



Dharmsinh Desai University, Nadiad

Faculty of Technology

Department of Computer Engineering

B. Tech. CE Semester – VI

Subject: (CE – 620) **Object Oriented Software Engineering**

Project Title: ***Pixby***

Submitted by:

Name: **Vyom Pathak**

Roll No: **CE099**

ID: **17CEUON038**

Guided By:

Prof. Prashant M. Jadav

Associate Professor, CE Dept.,
Dharmsinh Desai University



Dharmsinh Desai University, Nadiad

Faculty of Technology, Department of Computer Engineering

CERTIFICATE

This is to certify that Object Oriented Software Engineering's project entitled "Pixby" is the bonafied report of work carried out by

1) Pathak Vyom (17CEUON038)

Of Department of Computer Engineering, Semester VI, academic year 2019-20,
under our supervision and guidance.

Guide

Prof. Prashant M. Jadav

Associate Professor of
Department of Computer
Engineering, Dharmsinh Desai University,
Nadiad.

HOD

Dr. C. K. Bhensdadia

Head of the Department of
Department of Computer
Engineering, Dharmsinh Desai University,
Nadiad.

No.	Content	Page No.
1	Introduction	5
2	Software Requirement Specifications	6
3	Design Documents	12
4	Implementation Details	22
5	Functionality Prototype	23
6	Testing Details	35
7	Conclusion	39
8	Limitations and Future Extensions	40
9	Bibliography	41




Overview

This application is used for editing photos and saving them and downloading them. User can upload image and can edit the images and than download or save according to their sign in status.



Introduction

This application is used for editing photos. User can upload the photo to the system and edit according to their needs. The software provides functionalities like image RGB value modification, image styling, and image flipping along the axis, image rotation as well as applying some filters on the images like Sepia, Greyscaling, and Inverting etc. User can then save the edited image in his/her account on the server or can download the images. User can also see their saved images as well as re-edit them. User can also perform undo and redo operation on their images to remove or add the features. Admin can remove user and can add new filters.

Technology Used:

-  Windows Form Application to create application.
-  Microsoft Sql Server as database. [MSSQL]
-  UMLET for Designing Different UML Diagram.

Tools Used:

-  I used Github as a version control method and to manage the project.
-  I used Visual Studio for development of the code.

Software Requirement Specification

Pixby

FUNCTIONAL REQUIREMENTS:

→ Admin

R1. Manage User:

DESCRIPTION: Admin can view all users ,remove users according to his/her desire.

R1.1 Remove User:

INPUT: Admin selects the user.

OUTPUT: Selected user will be removed.

R1.2 View Users:

INPUT: Admin selects view option.

OUTPUT: User list will be shown

R2 Manage Filters:

DESCRIPTION: Admin can add and delete any filter he/she desires.

R2.1 Add Filter:

Input: Filter code.

Output: Confirmation Status Message will be shown.

R2.2 Remove Filter:

Input: Select Filter Code.

Output: Confirmation Status Message will be shown.

→ Client

R1. Manage Images:

Description: This functionality allows user to upload images on the editor using the local file manager. It also allows user to save the uploaded image in the server after all the processing has been completed. It also allows user to remove image from the editor. User can also download his/her images from his/her account.

R1.1 Upload Image:

INPUT: User selection from file manager for his/her desired image.

OUTPUT: Confirmation message.

PROCESS: The selected image will be validated i.e. checked if the image uploaded is of the correct format or not.

NEXT: The user will make his/her desired changes on the image.

EXCEPTION FLOW: If the selected image is not in correct format error will be thrown.

R1.2 Save Image:

INPUT: User selects the save option.

OUTPUT: Confirmation message.

PROCESS: The specified the new image will be validated.

EXCEPTION FLOW: If the image is not valid error will be thrown.

R1.3 Remove Image:

DESCRIPTION: User selected image will be removed.

INPUT: User selection.

OUTPUT: Confirmation message.

R1.4 Download Image:

DESCRIPTION: User selected image will be downloaded.

INPUT: User selection.

OUTPUT: Confirmation message.

EXCEPTION FLOW: If the file path is not valid error will be thrown.

R2. Manage Filters:

DESCRIPTION: This functionality will allow user to use various type of filter according to his/her desire. It includes: image cropping, brightness adjustment, apply different styles, image flipping, image rotation and image RGB value modification.

R2.1 Manage Crop Filter:

INPUT: User selects the area to be cropped.

OUTPUT: Image transformation will be shown.

R2.2 Manage Rotation Filter:

DESCRIPTION: User selects which type of rotation to perform and selects the value of rotation.

INPUT: User gives degree of rotation and type of rotation.

OUTPUT: Image transformation will be shown.

R2.3 Manage Flipping Filter:

DESCRIPTION: User will select which type of flip to perform horizontal or vertical.

INPUT: User selects type of flip.

OUTPUT: Image transformation will be shown.

R2.4 Manage Color Filter:

DESCRIPTION: User will select the appropriate color value for red, green and blue from range 0-255.

INPUT: User type of color and give the appropriate color level.

OUTPUT: Image transformation will be shown.

R2.5 Manage Brightness Filter:

INPUT: User selects appropriate level of brightness.

OUTPUT: Image transformation will be shown.

R2.6: Manage Style Filter:

DESCRIPTION: User can select from a range of different styles like greyscale,

negative effect and Arctic effect.

INPUT: User selects appropriate styling for the image.

OUTPUT: Image transformation will be shown.

R3. Error Handling:

DESCRIPTION: If during any process error occurs appropriate error message will be displayed. For upload image, saving image, download image or applying filter error handling is done.

R3.1 Upload Error:

DESCRIPTION: If any type of error occurs during upload user will be prompted with a dialog-box.

INPUT: User selects an image from his/her local machine.

OUTPUT: Message will be prompted to the user.

R3.2 Saving Error:

DESCRIPTION: If any kind of error occurs during the saving stage of an image, error will be shown to the user.

INPUT: User selects the save option.

OUTPUT: Message will be prompted to the user.

R3.3 Downloading Error:

DESCRIPTION: If any kind of error occurs during the downloading stage of an image, error will be shown to the user.

INPUT: User selects the download option.

OUTPUT: Message will be prompted to the user.

R4. Manage User:

DESCRIPTION: User can create new account , login to existing account and remove his/her account.

R4.1 Sign-Up User:

DESCRIPTION: User enters his/her details and will be validated from the database.

INPUT: User details.

OUTPUT: Confirmation Message.

R4.2 Login User:

DESCRIPTION: User will enter his/her details and will be saved in the database.

INPUT: User details.

OUTPUT: Confirmation Message.

R4.3 Remove User:

DESCRIPTION: User can delete his/her account from the application using the remove option.

INPUT: User selects the remove option.

OUTPUT: Confirmation Message.

NON FUNCTIONAL REQUIREMENTS:

SOFTWARE: Visual Studio, Github Desktop

OS: Windows

Design Documents

1. Use Case Diagram:



Figure 1 : Use Case Diagram

2. Class Diagram:

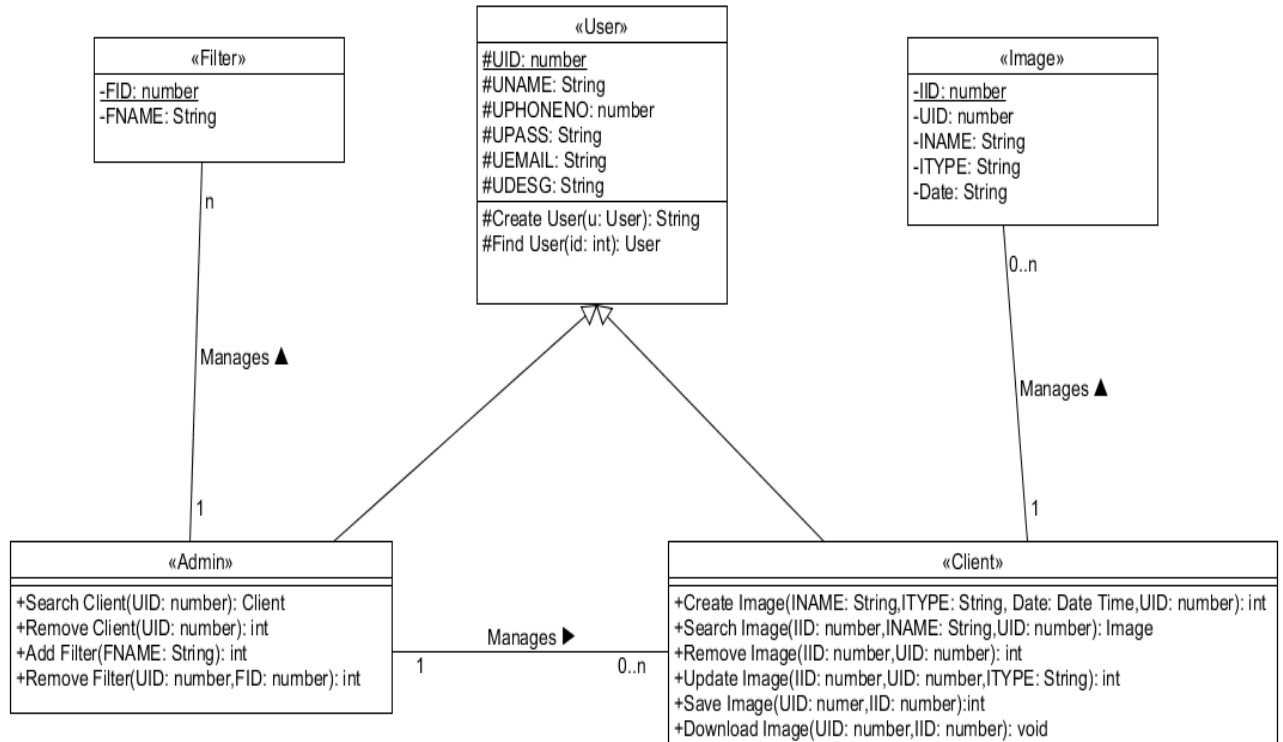


Figure 2 : Class Diagram

3. Sequence Diagram:

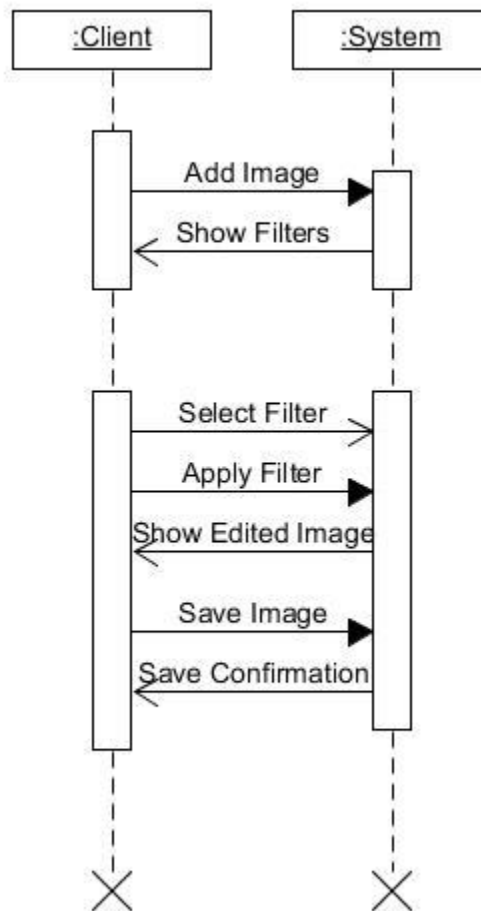


Figure 3 : Sequence Diagram

Apply Filter Scenario

4. Activity Diagrams :

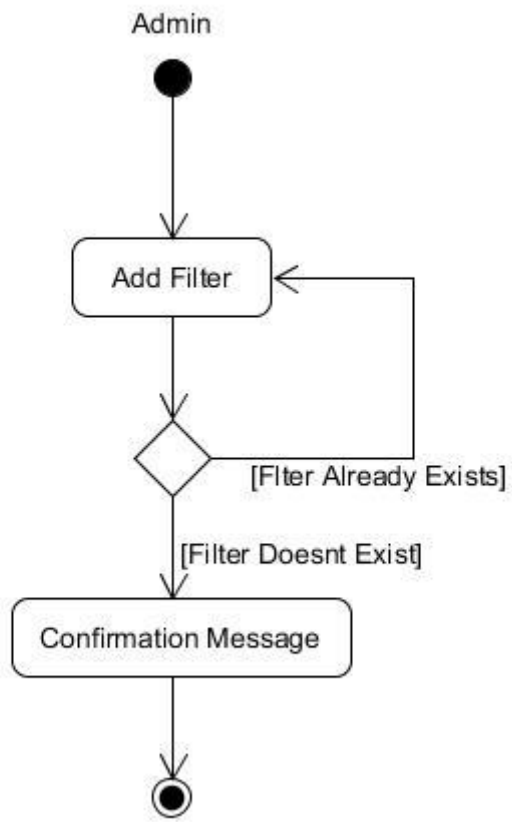


Figure 4 : Activity Diagram

Add Filter Activity

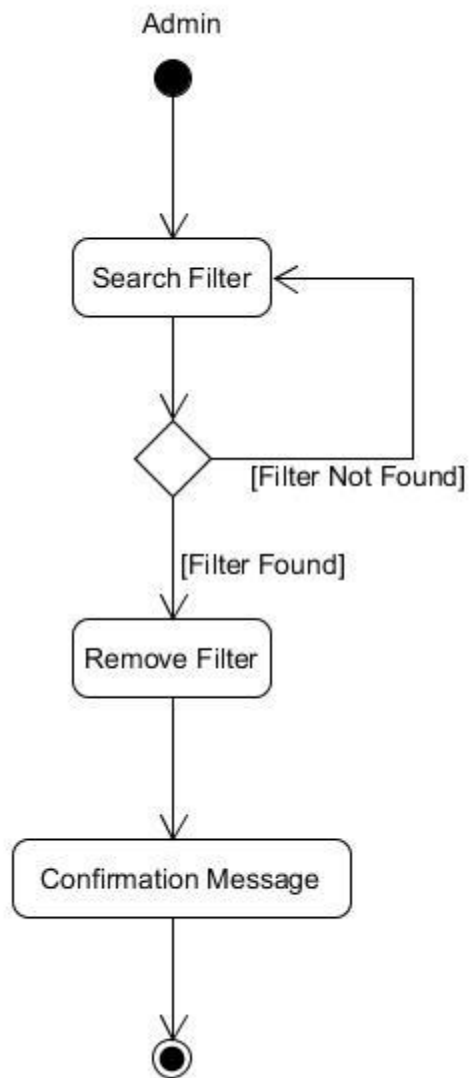


Figure 5 : Activity Diagram

Remove Filter Activity

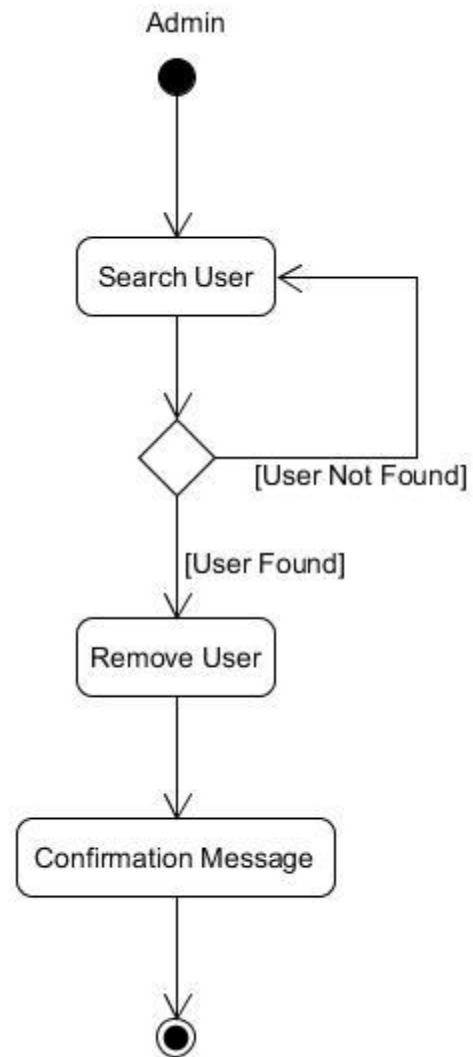


Figure 6 : Activity Diagram

Remove User Activity

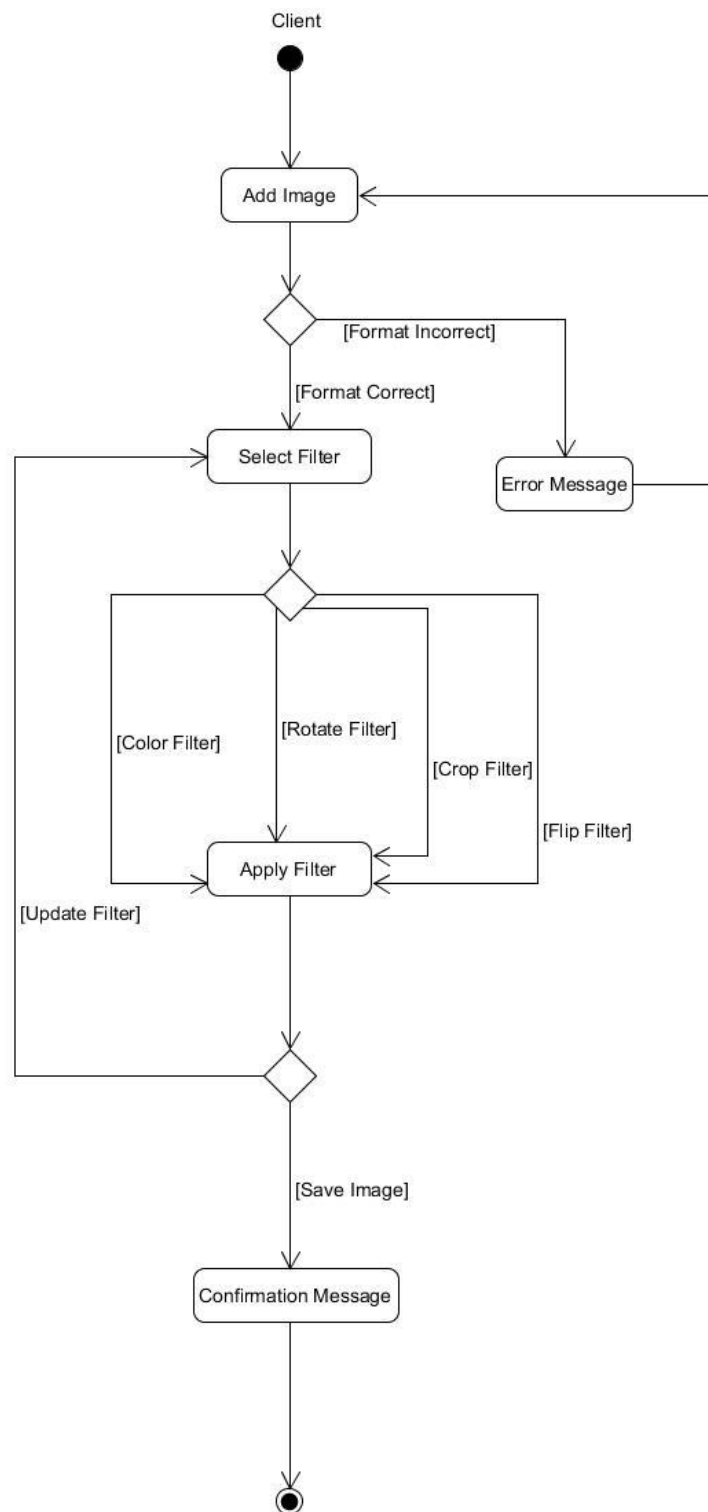


Figure 7 : Activity Diagram

Apply Filter Activity

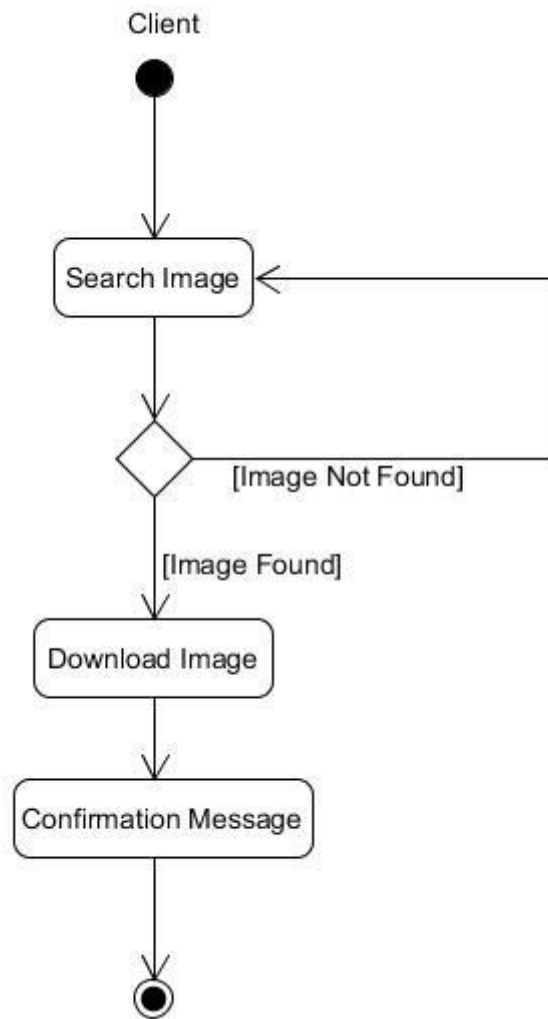


Figure 8 : Activity Diagram

Download Image Activity

5. State Diagram:

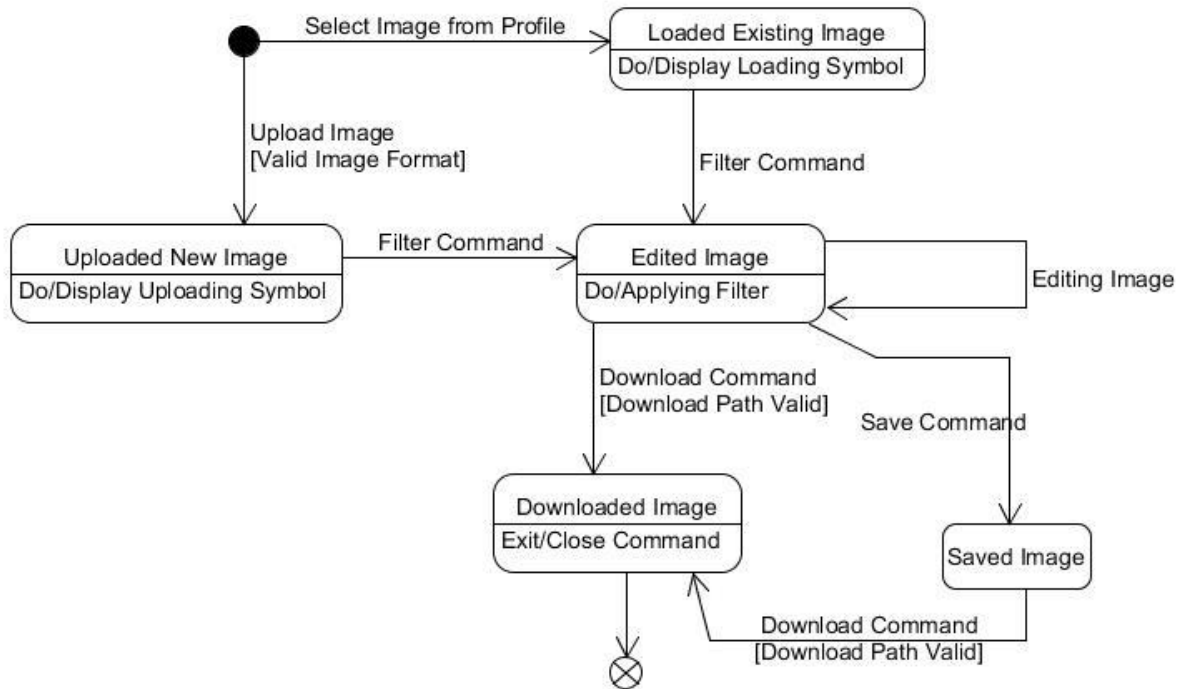


Figure 9 : State Diagram

Image State Diagram

6. E-R Diagram:

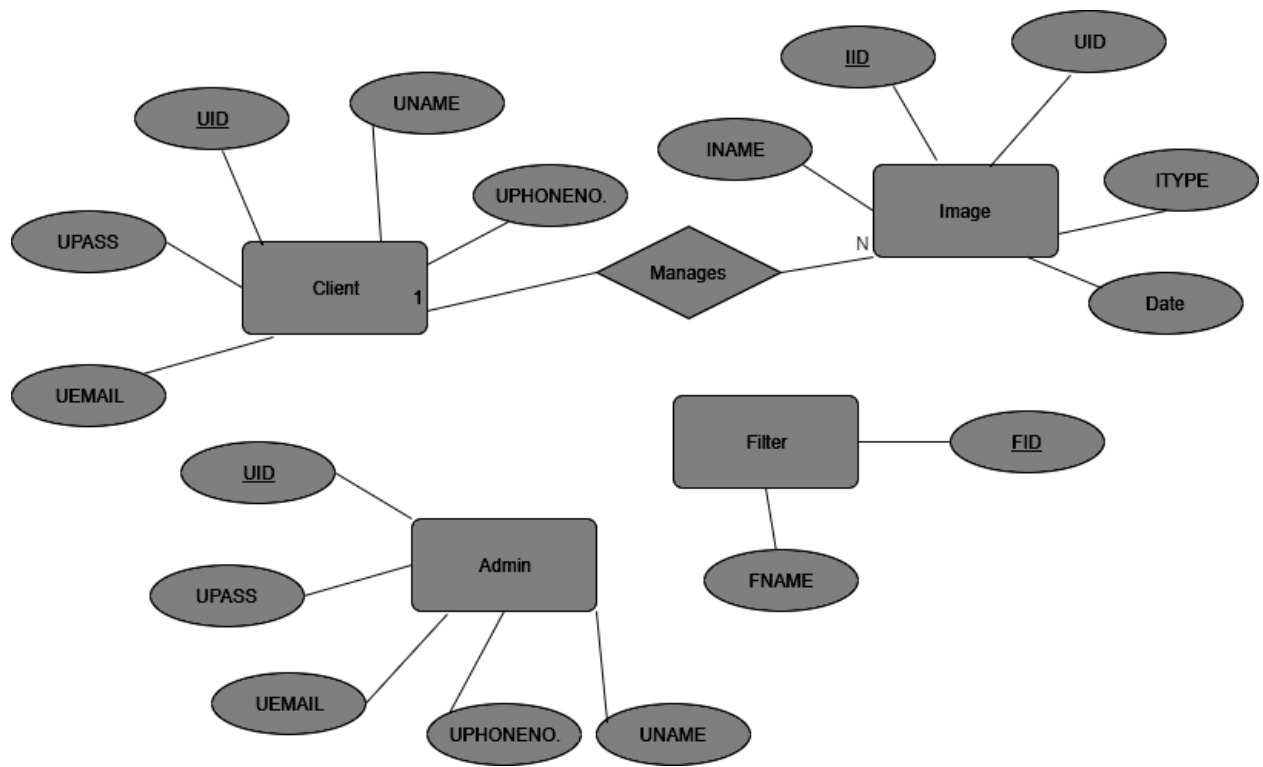


Figure 10 : E-R Diagram

7. Data Dictionary:

User			
Attributes	Datatype	Size	Feature
UID	int	20	AUTO_INC
UNAME	varchar	30	
UDES	varchar	20	
UEMAIL	varchar	20	
UPASS	varchar	20	
UPHONENO	number	10	

Table 1 : Data Dictionary

Filter			
Attributes	Datatype	Size	Feature
FID	varchar	20	AUTO_INC
FNAME	varchar	30	

Table 2 : Data Dictionary

Image			
Attributes	Datatype	Size	Feature
UID	int	20	FK
IID	int	20	AUTO_INC
INAME	varchar	20	
ITYPE	varchar	20	
Date	Date Time		

Table 3 : Data Dictionary

Implementation Details

Sign In And Sign Up Module:

I used a static variable to maintain the session of the user which has signed in. During sign in process, the user data is verified by the database and if the user is not in the database it throws an error. User can also register him/her-self using the registration form.

Editor Module:

I used the Image Processing Library of the C# to process the images that are uploaded by the user or are selected from his/her profile. I created the copy of Image Object by using the Clone Method of the Image Object thus using the prototype design pattern. User can select different variety of apply from like inversion of image, rotate image horizontally, greycaling the image, changing the contrast as well as the brightness of the image. User can also Undo and Redo the filters applied on the images. For this functionality I used the memento Design pattern to store the previous state of the Image Object.

Image Handling Module:

User can either upload image that he/she wants to edit using the upload image button. User can download the edited image using the download image button. User can save the image to his/her profile if he/she is logged in.

Database Module:

Whenever a user registers his/her self to the system the database of the users are updated. Whenever a user tries to login database is checked if the user is present in the system. Whenever Admin removes a user from the system, the database of the user is updated.

Function Prototype:

User Login Functionality:

```
private void button1_Click(object sender, EventArgs e)
{
    using (SqlConnection con = new SqlConnection(constring))
    {
        string query = "select * from [User] where Email=@email and Password=@pass";
        SqlCommand cmd = new SqlCommand(query, con);
        cmd.Parameters.Add(new SqlParameter("@email", textBox1.Text));
        cmd.Parameters.Add(new SqlParameter("@pass", textBox2.Text));
        con.Open();
        SqlDataReader rd = cmd.ExecuteReader();
        if (!rd.HasRows)
        {
            label14.Text = "Invalid Email or Password";
        }
        else
        {
            while (rd.Read())
            {
                email = rd["Email"].ToString();
                pass = rd["Password"].ToString();
                uid = Convert.ToInt32(rd["UIId"]);
                isadmin = Convert.ToBoolean(rd["Designation"]);
            }
            Login.status = true;
            this.Hide();
            this.par.ReloadEditor();
            this.par.Visible = true;
        }
    }
}
```

User Registration Functionality :

```
private void button1_Click(object sender, EventArgs e)
{
    string message = "Are These Correct Detail?\n" + "Name : " + textBox1.Text + "\n" + "Email : " + textBox2.Text + "\n" + "Password : " + textBox3.Text + "\n" + "Phone No : " + textBox4.Text;
    string details = "Details Checking";
    MessageBoxButtons buttons = MessageBoxButtons.YesNo;
    DialogResult result;

    // Displays the MessageBox.
    result = MessageBox.Show(message, details, buttons);
    if (result == System.Windows.Forms.DialogResult.Yes)
    {
        string constring = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=PhotoEditor;Integrated Security=SSPI";
        using (SqlConnection con = new SqlConnection(constring))
        {
            string query = "insert into [User](Name, Email, Password, PhoneNo, Designation) values(@name, @email, @password, @phone, @desig)";
            Boolean desig = false;
            SqlCommand cmd = new SqlCommand(query, con);
            cmd.Parameters.Add(new SqlParameter("@name", textBox1.Text));
            cmd.Parameters.Add(new SqlParameter("@email", textBox2.Text));
            cmd.Parameters.Add(new SqlParameter("@phone", textBox4.Text));
            cmd.Parameters.Add(new SqlParameter("@desig", desig));
            cmd.Parameters.Add(new SqlParameter("@pass", textBox3.Text));
            con.Open();
            cmd.ExecuteNonQuery();
        }
        // Closes the parent form.
        label15.Text="Registration was successful!";
        textBox1.Text = "";
        textBox2.Text = "";
        textBox3.Text = "";
        textBox4.Text = "";
    }
}
```

Remove User Functionality :

```
private void Ruser_Click(object sender, EventArgs e)
{
    string rmvusr = listBox1.GetItemText(listBox1.SelectedItem);

    if (MessageBox.Show("Do you want to remove user : " + rmvusr + " ?", "Remove User"))
    {
        int ruind = listBox1.SelectedIndex;
        int usrid = userList.ElementAt(ruind);
        using (SqlConnection con = new SqlConnection(constring))
        {
            string query = "DELETE FROM [User] WHERE UID = @userid";
            SqlCommand cmd = new SqlCommand(query, con);

            cmd.Parameters.Add(new SqlParameter("@userid", usrid));
            con.Open();
            cmd.ExecuteNonQuery();
        }
        listBox1.Items.RemoveAt(ruind);
        userList.RemoveAt(ruind);
        if (userlist.Count == 0)
            Ruser.Enabled = false;
    }
}
```


Display Saved Image :

```
private void contextMenuStrip1_ItemClicked(object sender, ToolStripItemClickedEventArgs e)
{
    bool displaysving = true;
    if (currimgid < 0 && pictureBox2.Image!=null)
    {
        if (MessageBox.Show("You have not saved the editor image, do you want to load the other :") == DialogResult.No)
        {
            displaysving = false;
        }
    }
    if (displaysving && contextMenuStrip1.Items.IndexOf(e.ClickedItem)>1)
    {
        var parent = (ContextMenuStrip)sender;
        int i = parent.Items.IndexOf(e.ClickedItem);
        int indedit = imglist.ElementAt(i - 2);
        string basepath = Directory.GetCurrentDirectory();
        Bitmap tpo1 = new Bitmap(basepath+"\\\"+(indedit-1)+\".png");
        Bitmap tpo2 = new Bitmap(basepath + "\\\" + (indedit)+\".png");
        pictureBox1.Image = tpo1.GetThumbnailImage(350, 350, null, new IntPtr());
        pictureBox2.Image = tpo2.GetThumbnailImage(350, 350, null, new IntPtr());
        tempbm = (Bitmap)pictureBox2.Image.Clone();
        tpo1.Dispose();
        tpo2.Dispose();
        currimgid = indedit;
        textBox1.Text = e.ClickedItem.Text;

        Clear.Enabled = true;
        Download.Enabled = true;
        Save.Enabled = true;
        Invert.Enabled = true;
        Greyscale.Enabled = true;
        Flip.Enabled = true;
        Sepia.Enabled = true;
        Fog.Enabled = true;
        Vertical.Enabled = true;
        Brightness.Enabled = true;
        Contrast.Enabled = true;
        UChanges.Clear();
        RChanges.Clear();
        Brightness.Value = 0;
        Contrast.Value = 0;
    }
}

private void Buser_Click(object sender, EventArgs e)
```

Undo and Redo Functionality :

```
private void Undo_Click(object sender, EventArgs e)
{
    if (UChanges.Count != 0)
    {
        RCAAdd((Bitmap)pictureBox2.Image.Clone());
        pictureBox2.Image = UChanges.Pop();
        tempbm = (Bitmap)pictureBox2.Image.Clone();
        if (UChanges.Count == 0)
        {
            Undo.Enabled = false;
        }
        else
        {
            Undo.Enabled = true;
        }
    }
}

private void Redo_Click(object sender, EventArgs e)
{
    if (RChanges.Count != 0)
    {
        UCAdd((Bitmap)pictureBox2.Image.Clone());
        pictureBox2.Image = RChanges.Pop();
        tempbm = (Bitmap)pictureBox2.Image.Clone();
        if (RChanges.Count == 0)
        {
            Redo.Enabled = false;
        }
        else
        {
            Redo.Enabled = true;
        }
    }
}
```

Upload Image Functionality :

```
private void Upload_Click(object sender, EventArgs e)
{
    OpenFileDialog opnfd = new OpenFileDialog();
    opnfd.Filter = "Image Files(*.jpeg;*.bmp;*.png;*.jpg)|*.jpeg;*.bmp;*.png;*.jpg";
    if (opnfd.ShowDialog() == DialogResult.OK)
    {
        pictureBox1.Image = new Bitmap(opnfd.FileName);
        tempbm = new Bitmap(opnfd.FileName);
        Clear.Enabled = true;
        Invert.Enabled = true;
        Flip.Enabled = true;
        Greyscale.Enabled = true;
        Contrast.Enabled = true;
        Fog.Enabled = true;
        Sepia.Enabled = true;
        Brightness.Enabled = true;
        Vertical.Enabled = true;
        pictureBox2.Image = null;
        Download.Enabled = false;
        Save.Enabled = false;
        RChanges.Clear();
        UChanges.Clear();
    }
}
```

Download Image Functionality :

```
private void Download_Click(object sender, EventArgs e)
{
    if (pictureBox2.Image != null)
    {
        SaveFileDialog sfd = new SaveFileDialog();
        sfd.DefaultExt = "png";
        sfd.Filter = "Images|*.png;*.bmp;*.jpg";
        ImageFormat format = ImageFormat.Png;
        if (sfd.ShowDialog() == DialogResult.OK)
        {
            string ext = System.IO.Path.GetExtension(sfd.FileName);
            switch (ext)
            {
                case ".jpg":
                    format = ImageFormat.Jpeg;
                    break;
                case ".bmp":
                    format = ImageFormat.Bmp;
                    break;
            }
            pictureBox2.Image.Save(sfd.FileName, format);
            MessageBox.Show("Image Downloaded Succesfully at location : \n" + sfd.FileName);
        }
    }
}
```

Save Image Functionality :

```
private void Save_Click(object sender, EventArgs e)
{
    string basepath = Directory.GetCurrentDirectory();
    string Ignm = "Image";
    DateTime currdate = DateTime.Now;
    if (textBox1.Text != "")
        Ignm = textBox1.Text;
    if (currimgid < 0)
    {
        int imgorg = 00;
        using (SqlConnection con = new SqlConnection(constring))
        {
            string query = "insert into [Image](UID, ImageName, ImageType, Date) OUTPUT INSERTED.IId values(@uid, @name, @type, @date)";
            SqlCommand cmd = new SqlCommand(query, con);
            cmd.Parameters.Add(new SqlParameter("@name", Ignm));
            cmd.Parameters.Add(new SqlParameter("@uid", Login.uid));
            cmd.Parameters.Add(new SqlParameter("@type", true));
            cmd.Parameters.Add(new SqlParameter("@date", currdate));
            con.Open();
            imgorg = (int)cmd.ExecuteScalar();
        }
        string basepath1 = basepath + "\\\"+imgorg.ToString()+".png";
        pictureBox1.Image.Save(basepath1, ImageFormat.Png);
        using (SqlConnection con = new SqlConnection(constring))
        {
            string query = "insert into [Image](UID, ImageName, ImageType, Date) OUTPUT INSERTED.IId values(@uid, @name, @type, @date)";
            SqlCommand cmd = new SqlCommand(query, con);
            cmd.Parameters.Add(new SqlParameter("@name", Ignm));
            cmd.Parameters.Add(new SqlParameter("@uid", Login.uid));
            cmd.Parameters.Add(new SqlParameter("@type", false));
            cmd.Parameters.Add(new SqlParameter("@date", currdate));
            con.Open();
        }
    }
}
```

```
string query = "insert into [Image](UID, ImageName, ImageType, Date) OUTPUT INSERTED.IId values(@uid, @name, @type, @date)";
SqlCommand cmd = new SqlCommand(query, con);
cmd.Parameters.Add(new SqlParameter("@name", Ignm));
cmd.Parameters.Add(new SqlParameter("@uid", Login.uid));
cmd.Parameters.Add(new SqlParameter("@type", false));
cmd.Parameters.Add(new SqlParameter("@date", currdate));
con.Open();
currimgid = (int)cmd.ExecuteScalar();
}
string basepath2 = basepath + "\\\" + currimgid.ToString() + ".png";
imglist.Add(currimgid);
contextMenuStrip1.Items.Add(Ignm);
pictureBox2.Image.Save(basepath2, ImageFormat.Png);
MessageBox.Show("Image Stored successfully for the first time.");
}
else
{
    if (MessageBox.Show("You have already saved this image do you wish to overwrite it ?", "Override", MessageBoxButtons.YesNo) == DialogResult.Yes)
    {
        using (SqlConnection con = new SqlConnection(constring))
        {
            currdate = DateTime.Now;
            string query = "UPDATE Image SET Date = @dnew, ImageName = @ignm WHERE IId = @imgid";
            SqlCommand cmd = new SqlCommand(query, con);
            cmd.Parameters.Add(new SqlParameter("@ignm", Ignm));
            cmd.Parameters.Add(new SqlParameter("@dnew", currdate));
            cmd.Parameters.Add(new SqlParameter("@imgid", currimgid));
            con.Open();
            cmd.ExecuteNonQuery();
        }
        string basepath3 = basepath + "\\\" + currimgid.ToString() + ".png";
        pictureBox2.Image.Save(basepath3, ImageFormat.Png);
    }
}
```

Login , Registration and Logout Functionality for editor :

```
private void registerToolStripMenuItem_Click(object sender, EventArgs e)
{
    r = new Registration(this);
    r.Show();
    this.Hide();
}

private void loginToolStripMenuItem_Click(object sender, EventArgs e)
{
    l = new Login(this);
    l.Show();
    this.Hide();
}

private void logoutToolStripMenuItem_Click(object sender, EventArgs e)
{
    l.Dispose();
    contextMenuStrip1.Items.RemoveAt(1);
    int rmv = imglist.Count;
    for (int i = 0; i < rmv; i++)
        contextMenuStrip1.Items.RemoveAt(1);
    imglist.Clear();
    NoImage();
    Profile.Hide();
    Options.Show();
}
```

Different Filters :

```
private void Sepia_Click(object sender, EventArgs e)
{
    if (pictureBox1.Image != null)
    {
        Download.Enabled = true;
        if (Login.status)
            Save.Enabled = true;
        Bitmap bmap = tempbm;

        UGAdd((Bitmap)tempbm.Clone());
        Bitmap bmpInverted = new Bitmap(bmap.Width, bmap.Height);
        ImageAttributes ia = new ImageAttributes();
        ColorMatrix cmPicture = new ColorMatrix(new float[][]
        {
            new float[] { .393f, .349f, .272f, 0, 0 },
            new float[] { .769f, .686f, .534f, 0, 0 },
            new float[] { .189f, .168f, .131f, 0, 0 },
            new float[] { 0, 0, 0, 1, 0 },
            new float[] { 0, 0, 0, 0, 1 }
        });
        ia.SetColorMatrix(cmPicture);
        Graphics g = Graphics.FromImage(bmpInverted);
        g.DrawImage(bmap, new Rectangle(0, 0, bmap.Width, bmap.Height), 0, 0, bmap.Width, bmap.Height, GraphicsUnit.Pixel, ia);
        g.Dispose();
        pictureBox2.Image = bmpInverted;
        tempbm = bmpInverted;
        RChanges.Clear();
    }
}
```

```
public void SetBrightness(int brightness)
{
    Download.Enabled = true;
    if (Login.status)
        Save.Enabled = true;
    //Bitmap temp = (Bitmap)_currentBitmap;
    Bitmap bmap = (Bitmap)tempbm.Clone();
    if (brightness < -255) brightness = -255;
    if (brightness > 255) brightness = 255;
    Color c;
    for (int i = 0; i < bmap.Width; i++)
    {
        for (int j = 0; j < bmap.Height; j++)
        {
            c = bmap.GetPixel(i, j);
            int cR = c.R + brightness;
            int cG = c.G + brightness;
            int cB = c.B + brightness;

            if (cR < 0) cR = 1;
            if (cR > 255) cR = 255;

            if (cG < 0) cG = 1;
            if (cG > 255) cG = 255;

            if (cB < 0) cB = 1;
            if (cB > 255) cB = 255;

            bmap.SetPixel(i, j,
                Color.FromArgb((byte)cR, (byte)cG, (byte)cB));
        }
    }
    pictureBox2.Image = bmap;
}
```

```

private void Flip_Click(object sender, EventArgs e)
{
    if (pictureBox1.Image != null)
    {
        Download.Enabled = true;
        if (Login.status)
            Save.Enabled = true;
        Bitmap bmap = tempbm;

        UCAAdd((Bitmap)tempbm.Clone());
        bmap.RotateFlip(RotateFlipType.Rotate180FlipY);
        pictureBox2.Image = bmap;

        tempbm = bmap;
        RChanges.Clear();
    }
}

private void Vertical_Click(object sender, EventArgs e)
{
    if (pictureBox1.Image != null)
    {
        Download.Enabled = true;
        if (Login.status)
            Save.Enabled = true;
        Bitmap bmap = tempbm;

        UCAAdd((Bitmap)tempbm.Clone());
        bmap.RotateFlip(RotateFlipType.Rotate180FlipX);
        pictureBox2.Image = bmap;

        tempbm = bmap;
        RChanges.Clear();
    }
}

```

```

public void SetContrast(double contrast)
{
    Download.Enabled = true;
    if (Login.status)
        Save.Enabled = true;
    Bitmap bmap = (Bitmap)tempbm.Clone();
    if (contrast < -100) contrast = -100;
    if (contrast > 100) contrast = 100;
    contrast = (100.0 + contrast) / 100.0;
    contrast *= contrast;
    Color c;
    for (int i = 0; i < bmap.Width; i++)
    {
        for (int j = 0; j < bmap.Height; j++)
        {
            c = bmap.GetPixel(i, j);
            double pR = c.R / 255.0;
            pR -= 0.5;
            pR *= contrast;
            pR += 0.5;
            pR *= 255;
            if (pR < 0) pR = 0;
            if (pR > 255) pR = 255;

            double pG = c.G / 255.0;
            pG -= 0.5;
            pG *= contrast;
            pG += 0.5;
            pG *= 255;
            if (pG < 0) pG = 0;
            if (pG > 255) pG = 255;

            double pB = c.B / 255.0;
            pB -= 0.5;
            pB *= contrast;
            pB += 0.5;
            pB *= 255;
            if (pB < 0) pB = 0;
            if (pB > 255) pB = 255;

            bmap.SetPixel(i, j,
                Color.FromArgb((byte)pR, (byte)pG, (byte)pB));
        }
    }
    pictureBox2.Image = bmap;
}

```



```

private void Fog_Click(object sender, EventArgs e)
{
    if (pictureBox1.Image != null)
    {
        Download.Enabled = true;
        if (Login.status)
            Save.Enabled = true;
        Bitmap bmap = tempbm;

        UCAdd((Bitmap)tempbm.Clone());
        Bitmap bmpInverted = new Bitmap(bmap.Width, bmap.Height);
        ImageAttributes ia = new ImageAttributes();
        ColorMatrix cmPicture = new ColorMatrix(new float[][]
        {
            new float[] {1+0.3f, 0, 0, 0, 0},
            new float[] {0, 1+0.7f, 0, 0, 0},
            new float[] {0, 0, 1+1.3f, 0, 0},
            new float[] {0, 0, 0, 1, 0},
            new float[] {0, 0, 0, 0, 1}
        });
        ia.SetColorMatrix(cmPicture);
        Graphics g = Graphics.FromImage(bmpInverted);
        g.DrawImage(bmap, new Rectangle(0, 0, bmap.Width, bmap.Height), 0, 0, bmap.Width, bmap.Height, GraphicsUnit.Pixel, ia);
        g.Dispose();
        pictureBox2.Image = bmpInverted;
        tempbm = bmpInverted;
        RChanges.Clear();
    }
}

private void Invert_Click(object sender, EventArgs e)
{
    if (pictureBox1.Image != null)
    {
        Download.Enabled = true;
        if (Login.status)
            Save.Enabled = true;
        Bitmap bmap = tempbm;

        UCAdd((Bitmap)tempbm.Clone());
        Bitmap bmpInverted = new Bitmap(bmap.Width, bmap.Height);
        ImageAttributes ia = new ImageAttributes();
        ColorMatrix cmPicture = new ColorMatrix(new float[][]
        {
            new float[] {-1, 0, 0, 0, 0},
            new float[] {0, -1, 0, 0, 0},
            new float[] {0, 0, -1, 0, 0},
            new float[] {0, 0, 0, 1, 0},
            new float[] {1, 1, 1, 0, 1}
        });
        ia.SetColorMatrix(cmPicture);
        Graphics g = Graphics.FromImage(bmpInverted);
        g.DrawImage(bmap, new Rectangle(0, 0, bmap.Width, bmap.Height), 0, 0, bmap.Width, bmap.Height, GraphicsUnit.Pixel, ia);
        g.Dispose();
        pictureBox2.Image = bmpInverted;
        tempbm = bmpInverted;
        RChanges.Clear();
    }
}

```

```

private void Clear_Click(object sender, EventArgs e)
{
    Brightness.Value = 0;
    Contrast.Value = 0;
    NoImage();
    RChanges.Clear();
    UChanges.Clear();
    Undo.Enabled = false;
    Redo.Enabled = false;
}

```

```

private void Greyscale_Click(object sender, EventArgs e)
{
    if (pictureBox1.Image != null)
    {
        Download.Enabled = true;
        if (Login.status)
            Save.Enabled = true;
        Bitmap bmap = tempbm;

        UCAAdd((Bitmap)tempbm.Clone());
        Bitmap bmpInverted = new Bitmap(bmap.Width, bmap.Height);
        ImageAttributes ia = new ImageAttributes();
        ColorMatrix cmPicture = new ColorMatrix(
new float[][]
{
    new float[] { .3f, .3f, .3f, 0, 0 },
    new float[] { .59f, .59f, .59f, 0, 0 },
    new float[] { .11f, .11f, .11f, 0, 0 },
    new float[] { 0, 0, 0, 1, 0 },
    new float[] { 0, 0, 0, 0, 1 }
});
        ia.SetColorMatrix(cmPicture);
        Graphics g = Graphics.FromImage(bmpInverted);
        g.DrawImage(bmap, new Rectangle(0, 0, bmap.Width, bmap.Height), 0, 0, bmap.Width, bmap.Height, GraphicsUnit.Pixel, ia);
        g.Dispose();
        pictureBox2.Image = bmpInverted;
        tempbm = bmpInverted;
        RChanges.Clear();
    }
}

```

Testing Details

Testing Methods:

Sign In Testing :

Username : abc	Username : abc
Password : 123	Password : abc
Output : Invalid	Output : Valid

Registration Testing :

Name : abcd

*Email : abc@gmail.com

*Password : 123abc

*Phone No. : 1234567890

Output : Registration Successful.

If any of the * fields is empty :

Output : Registration Failed.

Upload Testing :

Input : Valid Image File Path

Output : Picture Displayed in the PictureBox

Download Testing :

Input : Valid Destination Path with Valid Image format

Output : Picture saved successfully and MessageBox displayed.

Save Testing :

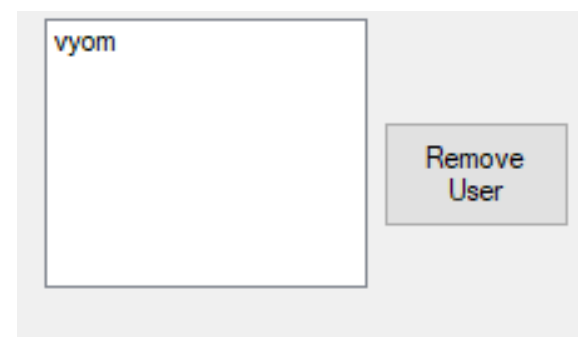
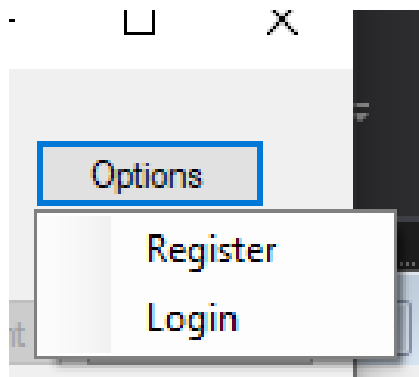
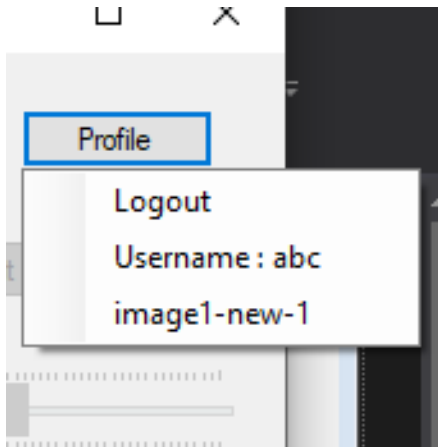
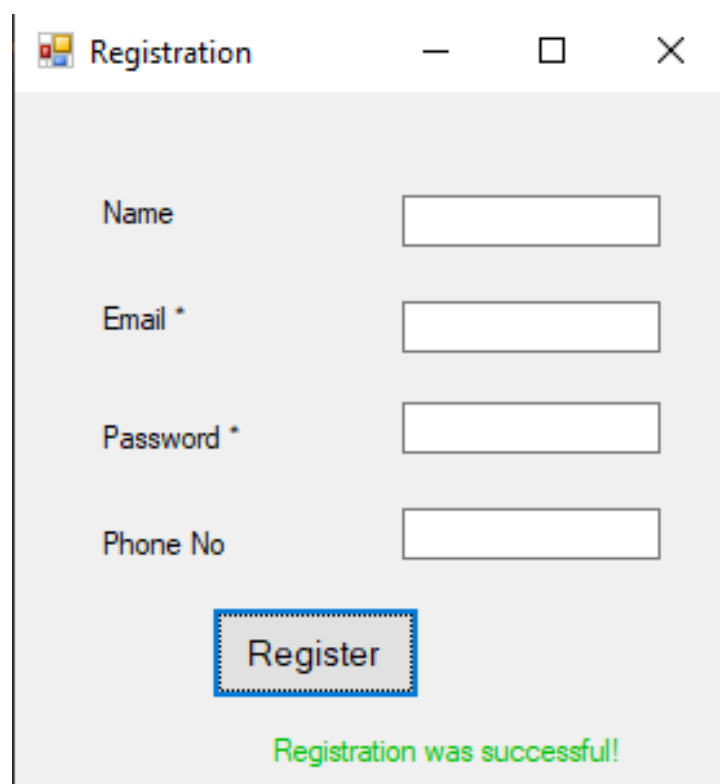
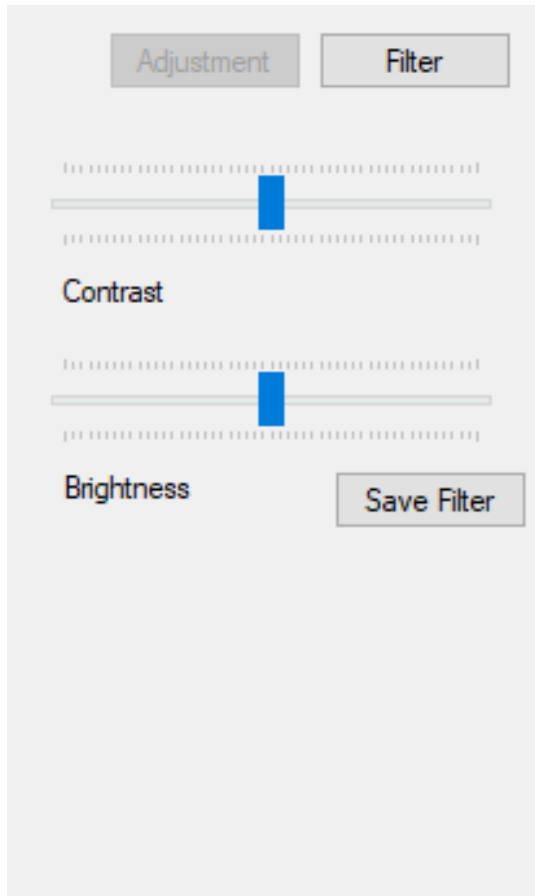
Input : New Image in the PictureBox

Output : Image stored in the server as well as the path stored in the database and MessageBox displayed.

Input : Overriding already saved image in the PictureBox

Output : Image name overridden if changed in the database as well as time of the image updated and MessageBox displayed.

Screen-Shots



Login

Email

Password

Login

Registration

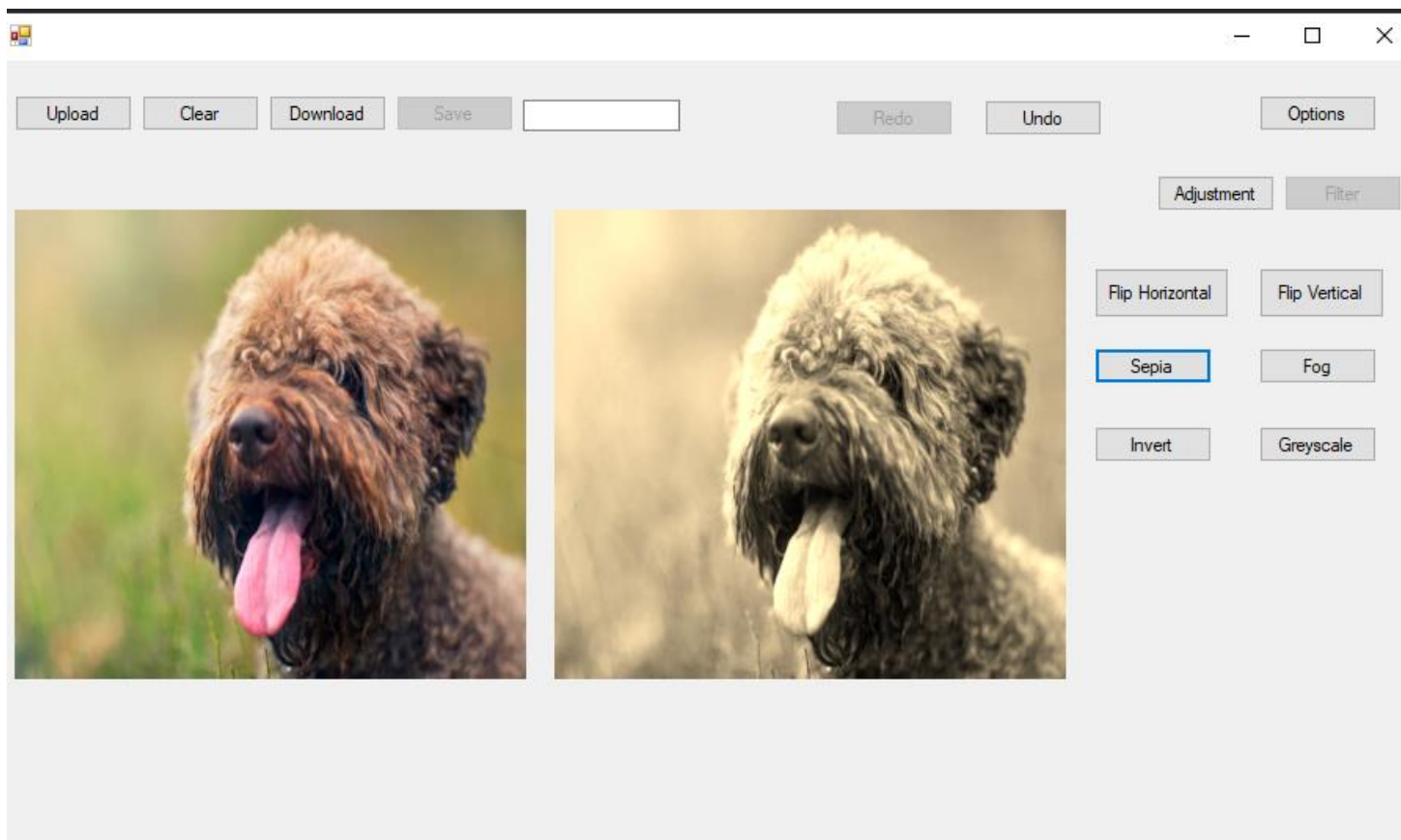
Name

Email *

Password *

Phone No

Register



Conclusion

I successfully implemented the user registration functionality as well as the user login functionality. I also successfully implemented the image upload functionality as well as image download functionality. I also successfully implemented the image editing functionality using the C# in-built Image Processing Library. I was also able to implement the Image saved functionality that saved images can be re-edited for a particular image. Admin side functionality to remove user was successfully implemented.

Limitation and Future Extension

Limitations:

- ✚ Undo and redo operation can only be done 5 times.
- ✚ User cannot download the saved images from her profile.
- ✚ User Images cannot be deleted from his profile.

Future Extension:

- ✚ More filters can be added to increase the image manipulation done by the user.
- ✚ Some styling/color can be added to the whole application looks too simple.
- ✚ User can remove his/her image from the profile.

Bibliography

References:

<https://stackoverflow.com/>
<https://www.youtube.com/>
<https://www.geeksforgeeks.org/>
<https://docs.microsoft.com/en-us/dotnet/>
<https://www.codeproject.com/>
<https://www.c-sharpcorner.com/>