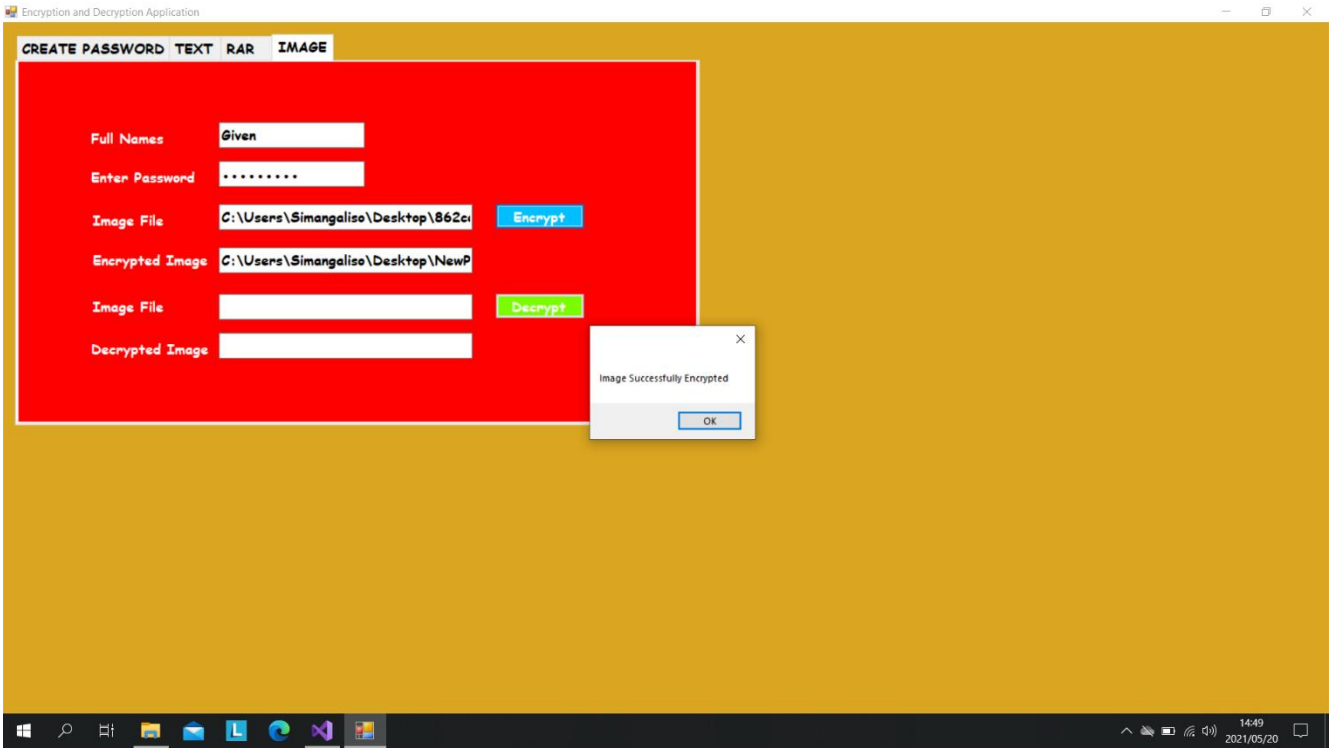
Buttons for RAR FILE work similary as the one for image.



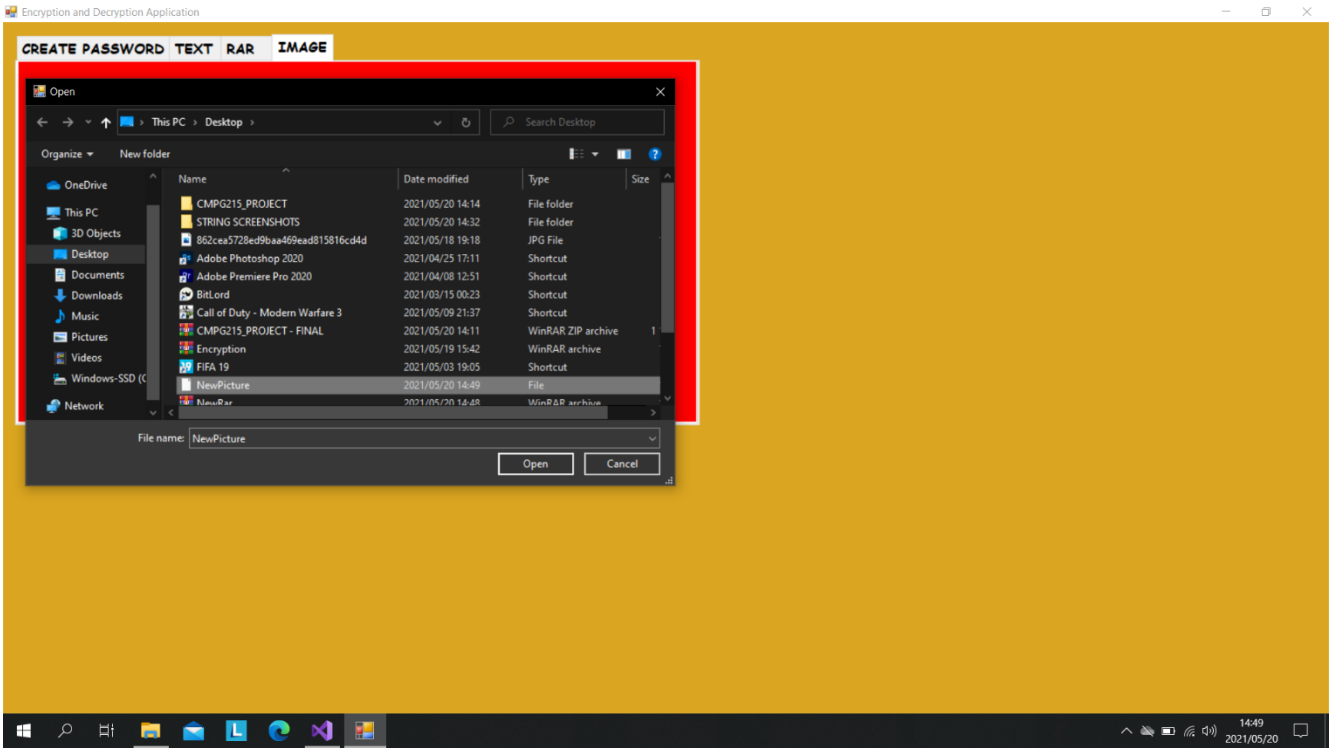
After pressing encrypt the program will allow you to select the file you want to encrypt, click on it then press open.



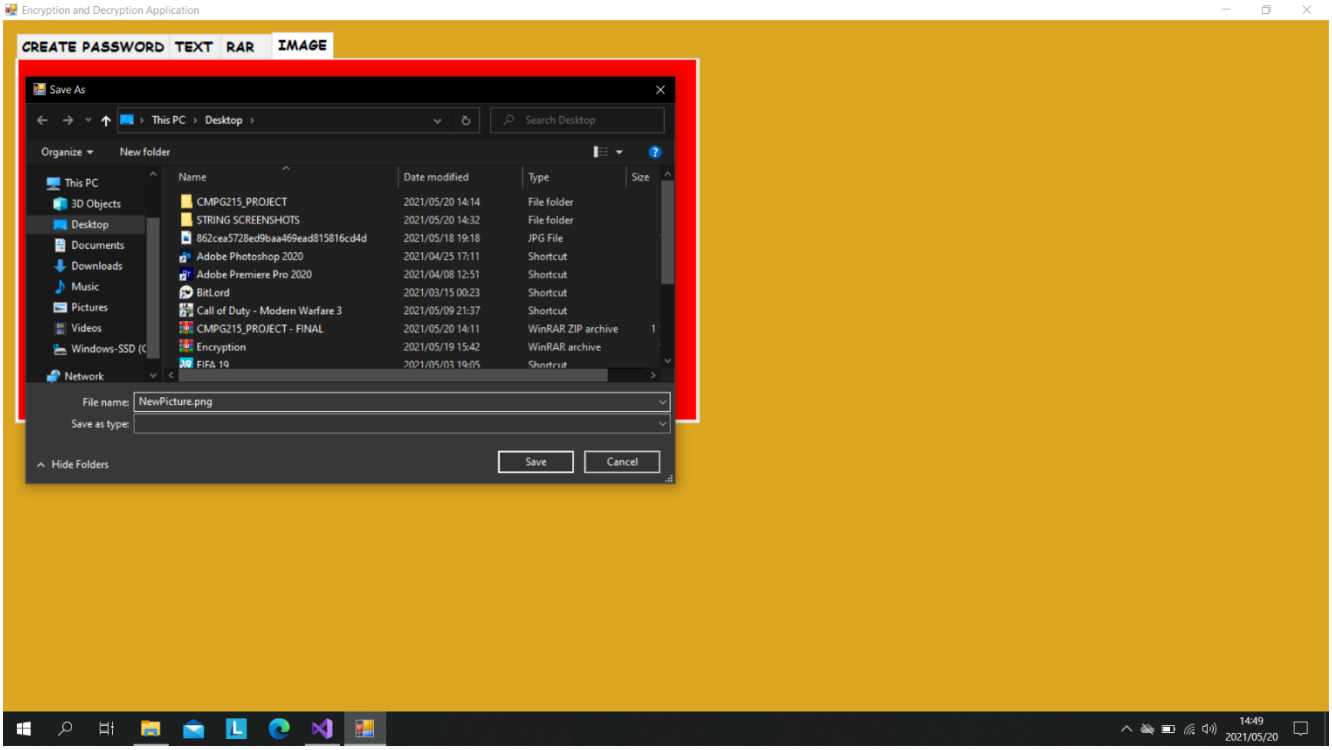
The user must input a name that will be used to save the encrypted file as.



Similar to the text encryption the program will only encrypt the file if the entered credentials match the ones in the database.



After pressing button decrypt the program will allow you to choose the decrypted folder.



You will now be required to enter the name that you want to save the decrypted file as.

Not: The file extension must match the original file before encryption.