Technical Documentation - My Diary

Rahul R Huilgol, 11010156

Working Flow

On first load of the software, the user gets the Set Password form to set a password.

Set Password: The string typed is hashed and stored in “pass.p” file.

Lockscreen: If the user has already set a password, he goes directly to Lockscreen form. Else he comes here after he sets a password.

On entering the correct password, it is hashed, compared with the passwordhash stored in pass.p. Then the user enters the main form, Main (Diary.vb).

Main: This is where the user can see the entries that he has added in ListView. The items are shown here by scanning all .encrypted files in output folder. On clicking create button, a form opens up. The entries made here are stored in a .txt file initially on clicking save. This .txt file is immediately encrypted to a .encrypted file and the original .txt file is deleted. The ListView items are cleared and refreshed to add the file created. An item can be selected and delete button can be pressed to delete the entry corresponding to that item.

Edit: On double clicking an entry, the encrypted file is decrypted into a txt file. The contents of the textboxes are loaded from here. On clicking save, the old encrypted file is deleted. The edited contents are saved into a txt file. This file is then encrypted to create a .encrypted file and the txt file is deleted.

Create: On clicking create entry button an object of create form is created. After entering the strings, ands clicking save the details are stored into a .txt file. This file is then encrypted to create a .encrypted file and the .txt file is deleted. If the user selects a date for which an entry already exists, a new entry will not be created.

Change: This facilitates the user to change the password if need be. The current password entry is hashed and compared to passwordhash in pass.p. If it matches then the new password is hashed and this is stored in pass.p

Individual Classes

Lock screen:

- Variables:
  - Strings – passwordhash, FILE_NAME
- Form Elements
  - Label1 = “Enter Your Password”
  - TextBox1 for entering password
  - Label2 = “Select a password to get started”
  - Button1 =”Unlock”
  - Button2 = “Click here to set Password”
- Functions
  - Button2_Click (SetPassword)
    - Creates an object of SetPassword and shows it
  - Lockscreen_Load
    - If FILE_NAME exists, then read from file and assign it to passwordHash, set visible unlock button
    - If output folder doesn’t exist, create it. Set visible SetPassword button
    - Delete all .encrypted files if any
  - Button1_Click_1 (Unlock)
    - TextBox1.text is hashed and compared with the passwordhash stored in “pass.p”. Please refer to hashing method at the end of the file.
    - If it matches Main Form is shown
    - Else enter correct password

SetPassword:

- Variables:
  - Strings – passwordhash, FILE_NAME
- Form Elements
  - Label1 = “Set Password”
  - Label2 = “Enter password”
  - TextBox1 for entering password
  - Button2 = “set Password”
- Functions
  - Button2_Click (SetPassword)
    - Hashes the textbox1.text using the method at the end of the file and writes it into FILE_NAME
    - Shows lockscreen by calling an object
  - SetPassword_Load

        - If output folder doesn't exist, create it. Set visible SetPassword button
    - frmProgramma_FormClosing
        - Dispose Lockscreen which is the parent of all. So disposing that will close all forms

File_Crypt

- Functions
    - EncryptFile
        - EncryptedFile Stream fsEncrypted is created
        - fsInput is the input file stream
        - DESCryptoServiceProvider() is created to help us use DES algorithm
        - DES.Key takes the encoded value(ASCIIEncoding) of the bytes of the key given
        - The vector required for DES algorithm is initialized

```
DES.IV  = ASCIIEncoding.ASCII.GetBytes(sKey)
```

        - Then the sentence is converted into a sequence using DES Encryption. The text file is read into the bytearrayinput
        - Write the encrypted data to file
        - Close the input output streams
        - Delete the text file which is given as input
    - DecryptFile
        - It requires a 64-bit key and IV for this provider. Set the secret key for the DES algorithm and the initialization vector
        - Create the file stream back to read the encrypted file
        - Create descriptor DES from our instance of DES
        - Create sequences set to read and perform encryption
        - DES decryption transformation is performed on the incoming bytes and finally the output is written to the file
        - Close all streams

Edit

- Variables
    - Initial (.encrypted) is a string set from Diary.vb when Edit has been called
    - Final is the final name which is the .txt
    - Initialcut is the name of the entry without any extensions
- Form Elements
    - TextBox2 is box for Title
    - DateTimePicker2 for selecting date

    - TextBox3 for the contents entry
    - TimeBox for time
    - Save Button
- Functions
    - Button2_Click (On clicking save)
        - Delete the initial .encrypted file
        - On clicking save, all the TextBoxes are written to final file (.txt)
        - It is written in this format
            - Line1: Date
            - Line2: Time
            - Line3:Title
            - Line4 onwards: The contents of entry
        - Encrypt this .txt file using EncryptFile() function from File_Crypt
        - Clear items list
        - Reload Items list

Create

- Variables
    - sSecretKey which has the value “Password”. This has to be a 64 or 8 bits for the DES algorithm to work
    - FILE_NAME
    - isExist a boolean to determine if the entry of that day already exists
- Form Elements
    - TextBox2 is box for Title
    - DateTimePicker2 for selecting date
    - TextBox3 for the contents entry
    - Save Button
    - Cancel Button
- Functions
    - Button2_Click
        - Check if file with that name already exists. If yes the Error MessageBox
        - If not then continue
        - Write the contents of all boxes to the file FILE_NAME.
        - This is encrypted using EncryptFile(0 Function from File_Crypt
        - Clear items list
        - Reload Items list
    - Button3_Click
        - On clicking the cancel button, dispose present form

Diary

- Variables
    - Newfile – String
    - sSecretKey used to encrypt and decrypt files. sSecretKey = “password”
- Form Elements
    - MenuBar with 3 Menu Items – Main, Settings, Help
    - Menu contains Exit
    - Settings contains Change Password
    - Help contains About and link to ReadMe file
    - Button1 - Create Entry button
    - Button2 - Delete Entry button
    - ListView1 which lists all the entries by scanning for all .encrypted files in output folder
    - Label with welcome message
    - An Image
- Functions
    - Button1 – Create
        - Creates an object of class Create and shows it
    - frmProgramma_FormClosing
        - Confirm Message
        - If yes, delete all .txt files left in output folder if any, close the parent of all which is Lockscreen
        - Else, cancel dialog box
    - Main_Load
        - RefreshList is called
    - ListView1_MouseDoubleClick
        - Identify item, if it is not nothing (i.e. some item is selected), create an object of form Edit and show
        - Decrypt the file corresponding to the entry using DecryptFile from File_Crypt.vb
        - Read from the txt file and assign the values to textboxes from the textfile
    - Refresh File
        - Find all .encrypted files
        - For every .encrypted file add to List View1
            - Decrypt such file, read the title line and add that using itmx.SubItems.Add(reader.ReadLine())
            - Now delete the decrypted file
    - ChangePasswordToolStripMenuItem4_Click
        - Create object of changepassword and show

- AboutSoftwareToolStripMenuItem5_Click
  - Show details of software in MessageBox
- ReadMeToolStripMenuItem4_Click
  - Starts the process Notepad.exe to open the file ReadMe.txt
- Button2_Click (Delete)
  - Delete the .encrypted file corresponding to the item selected
- ExitToolStripMenuItem7_Click
  - Confirms and closes the parent of all forms, LockScreen

Change

- Variables**:**
  - Strings – passwordhash, FILE_NAME
- Form Elements
  - Label1 = “Enter Current Password”
  - Label2 = “Enter New password”
  - TextBox1 for entering current password
  - TextBox2 for entering new password
  - ChangeButton = “Change Password”
- Functions
  - ChangeButton_Click
    - First fetch the passwordhash stored in “pass.p”
    - Then hash the TextBox1.Text using the method below
    - If hash of TextBox1.Text is equal to passwordhash, then proceed. Else Show error in MessageBox
    - While proceeding, hash the TextBox2.Text (New password) and store it in “pass.p”
    - Close Input Output streams

Hashing Process

The passwords are hashed and saved in the file pass.p. For any handling related to passwords, this process is used.
First the string from `TextBox` is converted to a byte array. This is done by using UTF8encoding. More specifically, this is done by the function encoder.GetBytes(string).
Similarly the salt string is also converted to a byte array.
A new array with size the sum of salt string and password string is created. The “password” bytes are stored in it. Append “salt” bytes to the same array.
Compute the hash value for password and salt bytes by using md5hasher which is a MD5CryptoServiceProvider. This MD5 hashed data is converted to base64-encoded string.