Contents

List of tables

List of Figures

Overview

Universities does a lot of researches. Wither it is from a student or professor, it is essential for author to know the quality of their research before gaining confident to finally submit or publish it. The data we collect from different sources might not always be correct hence creating an error in our research, which usually goes unnoticed by the author. And to prevent such errors and allow author to be confident about their research, and correct possible anomalies in research documents they prepared, This Project is expected to be a solution. Not only the options provided includes “universities”, but also the project best fits it’s context. So university is selected.

Benefits:

- Error Free Research.
- Expert’s review which is free of cost.
- Use and sharing or knowledge in appropriate way.

Introduction

“Peer review tracking system for university” allows users to register and then upload research documents which will be reviewed. The author will have an option to select the skill required and minimum rating required for reviewer. Users are allowed to select their specialist skills while creating account, which they can add later from profile. Users will be assigned works based on their skill points on particular topic. The peer review task will be ideally balanced such that each user is assigned almost equal numbers of tasks. However, a little leeway is given to the work where reviewers are in small amount. A reviewer can add multiple reviews to the research documents. These reviews will also be visible for the author who uploaded document. Author are allowed to rate each piece of review that they receive. After that the average of ratings are converted to score that are visible to reviewers on their profile. These sources are assigned depending on specialism of the work. Multiple scores relating to different skills is possible for reviewers. The system allows users to search comments and captions on their work, the application also permits users to access the things that they have submitted in the system by permitting to delete or edit them.

Task 1: Candidate class list and Diagram

Nouns as Classes
Program, listPanel, Post, Add_Post, Add_Topic, Admin, Edit, Home, Login, Register, Crypto, DBManager

Nouns for Classes(selected)	*Reasons*
Program	Main Entry Point of Program
listPanel	Custom Widget made by implementing and modifying default System.Windows.Forms.Panel to list out users for Admin
Post	Custom Widget made by implementing and modifying default System.Windows.Forms.Panel To form a container which holds all information related to a post i.e. Actions, Reviews etc.
Add_Post	A class which creates Full Featured window to add a post in user profile.
Add_Topic	A class which is responsible to add topics(i.e. special skills)
Admin	A class to display windows which organizes all activities which can be operated by an admin.
Edit	Class to form a simple edit text dialog box. To edit post/review.
Home	Main window that is shown to user after they login, it is the window where users interact with own posts and assigned ones.
Login	Class that manages login operation.
Register	Class that manages registration and account updating actions.
Crypto	Class that is used to encode passwords with SHA256 encryption.
DBManager	Class that manages overall database related activities(Except few minor ones)

Table 1: List of candidate class

Class diagrams in Unified Modeling Language can be defined as a static structure diagram that is used to describe basic structure of classes in order to give clean concept on how a particular class is formed, how is it related to any other classes, how different classes connects to each other to form complete system and what might be the actual purpose of particular class in software to be (or which is) developed.

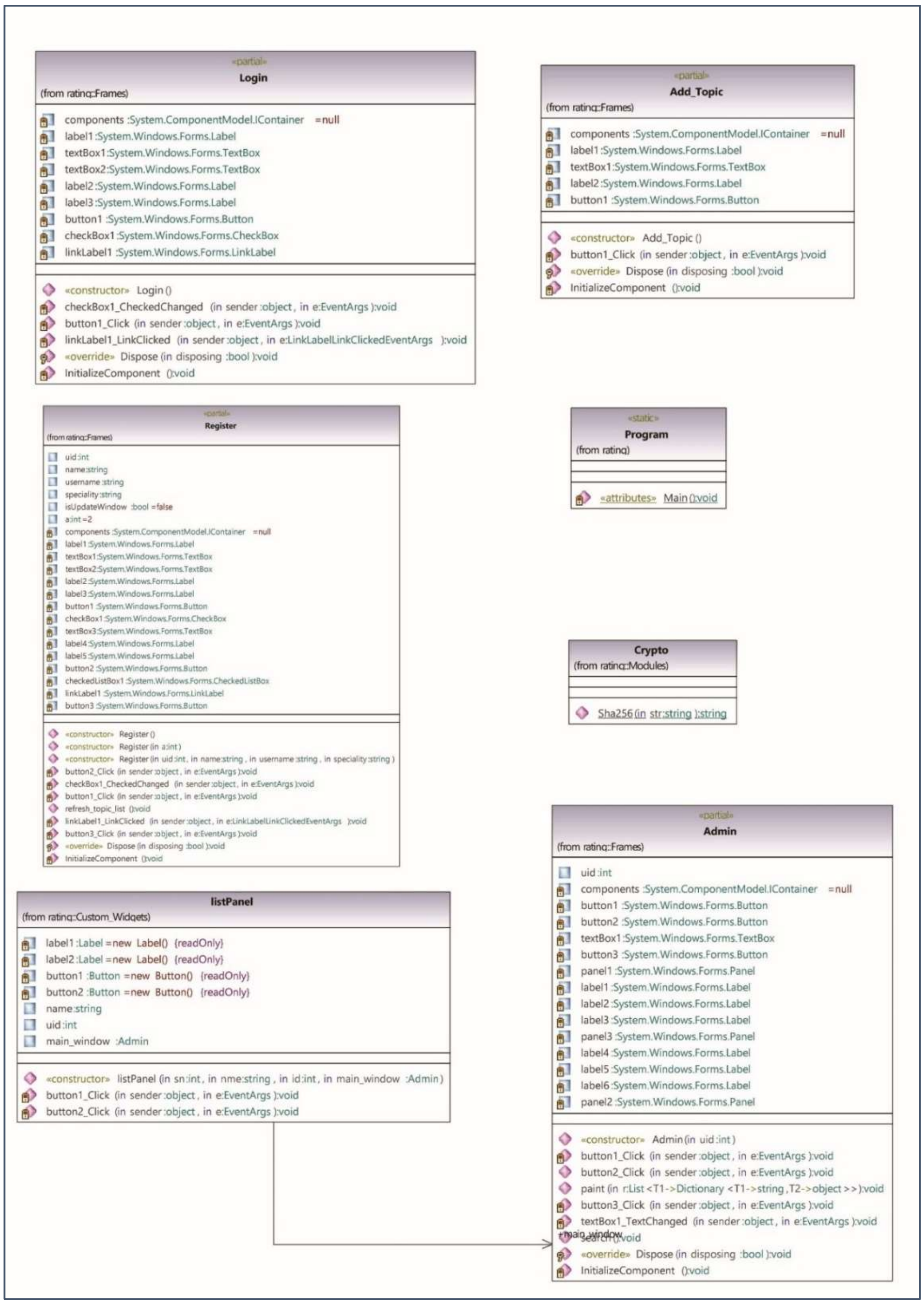
«partial»
Login
(from rating:Frames)
components :System.ComponentModel.IContainer =null
label1 :System.Windows.Forms.Label
textBox1 :System.Windows.Forms.TextBox
textBox2 :System.Windows.Forms.TextBox
label2 :System.Windows.Forms.Label
label3 :System.Windows.Forms.Label
button1 :System.Windows.Forms.Button
checkBox1 :System.Windows.Forms.CheckBox
linkLabel1 :System.Windows.Forms.LinkLabel
«constructor» Login ()
checkBox1_CheckedChanged (in sender :object , in e:EventArgs):void
button1_Click (in sender :object , in e:EventArgs):void
linkLabel1_LinkClicked (in sender :object , in e:LinkLabelLinkClickedEventArgs):void
«override» Dispose (in disposing :bool):void
InitializeComponent ():void
«partial»
Add_Topic
(from rating:Frames)
components :System.ComponentModel.IContainer =null
label1 :System.Windows.Forms.Label
textBox1 :System.Windows.Forms.TextBox
label2 :System.Windows.Forms.Label
button1 :System.Windows.Forms.Button
«constructor» Add_Topic ()
button1_Click (in sender :object , in e:EventArgs):void
«override» Dispose (in disposing :bool):void
InitializeComponent ():void
«partial»
Register
(from rating:Frames)
uid :int
name :string
username :string
speciality :string
isUpdateWindow :bool =false
a :int =2
components :System.ComponentModel.IContainer =null
label1 :System.Windows.Forms.Label
textBox1 :System.Windows.Forms.TextBox
textBox2 :System.Windows.Forms.TextBox
label2 :System.Windows.Forms.Label
label3 :System.Windows.Forms.Label
button1 :System.Windows.Forms.Button
checkBox1 :System.Windows.Forms.CheckBox
textBox3 :System.Windows.Forms.TextBox
label4 :System.Windows.Forms.Label
label5 :System.Windows.Forms.Label
button2 :System.Windows.Forms.Button
checkedListBox1 :System.Windows.Forms.CheckedListBox
linkLabel1 :System.Windows.Forms.LinkLabel
button3 :System.Windows.Forms.Button
«constructor» Register ()
«constructor» Register (in a:int)
«constructor» Register (in uid :int , in name:string , in username :string , in speciality :string)
button2_Click (in sender :object , in e:EventArgs):void
checkBox1_CheckedChanged (in sender :object , in e:EventArgs):void
button1_Click (in sender :object , in e:EventArgs):void
refresh_topic_list ():void
linkLabel1_LinkClicked (in sender :object , in e:LinkLabelLinkClickedEventArgs):void
button3_Click (in sender :object , in e:EventArgs):void
«override» Dispose (in disposing :bool):void
InitializeComponent ():void
«static»
Program
(from rating)
«attributes» Main ():void
Crypto
(from rating:Modules)
Sha256 (in str:string):string
listPanel
(from rating:Custom_Widgets)
label1 :Label =new Label() {readOnly}
label2 :Label =new Label() {readOnly}
button1 :Button =new Button() {readOnly}
button2 :Button =new Button() {readOnly}
name :string
uid :int
main_window :Admin
«constructor» listPanel (in sn:int , in nme:string , in id:int , in main_window :Admin)
button1_Click (in sender :object , in e:EventArgs):void
button2_Click (in sender :object , in e:EventArgs):void
«partial»
Admin
(from rating:Frames)
uid :int
components :System.ComponentModel.IContainer =null
button1 :System.Windows.Forms.Button
button2 :System.Windows.Forms.Button
textBox1 :System.Windows.Forms.TextBox
button3 :System.Windows.Forms.Button
panel1 :System.Windows.Forms.Panel
label1 :System.Windows.Forms.Label
label2 :System.Windows.Forms.Label
label3 :System.Windows.Forms.Label
panel3 :System.Windows.Forms.Panel
label4 :System.Windows.Forms.Label
label5 :System.Windows.Forms.Label
label6 :System.Windows.Forms.Label
panel2 :System.Windows.Forms.Panel
«constructor» Admin (in uid :int)
button1_Click (in sender :object , in e:EventArgs):void
button2_Click (in sender :object , in e:EventArgs):void
paint (in r:List <T1->Dictionary <T1->string ,T2->object >>):void
button3_Click (in sender :object , in e:EventArgs):void
textBox1_TextChanged (in sender :object , in e:EventArgs):void
search ():void
+main_window
«override» Dispose (in disposing :bool):void
InitializeComponent ():void

Figure 1: Class Diagram for Peer review tracking system (Part - 1)

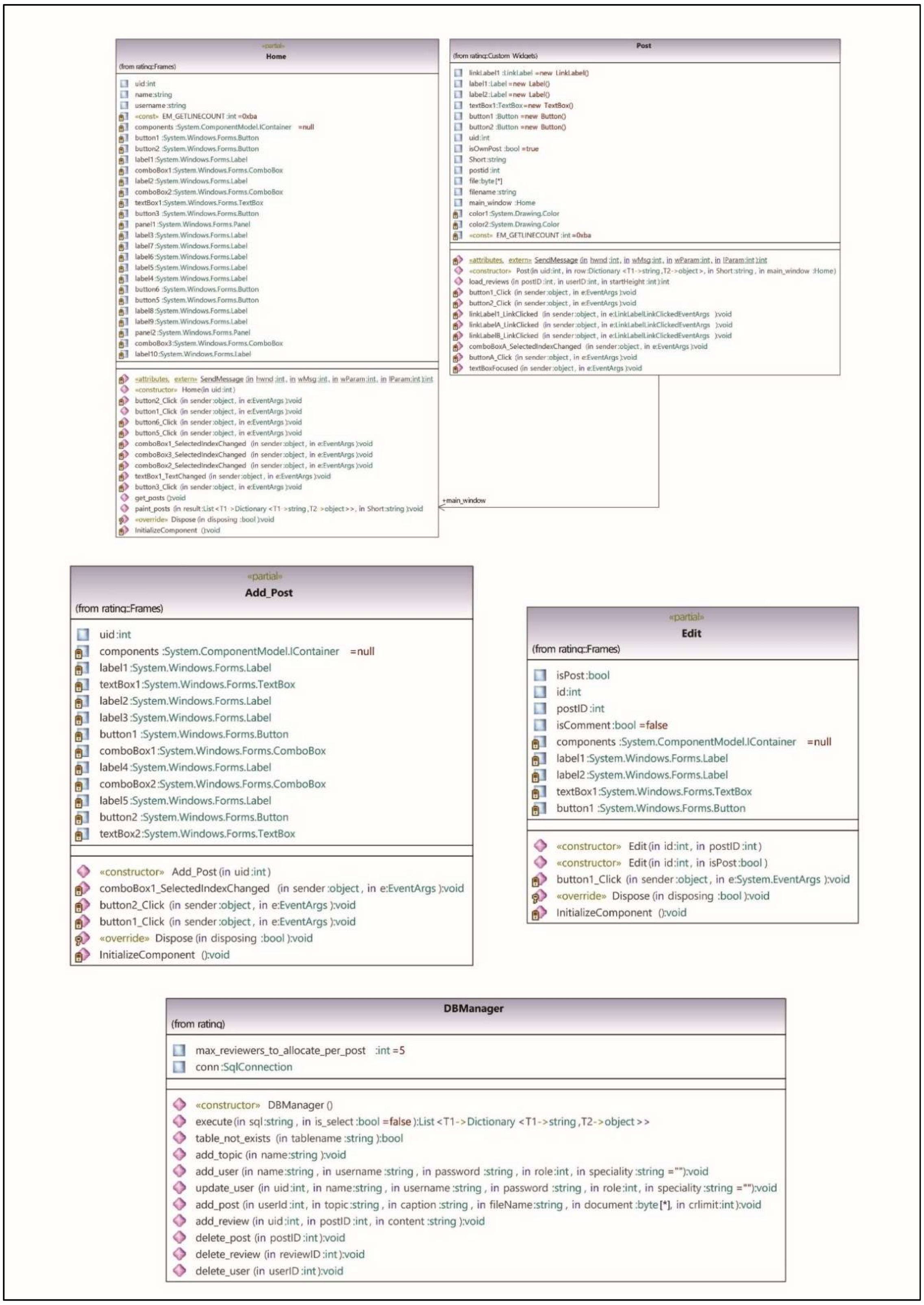
«partial»
Home
(from rating::Frames)
uid:int
name:string
username:string
«const» EM_GETLINECOUNT :int =0xba
components :System.ComponentModel.IContainer =null
button1 :System.Windows.Forms.Button
button2 :System.Windows.Forms.Button
label1 :System.Windows.Forms.Label
comboBox1 :System.Windows.Forms.ComboBox
label2 :System.Windows.Forms.Label
comboBox2 :System.Windows.Forms.ComboBox
textBox1 :System.Windows.Forms.TextBox
button3 :System.Windows.Forms.Button
panel1 :System.Windows.Forms.Panel
label3 :System.Windows.Forms.Label
label7 :System.Windows.Forms.Label
label6 :System.Windows.Forms.Label
label5 :System.Windows.Forms.Label
label4 :System.Windows.Forms.Label
button6 :System.Windows.Forms.Button
button5 :System.Windows.Forms.Button
label8 :System.Windows.Forms.Label
label9 :System.Windows.Forms.Label
panel2 :System.Windows.Forms.Panel
comboBox3 :System.Windows.Forms.ComboBox
label10 :System.Windows.Forms.Label
«attributes, extern» SendMessage (in hwnd :int, in wMsg :int, in wParam :int, in lParam :int):int
«constructor» Home(in uid :int)
button2_Click (in sender :object, in e:EventArgs):void
button1_Click (in sender :object, in e:EventArgs):void
button6_Click (in sender :object, in e:EventArgs):void
button5_Click (in sender :object, in e:EventArgs):void
comboBox1_SelectedIndexChanged (in sender :object, in e:EventArgs):void
comboBox3_SelectedIndexChanged (in sender :object, in e:EventArgs):void
comboBox2_SelectedIndexChanged (in sender :object, in e:EventArgs):void
textBox1_TextChanged (in sender :object, in e:EventArgs):void
button3_Click (in sender :object, in e:EventArgs):void
get_posts ():void
paint_posts (in result :List <T1 ->Dictionary <T1 ->string ,T2 ->object >>, in Short:string):void
«override» Dispose (in disposing :bool):void
InitializeComponent ():void
+main_window
Post
(from rating::Custom Widgets)
linkLabel1 :LinkLabel =new LinkLabel()
label1 :Label =new Label()
label2 :Label =new Label()
textBox1 :TextBox =new TextBox()
button1 :Button =new Button()
button2 :Button =new Button()
uid :int
isOwnPost :bool =true
Short :string
postid :int
file :byte [*]
filename :string
main_window :Home
color1 :System.Drawing.Color
color2 :System.Drawing.Color
«const» EM_GETLINECOUNT :int =0xba
«attributes, extern» SendMessage (in hwnd :int, in wMsg :int, in wParam :int, in lParam :int):int
«constructor» Post(in uid :int, in row :Dictionary <T1->string ,T2->object >, in Short :string , in main_window :Home)
load_reviews (in postID :int, in userID :int, in startHeight :int):int
button1_Click (in sender :object, in e:EventArgs):void
button2_Click (in sender :object, in e:EventArgs):void
linkLabel1_LinkClicked (in sender :object, in e:LinkLabelLinkClickedEventArgs):void
linkLabelA_LinkClicked (in sender :object, in e:LinkLabelLinkClickedEventArgs):void
linkLabelB_LinkClicked (in sender :object, in e:LinkLabelLinkClickedEventArgs):void
comboBoxA_SelectedIndexChanged (in sender :object, in e:EventArgs):void
buttonA_Click (in sender :object, in e:EventArgs):void
textBoxFocused (in sender :object, in e:EventArgs):void
«partial»
Add_Post
(from rating::Frames)
uid :int
components :System.ComponentModel.IContainer =null
label1 :System.Windows.Forms.Label
textBox1 :System.Windows.Forms.TextBox
label2 :System.Windows.Forms.Label
label3 :System.Windows.Forms.Label
button1 :System.Windows.Forms.Button
comboBox1 :System.Windows.Forms.ComboBox
label4 :System.Windows.Forms.Label
comboBox2 :System.Windows.Forms.ComboBox
label5 :System.Windows.Forms.Label
button2 :System.Windows.Forms.Button
textBox2 :System.Windows.Forms.TextBox
«constructor» Add_Post (in uid :int)
comboBox1_SelectedIndexChanged (in sender :object, in e:EventArgs):void
button2_Click (in sender :object, in e:EventArgs):void
button1_Click (in sender :object, in e:EventArgs):void
«override» Dispose (in disposing :bool):void
InitializeComponent ():void
«partial»
Edit
(from rating::Frames)
isPost :bool
id :int
postID :int
isComment :bool =false
components :System.ComponentModel.IContainer =null
label1 :System.Windows.Forms.Label
label2 :System.Windows.Forms.Label
textBox1 :System.Windows.Forms.TextBox
button1 :System.Windows.Forms.Button
«constructor» Edit (in id :int, in postID :int)
«constructor» Edit (in id :int, in isPost :bool)
button1_Click (in sender :object, in e:System.EventArgs):void
«override» Dispose (in disposing :bool):void
InitializeComponent ():void
DBManager
(from rating)
max_reviewers_to_allocate_per_post :int =5
conn :SqlConnection
«constructor» DBManager ()
execute (in sql :string , in is_select :bool =false):List <T1->Dictionary <T1->string ,T2->object >>
table_not_exists (in tablename :string):bool
add_topic (in name :string):void
add_user (in name :string , in username :string , in password :string , in role :int, in speciality :string =""):void
update_user (in uid :int, in name :string , in username :string , in password :string , in role :int, in speciality :string =""):void
add_post (in userId :int, in topic :string , in caption :string , in fileName :string , in document :byte [*], in crlimit :int):void
add_review (in uid :int, in postID :int, in content :string):void
delete_post (in postID :int):void
delete_review (in reviewID :int):void
delete_user (in userID :int):void

Figure 2: Class Diagram for Peer review tracking system (Part - 2)

Task 2: Activity diagram

Activity Diagram is used to show behavior of system and provides information on how the system accomplish certain tasks, how the control flows, and what activity takes place for particular scenario or task. In other words, it illustrates the flow of control and activities carries out by system step by step making a path of flow for a specific task.

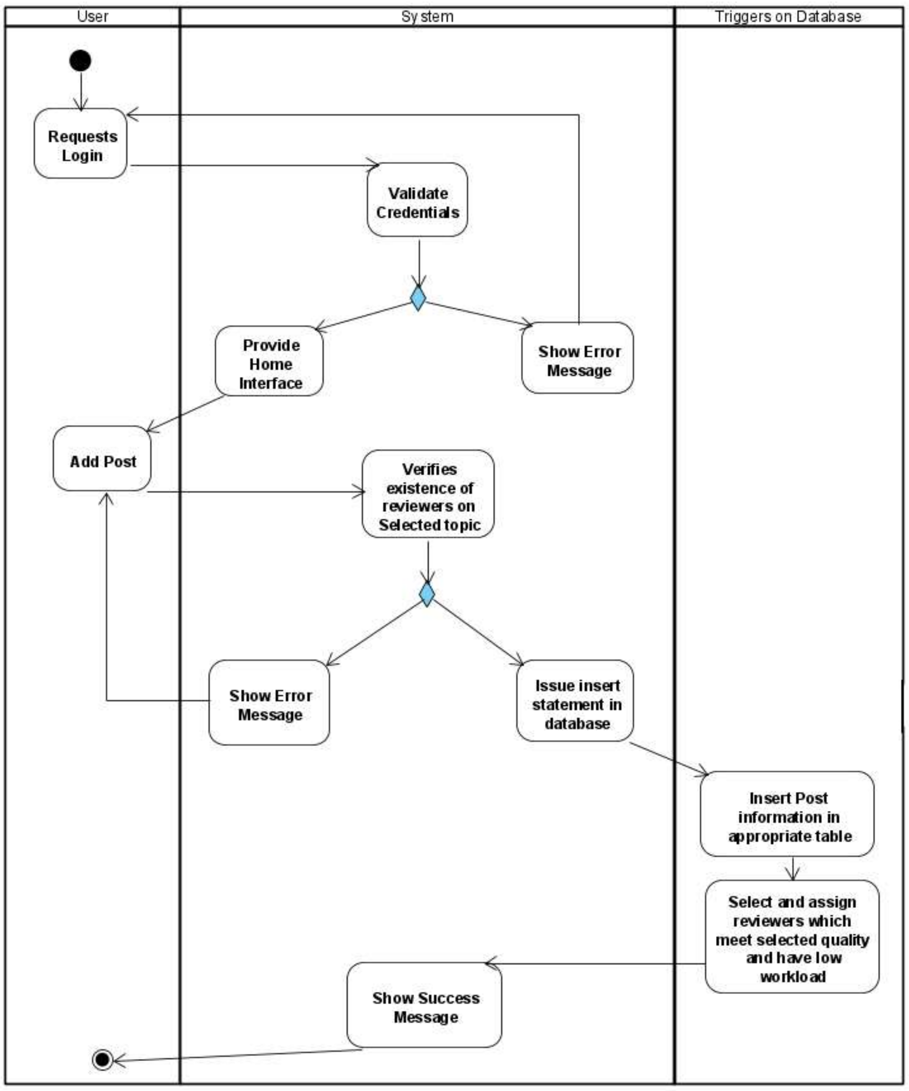
User
System
Triggers on Database
Requests Login
Validate Credentials
Provide Home Interface
Show Error Message
Add Post
Verifies existence of reviewers on Selected topic
Show Error Message
Issue insert statement in database
Insert Post information in appropriate table
Select and assign reviewers which meet selected quality and have low workload
Show Success Message

Figure 3: Activity diagram for the process of assigning peer reviewers a piece of work

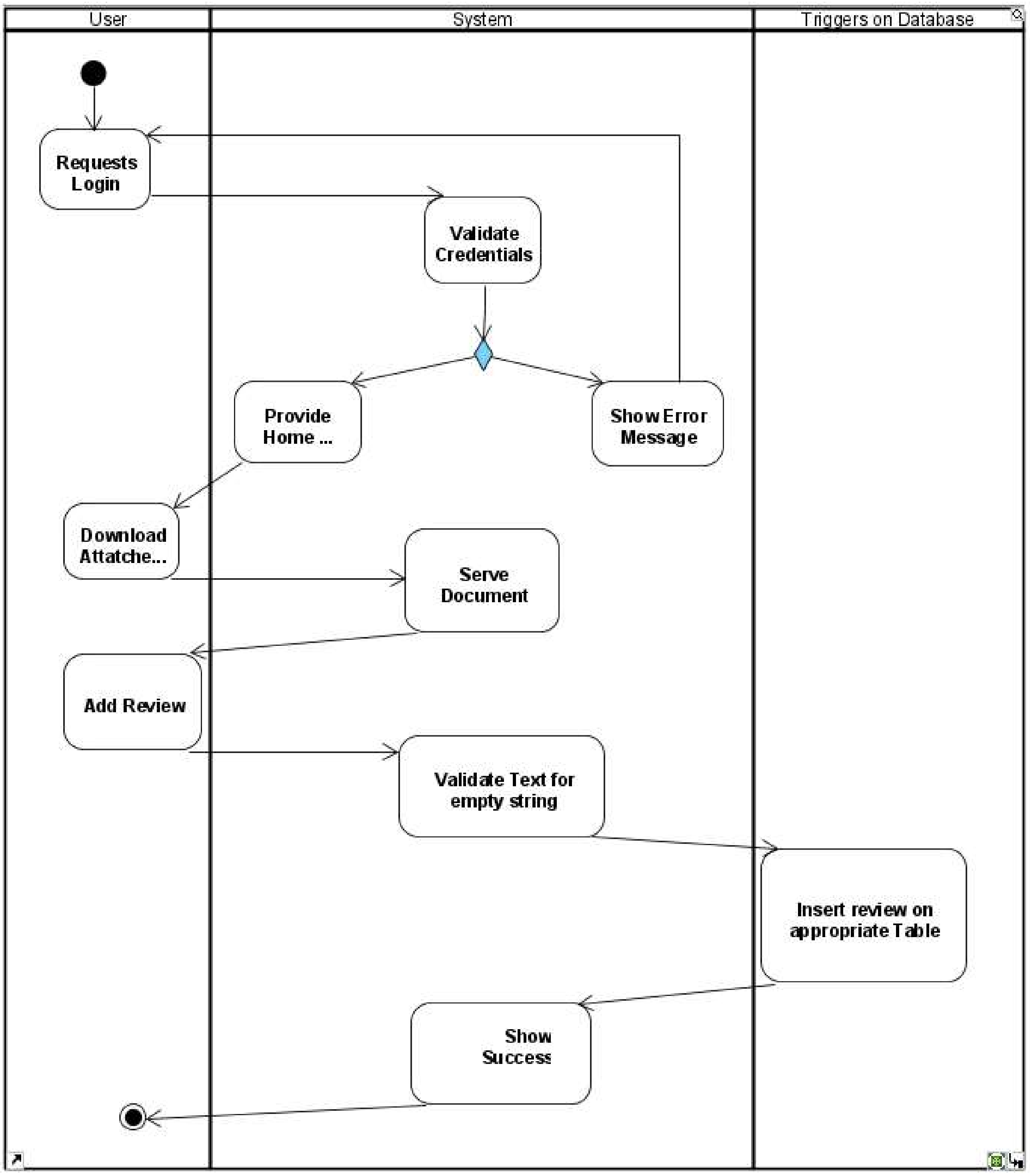

Figure 4: Activity diagram for the process of reviewing peer author's work

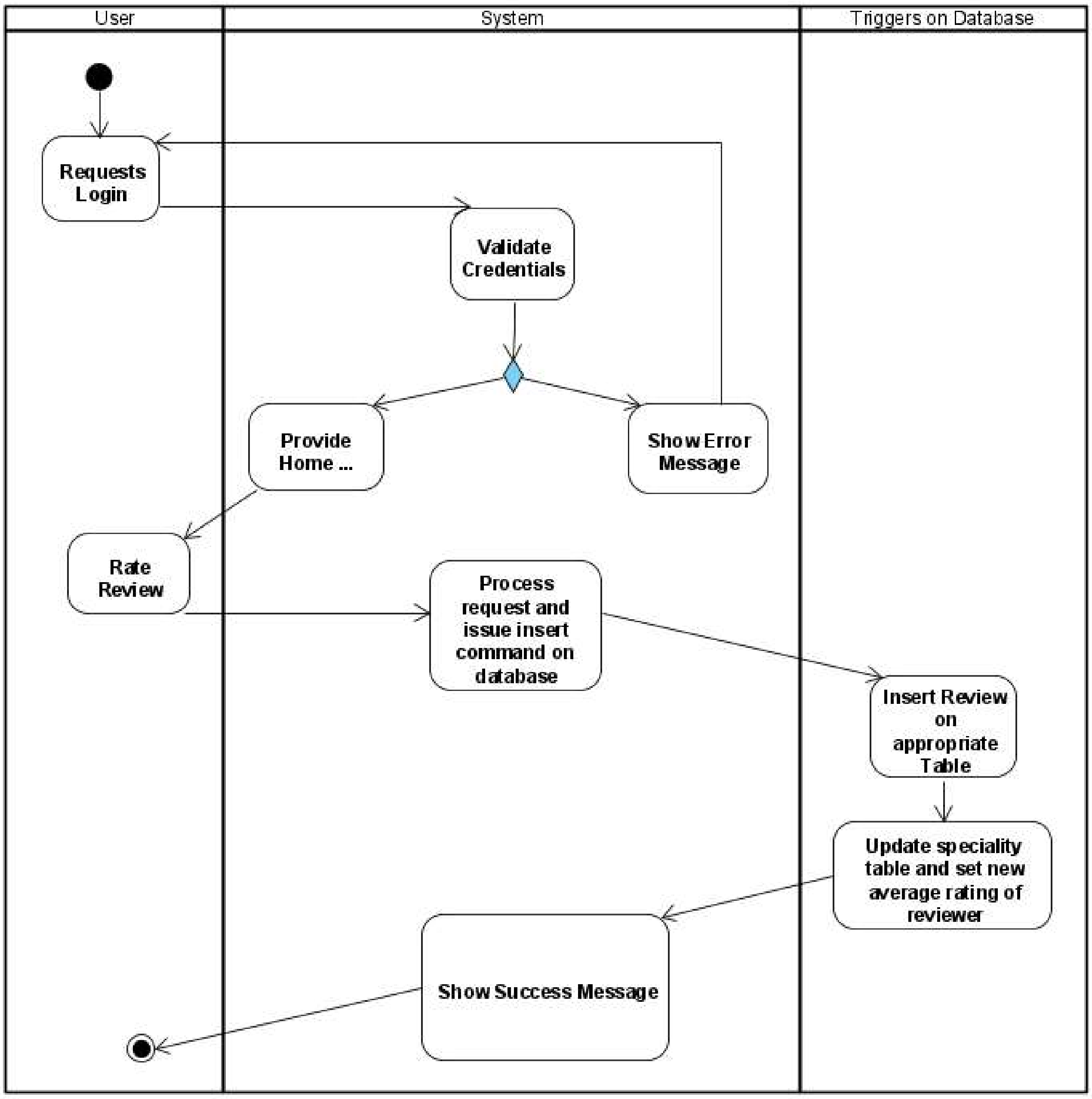
User
System
Triggers on Database
Requests Login
Validate Credentials
Provide Home ...
Show Error Message
Rate Review
Process request and issue insert command on database
Insert Review on appropriate Table
Update speciality table and set new average rating of reviewer
Show Success Message

Figure 5: Activity diagram for the process of rating peer reviewer's review

Task 3 Use Case diagram

Use-case can be described as the diagram to show workflow of the system from user's (Actor's) perspective. As it is used to show possible action that can be performed by actor to system and necessary tasks that systems perform, it also builds a scratch or structure for implementing classes in the system. It basically shows the set of actions that a system should perform or a user can access.

Below is the Use case diagram for Peer review tracking system:

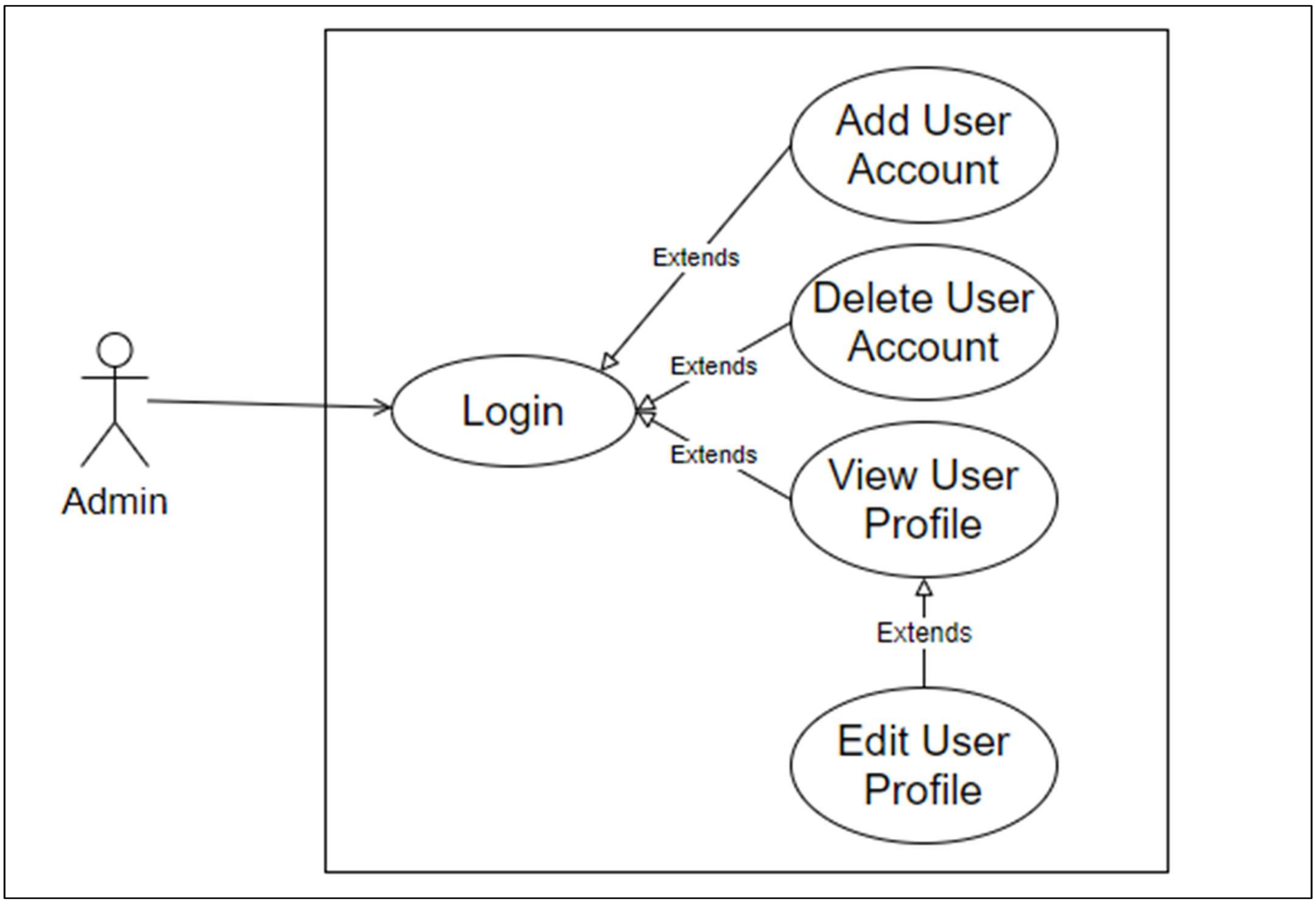

Figure 6: Use Case Diagram for Peer review tracking system (Admin user's scenario)

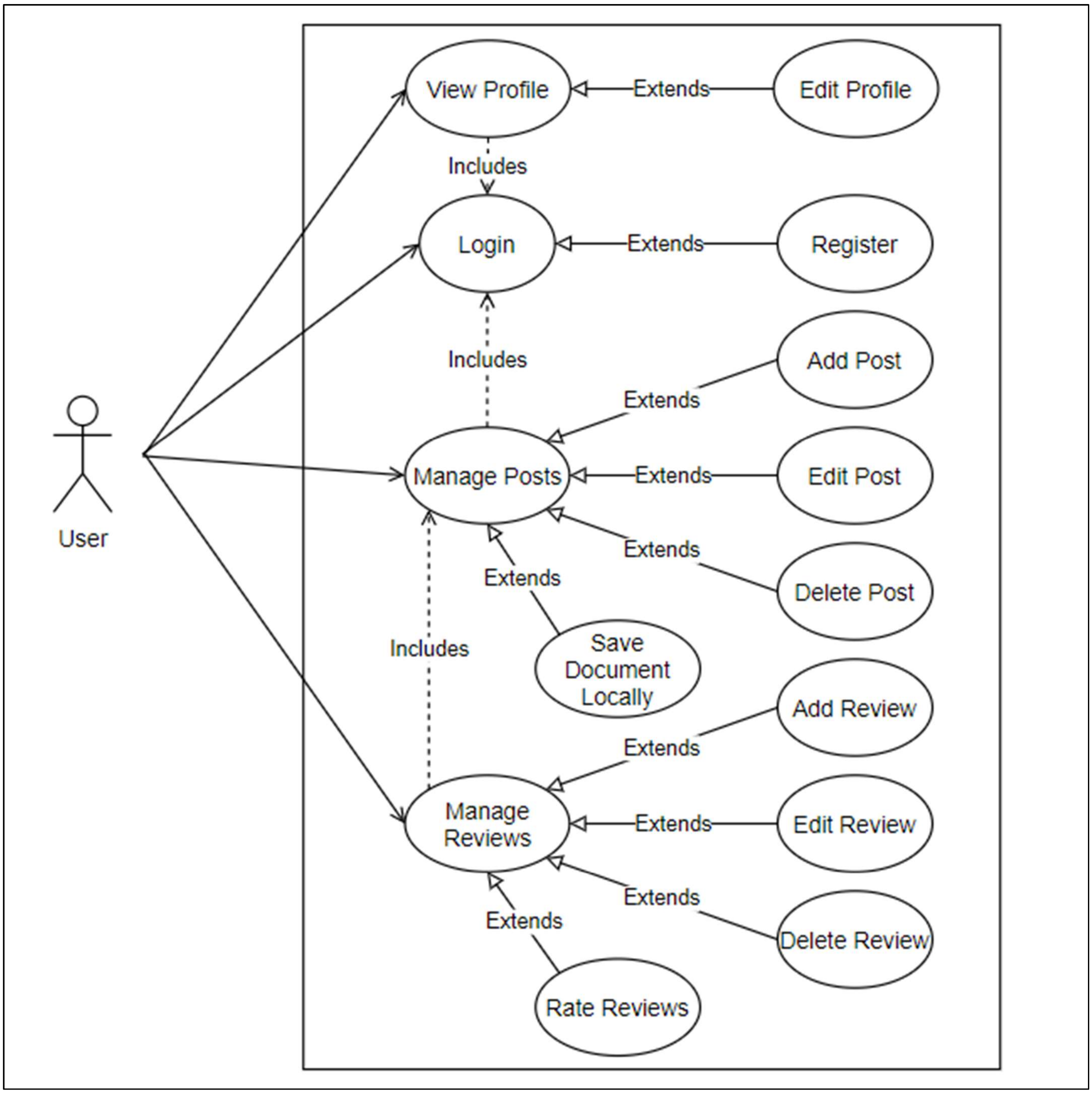

Figure 7: Use Case Diagram for Peer review tracking system (Normal user's scenario)

Task 4: Code architecture

This Project has been developed using Object-Oriented Programming Language C#. All main features of OOP have been implemented in this project. This project uses appropriate level of coupling and cohesion, along with the concept of inheritance, encapsulation and polymorphism. The relation and structure/architecture of codes and description of modules are given below.

Modules:

- Register and Login: Group of classes responsible for user's registration and login
- Post and Admin: Group of classes responsible for Admin's actions and posts management
- Database: Class responsible for database operation management throughout project's runtime
- Others: Other miscellaneous classes.

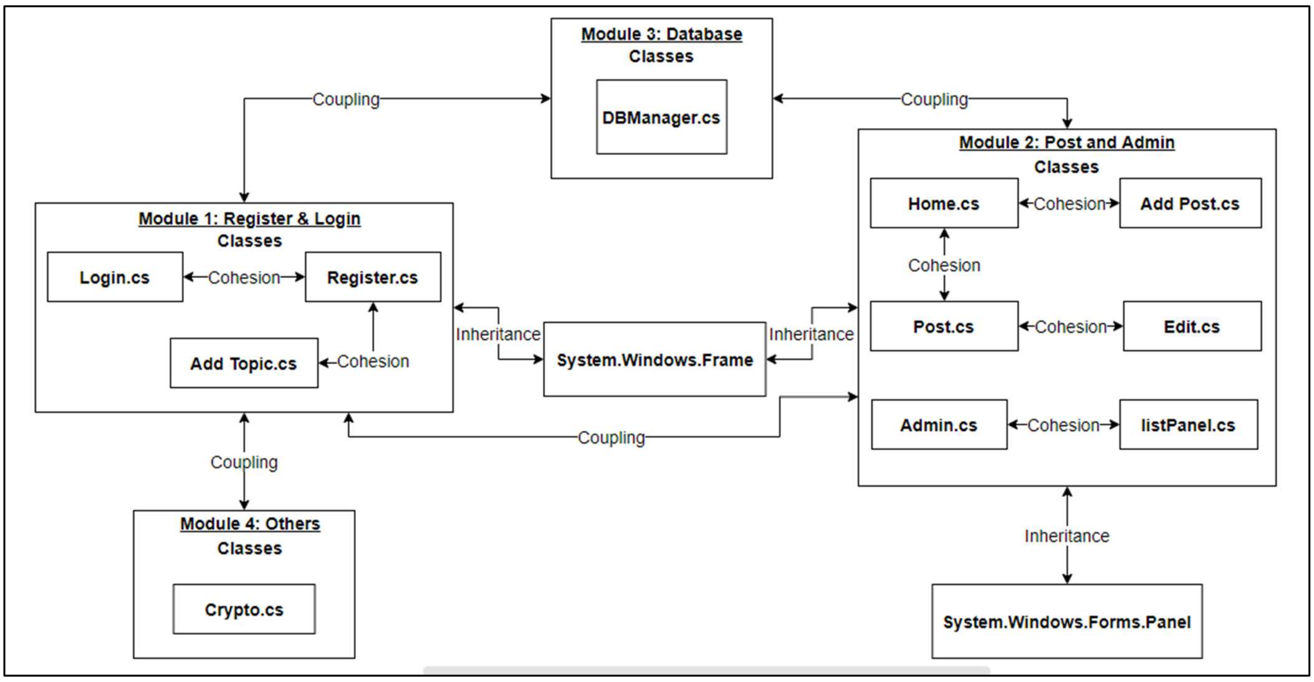

Figure 8: Code Architecture for peer review tracking system

Task 5: System implementation

- Program.cs

```
using System;
using System.Windows.Forms;

namespace rating
{
    static class Program
    {
        /// <summary>
        ///  The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.SetHighDpiMode(HighDpiMode.SystemAware);
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Frames.Login());
        }
    }
}
```

- listPanel.cs

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace rating.Custom_Widgets
{
    class listPanel : Panel
    {
        private readonly Label label1 = new Label();
        private readonly Label label2 = new Label();
        private readonly Button button1 = new Button();
        private readonly Button button2 = new Button();
        public string name;
        public int uid;
        public Frames.Admin main_window;
        public listPanel(int sn, string nme, int id, Frames.Admin main_window)
        {
            name = nme;
            uid = id;
            this.main_window = main_window;

            this.BackColor = System.Drawing.SystemColors.GradientActiveCaption;
            this.Controls.Add(this.button1);
            this.Controls.Add(this.button2);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.label2);
            this.Size = new System.Drawing.Size(758, 40);
            this.TabIndex = 0;
            this.button1.BackColor = System.Drawing.Color.Teal;
            this.button1.Font = new System.Drawing.Font("Segoe UI Semibold", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);
            this.button1.Location = new System.Drawing.Point(478, 5);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(90, 30);
            this.button1.TabIndex = 6;
            this.button1.Text = "Edit";
            this.button1.UseVisualStyleBackColor = false;
            this.button1.Click += new System.EventHandler(this.button1_Click);

            this.button2.BackColor = System.Drawing.Color.Crimson;
            this.button2.Font = new System.Drawing.Font("Segoe UI Semibold", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);
            this.button2.Location = new System.Drawing.Point(580, 5);
            this.button2.Name = "button2";
            this.button2.Size = new System.Drawing.Size(100, 30);
            this.button2.TabIndex = 6;
            this.button2.Text = "Delete";
            this.button2.UseVisualStyleBackColor = false;
            this.button2.Click += new System.EventHandler(this.button2_Click);

            this.label1.AutoSize = true;
            this.label1.Font = new System.Drawing.Font("Segoe UI Semibold", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);
            this.label1.Location = new System.Drawing.Point(153, 10);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(44, 17);
            this.label1.TabIndex = 1;
            this.label1.Text = nme;

            this.label2.AutoSize = true;
            this.label2.Font = new System.Drawing.Font("Segoe UI Semibold", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);
            this.label2.Location = new System.Drawing.Point(30, 10);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(28, 17);
            this.label2.TabIndex = 0;
            this.label2.Text = sn.ToString();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            string sql = "SELECT username FROM users WHERE userID=" + uid;
            DBManager dbm = new DBManager();
            List<Dictionary<string, object>> r = dbm.execute(sql);
            Dictionary<string, object> row = r[0];
            string username = row["username"].ToString();
            sql = "SELECT topics.topic as topic FROM reviewers_speciality INNER JOIN topics ON
reviewers_speciality.topicID = topics.topicID WHERE userID = " + uid;
            r = dbm.execute(sql);
            string speciality = "";
            foreach (Dictionary<string, object> a in r)
            {
                speciality = speciality + a["topic"].ToString() + "; ";
            }
            Frames.Register register = new Frames.Register(uid, name, username, speciality);
            register.Closed += main_window.button2_Click;
            register.Show();
        }
        private void button2_Click(object sender, EventArgs e)
        {
            var confirmResult = MessageBox.Show("Are you sure to delete this user??",
                                 "Confirm!!",
                                 MessageBoxButtons.YesNo);
            if (confirmResult == DialogResult.Yes)
            {
                DBManager dbm = new DBManager();
                dbm.delete_user(uid);
                main_window.button2_Click(this, new EventArgs());
            }
        }
    }
}
```

- Post.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Runtime.InteropServices;
using System.Windows.Forms;

namespace rating.Custom_Widgets
{
    class Post : Panel
    {
        public LinkLabel linkLabel1 = new LinkLabel();
        public Label label1 = new Label();
        public Label label2 = new Label();
        public TextBox textBox1 = new TextBox();
        public Button button1 = new Button();
        public Button button2 = new Button();

        public int uid;
        public bool isOwnPost = true;
        public string Short;
        public int postid;
        public string filename;
        public Frames.Home main_window;
        System.Drawing.Color color1;
        System.Drawing.Color color2;

        private const int EM_GETLINECOUNT = 0xba;
        [DllImport("user32", EntryPoint = "SendMessageA", CharSet = CharSet.Ansi, SetLastError =
true, ExactSpelling = true)]
        private static extern int SendMessage(int hwnd, int wMsg, int wParam, int lParam);

        public Post(int uid, Dictionary<string,object> row, string Short, Frames.Home main_window)
        {
            this.uid = uid;
            this.Short = Short;
            this.postid = Convert.ToInt32(row["postID"]);
            if (uid != Convert.ToInt32(row["userID"]))
                isOwnPost = false;
            if (isOwnPost)
            {
                color1 = System.Drawing.SystemColors.GradientActiveCaption;
                color2 = System.Drawing.SystemColors.GradientInactiveCaption;
            }
            else
            {
                color2 = System.Drawing.SystemColors.GradientActiveCaption;
                color1 = System.Drawing.SystemColors.GradientInactiveCaption;
            }
            this.filename = row["document_name"].ToString();
            this.main_window = main_window;

            this.BackColor = color1;
            this.Controls.Add(this.button1);
            this.Controls.Add(this.button2);
            this.Controls.Add(this.label2);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.textBox1);
            this.Controls.Add(this.linkLabel1);
            this.Size = new System.Drawing.Size(584, 307);

            this.button1.BackColor = System.Drawing.Color.Teal;
            this.button1.Font = new System.Drawing.Font("Segoe UI Semibold", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);
            this.button1.Location = new System.Drawing.Point(469, 5);
            this.button1.Size = new System.Drawing.Size(44, 30);
            this.button1.Text = "Edit";
            this.button1.UseVisualStyleBackColor = false;
            this.button1.Click += new System.EventHandler(this.button1_Click);


            this.button2.BackColor = System.Drawing.Color.Crimson;
            this.button2.Font = new System.Drawing.Font("Segoe UI Semibold", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);
            this.button2.Location = new System.Drawing.Point(519, 5);
            this.button2.Size = new System.Drawing.Size(56, 30);
            this.button2.Text = "Delete";
            this.button2.UseVisualStyleBackColor = false;
            this.button2.Click += new System.EventHandler(this.button2_Click);
            this.button2.Name = Convert.ToInt32(row["postID"]).ToString();


            this.label2.AutoSize = true;
            this.label2.Font = new System.Drawing.Font("Segoe UI Semibold", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);
            this.label2.Location = new System.Drawing.Point(2, 10);
            this.label2.MaximumSize = new System.Drawing.Size(580, 17);
            this.label2.MinimumSize = new System.Drawing.Size(580, 17);
            this.label2.Size = new System.Drawing.Size(580, 17);
            DateTime dt = Convert.ToDateTime(row["postedON"]);
            this.label2.Text = dt.ToString("ddd dd/MM/yyyy, hh:mm tt");
            this.label2.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;

            this.label1.AutoSize = true;
            this.label1.Font = new System.Drawing.Font("Segoe UI Bold", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);
            this.label1.Location = new System.Drawing.Point(9, 30);
            this.label1.Size = new System.Drawing.Size(34, 17);
            this.textBox1.Font = new System.Drawing.Font("Segoe UI Semibold", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);
            this.textBox1.Location = new System.Drawing.Point(19, 50);
            this.textBox1.MaxLength = 100000;
            this.textBox1.Multiline = true;
            this.textBox1.Size = new System.Drawing.Size(554, 31);
            this.textBox1.ReadOnly = true;
            this.textBox1.GotFocus += this.textBoxFocused;

            this.textBox1.BackColor = color1;
            this.textBox1.BorderStyle = BorderStyle.None;


            string caption = row["caption"].ToString().Trim();
            if (caption.Length == 0)
                caption = "No Caption...";
            this.textBox1.Text = caption;
            var numberOfLines = SendMessage(textBox1.Handle.ToInt32(), EM_GETLINECOUNT, 0,
0);
            int height = 0;
            if (numberOfLines != 1)
            {
                if (numberOfLines <= 10)
                {
                    height = (textBox1.Font.Height) * numberOfLines;
                    this.textBox1.Height = height;
                }
                else
                {
                    height = (textBox1.Font.Height) * 10;
                    this.textBox1.Height = height;
                    this.textBox1.WordWrap = true;
                    this.textBox1.ScrollBars = ScrollBars.Vertical;
                }
            }
            else
            {
                height = textBox1.Height;
            }

            int tempHeight = 50 + height + 5;

            this.linkLabel1.AutoSize = false;
            this.linkLabel1.Location = new System.Drawing.Point(2, tempHeight);
            this.linkLabel1.MaximumSize = new System.Drawing.Size(580, 15);
            this.linkLabel1.MinimumSize = new System.Drawing.Size(580, 15);
            this.linkLabel1.Size = new System.Drawing.Size(580, 15);
            this.linkLabel1.TabStop = true;
            this.linkLabel1.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
            this.linkLabel1.LinkClicked += new
LinkLabelLinkClickedEventHandler(this.linkLabel1_LinkClicked);
            this.linkLabel1.Text = row["document_name"].ToString().Trim();
            this.linkLabel1.Font = new System.Drawing.Font("Segoe UI Bold", 8F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);

            tempHeight = tempHeight + 15 + 5;
            int comments_height = load_reviews(Convert.ToInt32(row["postID"]), this.uid, tempHeight);
```

```
        this.Height = tempHeight + comments_height + 5;

        if (isOwnPost)
        {
            this.label1.Text = "You:";
        }
        else
        {
            this.label1.Text = "Anonymous Author:";
            this.button1.Hide();
            this.button2.Hide();
        }
    }
    public int load_reviews(int postID, int userID, int startHeight)
    {
        Panel panel = new Panel();
        Button buttonA = new Button();

        panel.BackColor = color1;
        panel.BorderStyle = BorderStyle.FixedSingle;
        panel.Location = new System.Drawing.Point(38, startHeight);
        panel.Size = new System.Drawing.Size(504, 156);

        DBManager dbm = new DBManager();
        string sql = "SELECT * FROM reviews WHERE postID=" + postID;
        if (!isOwnPost)
            sql += " AND userID=" + userID;
        sql += " ORDER BY postedON " + Short;
        List<Dictionary<string, object>> results = dbm.execute(sql);
        int PanelHeight = 10;
        if (results.Count != 0)
        {
            foreach (Dictionary<string, object> row in results)
            {
                Panel panelA = new Panel();
                Label labelA = new Label();
                Label labelB = new Label();
                Label labelC = new Label();
                Label labelD = new Label();
                LinkLabel linkLabelA = new LinkLabel();
                LinkLabel linkLabelB = new LinkLabel();
                TextBox textBoxA = new TextBox();
                ComboBox comboBoxA = new ComboBox();

                panelA.BackColor = color2;
                panelA.Controls.Add(labelA);
                panelA.Controls.Add(linkLabelA);
                panelA.Controls.Add(linkLabelB);
                panelA.Controls.Add(textBoxA);
                panelA.Controls.Add(labelB);
                panelA.Controls.Add(labelC);
                panelA.Controls.Add(labelD);
                panelA.Controls.Add(comboBoxA);
                panelA.Location = new System.Drawing.Point(17, PanelHeight);

                panelA.Size = new System.Drawing.Size(469, 94);

                labelA.AutoSize = true;
                labelA.Location = new System.Drawing.Point(9, 8);
                labelA.Size = new System.Drawing.Size(125, 15);
                labelA.Font = new System.Drawing.Font("Segoe UI Bold", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);

                linkLabelA.AutoSize = true;
                linkLabelA.Location = new System.Drawing.Point(423, 8);
                linkLabelA.Size = new System.Drawing.Size(27, 15);
                linkLabelA.TabStop = true;
                linkLabelA.Text = "Edit";
                linkLabelA.LinkClicked += new
LinkLabelLinkClickedEventHandler(this.linkLabelA_LinkClicked);
                linkLabelA.Name = Convert.ToInt32(row["reviewID"]).ToString();
                linkLabelA.Font = new System.Drawing.Font("Segoe UI Bold", 8F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);

                linkLabelB.AutoSize = true;
                linkLabelB.Location = new System.Drawing.Point(380, 8);
                linkLabelB.Size = new System.Drawing.Size(25, 15);
                linkLabelB.TabStop = true;
                linkLabelB.Text = "Delete";
                linkLabelB.LinkClicked += new
LinkLabelLinkClickedEventHandler(this.linkLabelB_LinkClicked);
                linkLabelB.Name = Convert.ToInt32(row["reviewID"]).ToString();
                linkLabelB.LinkColor = System.Drawing.Color.Crimson;
                linkLabelB.Font = new System.Drawing.Font("Segoe UI Bold", 8F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);


                textBoxA.Font = new System.Drawing.Font("Segoe UI Semibold", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);
                textBoxA.Location = new System.Drawing.Point(19, 25);
                textBoxA.MaxLength = 100000;
                textBoxA.Multiline = true;
                textBoxA.Size = new System.Drawing.Size(431, 31);
                textBoxA.ReadOnly = true;
                textBoxA.GotFocus += this.textBoxFocused;
                string review = row["content"].ToString().Trim();
                textBoxA.Text = review;
                textBoxA.BackColor = color2;
                textBoxA.BorderStyle = BorderStyle.None;


                var numberOfLines = SendMessage(textBoxA.Handle.ToInt32(),
EM_GETLINECOUNT, 0, 0);
                int height = 0;
                if (numberOfLines != 1)
                {
                    if (numberOfLines <= 10)
                    {
                        height = (textBoxA.Font.Height) * numberOfLines;
                        textBoxA.Height = height;
                    }
                    else
                    {
                        height = (textBoxA.Font.Height) * 10;
                        textBoxA.Height = height;
                        textBoxA.WordWrap = true;
                        textBoxA.ScrollBars = ScrollBars.Vertical;
                    }
```

```
            }
            else
            {
                height = textBoxA.Height;
            }

            int tempHeight = 25 + height + 5;

            labelB.AutoSize = true;
            labelB.Font = new System.Drawing.Font("Segoe UI Semibold", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);
            labelB.Location = new System.Drawing.Point(9, tempHeight + 5);
            labelB.Size = new System.Drawing.Size(89, 13);
            labelB.Text = "Commented on:";

            labelC.AutoSize = true;
            labelC.Font = new System.Drawing.Font("Segoe UI Semibold", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);
            labelC.Location = new System.Drawing.Point(100, tempHeight + 5);
            labelC.Size = new System.Drawing.Size(133, 13);

            DateTime dt = Convert.ToDateTime(row["postedON"]);
            labelC.Text = dt.ToString("ddd dd/MM/yyyy, hh:mm tt");

            labelD.AutoSize = true;
            labelD.Location = new System.Drawing.Point(319, tempHeight);
            labelD.Size = new System.Drawing.Size(66, 15);
            labelD.Text = "Usefulness:";

            comboBoxA.FormattingEnabled = true;
            comboBoxA.Location = new System.Drawing.Point(391, tempHeight - 4);
            comboBoxA.Size = new System.Drawing.Size(59, 23);
            comboBoxA.DropDownStyle = ComboBoxStyle.DropDownList;
            comboBoxA.Name = Convert.ToInt32(row["reviewID"]).ToString();
            comboBoxA.BackColor = color2;

            tempHeight = tempHeight + 19 + 8;

            panelA.Height = tempHeight;
            int rate;
            try
            {
                rate = Convert.ToInt32(row["rate"]);
                for (int i = 0; i <= 10; i++)
                {
                    comboBoxA.Items.Add(i.ToString());

                }
                comboBoxA.SelectedIndex = comboBoxA.FindStringExact(rate.ToString());
            }
            catch (Exception)
            {
                for (int i = 0; i <= 10; i++)
                {
                    comboBoxA.Items.Add(i.ToString());
                }
            }
            if (isOwnPost)
            {
                labelA.Text = "Anonymous Reviewer:";
                linkLabelA.Hide();
                linkLabelB.Hide();
            }
            else
            {
                labelA.Text = "You:";
                comboBoxA.Enabled = false;
            }

            comboBoxA.SelectedIndexChanged += new
System.EventHandler(this.comboBoxA_SelectedIndexChanged);

            PanelHeight += tempHeight + 5;
            panel.Controls.Add(panelA);
        }
    }
    else
    {
        if (isOwnPost)
        {
            panel.Size = new System.Drawing.Size(0, 0);
        }
    }
    if (!isOwnPost)
    {
        buttonA.BackColor = System.Drawing.Color.Teal;
        buttonA.Font = new System.Drawing.Font("Segoe UI Semibold", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);
        buttonA.Location = new System.Drawing.Point(385, PanelHeight);
        buttonA.Size = new System.Drawing.Size(101, 30);
        buttonA.Text = "+ Add Review";
        buttonA.UseVisualStyleBackColor = false;
        buttonA.Click += new System.EventHandler(this.buttonA_Click);

        PanelHeight = PanelHeight + 30 + 10;
        panel.Height = PanelHeight;
        panel.Controls.Add(buttonA);
    }
    else
    {
        PanelHeight += 5;
        panel.Height = PanelHeight;
    }

    this.Controls.Add(panel);

    return PanelHeight;
}
private void button1_Click(object sender, EventArgs e)
{
    Frames.Edit edit = new Frames.Edit(postid, true);
    edit.Show();
    edit.Closed += main_window.button1_Click;
}
private void button2_Click(object sender, EventArgs e)
{
    var confirmResult = MessageBox.Show("Are you sure to delete this post??",
                 "Confirm!!",
                          MessageBoxButtons.YesNo);
            if (confirmResult == DialogResult.Yes)
            {
                DBManager dbm = new DBManager();
                dbm.delete_post(Convert.ToInt32(((Button)sender).Name));
                main_window.button1_Click(this, new EventArgs());
            }
        }
        private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
        {
            SaveFileDialog SaveFileDialog1 = new SaveFileDialog();
            SaveFileDialog1.InitialDirectory = @"C:\";
            SaveFileDialog1.RestoreDirectory = true;
            SaveFileDialog1.Title = "Save Document";
            SaveFileDialog1.DefaultExt = "pdf";
            SaveFileDialog1.Filter = "PDF Documents | *.pdf";
            SaveFileDialog1.FileName = filename;
            if (SaveFileDialog1.ShowDialog() == DialogResult.OK)
            {
                byte[] file;
                DBManager dbm = new DBManager();
                string sql = "SELECT document FROM posts WHERE postID=" + postid;
                List<Dictionary<string, object>> docRes = dbm.execute(sql);
                file = (byte[])docRes[0]["document"];
                File.WriteAllBytes(SaveFileDialog1.FileName, file);
                MessageBox.Show("Document Saved.");
            }
        }
        private void linkLabelA_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
        {
            Frames.Edit edit = new Frames.Edit(Convert.ToInt32(((LinkLabel)sender).Name), false);

            edit.Show();
            edit.Closed += main_window.button1_Click;
        }
        private void linkLabelB_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
        {
            var confirmResult = MessageBox.Show("Are you sure to delete this review??",
                          "Confirm!!",
                          MessageBoxButtons.YesNo);
            if (confirmResult == DialogResult.Yes)
            {
                DBManager dbm = new DBManager();
                dbm.delete_review(Convert.ToInt32(((LinkLabel)sender).Name));
                main_window.button1_Click(this, new EventArgs());
            }
        }
        private void comboBoxA_SelectedIndexChanged(object sender, EventArgs e)
        {
            var confirmResult = MessageBox.Show("Are you sure to Rate usefulness of this review with
    " + ((ComboBox)sender).Text.Trim() + " Points??",
                          "Confirm!!",
                          MessageBoxButtons.YesNo);
            if (confirmResult == DialogResult.Yes)
            {
                DBManager dbm = new DBManager();
                string sql = "UPDATE reviews SET rate=" + ((ComboBox)sender).Text.Trim() + " WHERE
    reviewID=" + ((ComboBox)sender).Name.Trim();
                dbm.execute(sql);
                main_window.button1_Click(this, new EventArgs());
            }
        }
        private void buttonA_Click(object sender, EventArgs e)
        {
            Frames.Edit edit = new Frames.Edit(uid, postid);
            edit.Show();
            edit.Closed += main_window.button1_Click;
        }
        private void textBoxFocused(object sender, EventArgs e)
        {
            this.Focus();
        }
    }
}
```

- Crypto.cs

```
using System;
using System.Collections.Generic;
using System.Security.Cryptography;
using System.Text;

namespace rating.Modules
{
    class Crypto
    {
        public static string Sha256(string str)
        {
            SHA256 sha256Hash = SHA256.Create();
            byte[] bytes = sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(str));
            StringBuilder builder = new StringBuilder();
            for (int i = 0; i < bytes.Length; i++)
            {
                builder.Append(bytes[i].ToString("x2"));
            }
            return builder.ToString();
        }
    }
}
```

- DBManager.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data.SqlClient;
using System.Data.SqlTypes;
using System.Windows.Forms;

namespace rating
{
    class DBManager
    {
        public int max_reviewers_to_allocate_per_post = 5;
        public SqlConnection conn;
        public DBManager()
        {
            string connectionString = "Server=DESKTOP-5B5II3T\\SQLEXPRESS;Initial 
    Catalog=rating;Integrated Security=SSPI;";
            conn = new SqlConnection(connectionString);
            if (table_not_exists("topics"))
            {
                string s = "CREATE TABLE topics (" +
                    "topicID int IDENTITY(1,1) PRIMARY KEY," +
                    "topic varchar(50) NOT NULL UNIQUE" +
                    ");";
                execute(s);
            }
            if (table_not_exists("users"))
            {
                string s = "CREATE TABLE users (" +
                    "userID int IDENTITY(1,1) PRIMARY KEY," +
                    "Name varchar(255) NOT NULL," +
                    "username varchar(50) NOT NULL UNIQUE," +
                    "psword varchar(255) NOT NULL," +
                    "user_role int NOT NULL," +
                    "works int NOT NULL DEFAULT 0" +
                    ");";
                execute(s);
            }
            if (table_not_exists("reviewers_speciality"))
            {
                string s = "CREATE TABLE reviewers_speciality (" +
                    "userID int NOT NULL," +
                    "topicID int NOT NULL," +
                    "AVGrate int NOT NULL DEFAULT 0," +
                    "FOREIGN KEY(userID) REFERENCES users(userID) ON DELETE CASCADE," +
                    "FOREIGN KEY(topicID) REFERENCES topics(topicID) ON DELETE CASCADE" +
                    ");";
                execute(s);
            }
            if (table_not_exists("posts"))
            {
                string s = "CREATE TABLE posts (" +
                    "postID int IDENTITY(1,1) PRIMARY KEY," +
                    "userID int NOT NULL," +
                    "topicID int NOT NULL," +
                    "postedON datetime NOT NULL DEFAULT CURRENT_TIMESTAMP," +
                    "caption VARCHAR(MAX) NULL," +
                    "document_name VARCHAR(255) NOT NULL," +
                    "document VARBINARY(MAX) NOT NULL," +
                    "creditLimit int NOT NULL DEFAULT 0," +
                    "FOREIGN KEY(userID) REFERENCES users(userID) ON DELETE CASCADE," +
                    "FOREIGN KEY(topicID) REFERENCES topics(topicID) ON DELETE CASCADE" +
                    ");";
                execute(s);
            }
            if (table_not_exists("assignments_posts"))
            {
                string s = "CREATE TABLE assignments_posts (" +
                    "postID int NOT NULL," +
                    "userID int NOT NULL," +
                "FOREIGN KEY(postID) REFERENCES posts(postID) ON DELETE CASCADE," +
                "FOREIGN KEY(userID) REFERENCES users(userID)" +
                ");";
            execute(s);
        }
        if (table_not_exists("reviews"))
        {
            string s = "CREATE TABLE reviews (" +
                "reviewID int IDENTITY(1,1) PRIMARY KEY," +
                "postID int NOT NULL," +
                "userID int NOT NULL," +
                "postedON datetime NOT NULL DEFAULT CURRENT_TIMESTAMP," +
                "content VARCHAR(MAX) NULL," +
                "rate int NULL," +
                "FOREIGN KEY(postID) REFERENCES posts(postID) ON DELETE CASCADE," +
                "FOREIGN KEY(userID) REFERENCES users(userID)" +
                ");";
            execute(s);
        }
        string sql = "IF NOT EXISTS (SELECT * FROM sys.triggers WHERE object_id =
OBJECT_ID(N'[dbo].[trgAssignments_postsInsert]')) " +
            "EXEC dbo.sp_executesql @statement = N' " +
            "CREATE TRIGGER trgAssignments_postsInsert " +
            "ON assignments_posts " +
            "AFTER INSERT " +
            "AS " +
            "BEGIN " +
            "UPDATE users SET works = works + 1 WHERE userID IN (SELECT userID FROM
inserted) " +
            "END'";
        execute(sql);
        sql = "IF NOT EXISTS (SELECT * FROM sys.triggers WHERE object_id =
OBJECT_ID(N'[dbo].[trgPostsInsert]')) " +
            "EXEC dbo.sp_executesql @statement = N' " +
            "CREATE TRIGGER trgPostsInsert " +
            "ON posts " +
            "AFTER INSERT " +
            "AS " +
            "BEGIN " +
            "INSERT INTO assignments_posts(postID, userID) " +
            "SELECT TOP " + max_reviewers_to_allocate_per_post + " tbl1.postID, tbl1.userID " +
            "FROM( " +
            "SELECT p.postID as postID, reviewers_speciality.userID as userID " +
            "FROM reviewers_speciality " +
            "INNER JOIN " +
            "inserted p " +
            "ON " +
            "reviewers_speciality.AVGrate >= p.creditLimit AND reviewers_speciality.topicID =
p.topicID " +
            "WHERE " +
            "reviewers_speciality.userID != p.userID) as tbl1 " +
            "INNER JOIN " +
            "users " +
            "ON " +
            "tbl1.userID = users.userID " +
            "ORDER BY users.works " +
            "END'";
        execute(sql);
```

```
            sql = "IF NOT EXISTS (SELECT * FROM sys.triggers WHERE object_id =
OBJECT_ID(N'[dbo].[trgReviewsInsertUpdate]')) " +
                "EXEC dbo.sp_executesql @statement = N' " +
                "CREATE TRIGGER trgReviewsInsertUpdate " +
                "ON reviews " +
                "AFTER UPDATE, INSERT " +
                "AS " +
                "DECLARE @topicid int " +
                "DECLARE @userid int " +
                "DECLARE @avgrate int " +
                "SELECT @topicid = topicID FROM posts, inserted i WHERE posts.postID = i.postID " +
                "SELECT @userid = userid from inserted i " +
                "SELECT @avgrate = AVG(reviews.rate) FROM posts INNER JOIN reviews ON
posts.postID = reviews.postID WHERE(reviews.userID = @userid) AND(posts.topicID = @topicid) "
+
                "BEGIN " +
                "IF(@avgrate IS NOT NULL) " +
                "UPDATE reviewers_speciality set AVGrate = @avgrate WHERE(topicID = @topicid)
AND(userID = @userid); " +
                "END'";
            execute(sql);
            sql = "IF NOT EXISTS (SELECT * FROM sys.triggers WHERE object_id =
OBJECT_ID(N'[dbo].[trgReviewrs_SpecialInsertInstead]')) " +
                "EXEC dbo.sp_executesql @statement = N' " +
                "CREATE TRIGGER trgReviewrs_SpecialInsertInstead " +
                "ON reviewers_speciality " +
                "INSTEAD OF INSERT " +
                "AS " +
                "DECLARE @uid int " +
                "DECLARE @tid int " +
                "SELECT @uid = i.userID, @tid = i.topicID FROM inserted i " +
                "BEGIN " +
                "IF(NOT EXISTS(SELECT r.userID FROM reviewers_speciality r WHERE r.userID =
@uid AND r.topicID = @tid)) " +
                "INSERT INTO reviewers_speciality(userID, topicID) VALUES(@uid, @tid); " +
                "END'";
            execute(sql);
            sql = "IF NOT EXISTS (SELECT * FROM sys.triggers WHERE object_id =
OBJECT_ID(N'[dbo].[trgUsersDelete]')) " +
                "EXEC dbo.sp_executesql @statement = N' " +
                "CREATE TRIGGER trgUsersDelete " +
                "ON users " +
                "INSTEAD OF DELETE " +
                "AS " +
                "DECLARE @uid int " +
                "SELECT @uid = d.userID FROM deleted d " +
                "BEGIN " +
                "DELETE FROM assignments_posts WHERE userID = @uid; " +
                "DELETE FROM reviews WHERE userID = @uid; " +
                "DELETE FROM users WHERE userID = @uid; " +
                "END'";
            execute(sql);

        }
        public List<Dictionary<string, object>> execute(string sql, bool is_select = false)
        {
            List<Dictionary<string, object>> table = new List<Dictionary<string, object>>();
            if (sql.Substring(0,6).ToUpper() == "SELECT")
            {
                is_select = true;
            }

            conn.Open();
            if (!is_select)
            {
                try
                {
                    SqlCommand command = new SqlCommand(sql, conn);
                    command.ExecuteNonQuery();
                }
                catch (Exception E)
                {
                    MessageBox.Show(E.Message);
                }
                finally
                {
                    conn.Close();
                }
            }
            else
            {
                try
                {
                    SqlCommand command = new SqlCommand(sql, conn);
                    SqlDataReader reader = command.ExecuteReader();
                    List<string> names = new List<string>();
                    for (int i = 0; i < reader.FieldCount; i++)
                        names.Add(reader.GetName(i).ToString());
                    if (reader.HasRows)
                    {
                        while (reader.Read())
                        {
                            Dictionary<string, object> row = new Dictionary<string, object>();
                            for (int i = 0; i < reader.FieldCount; i++)
                                row.Add(names[i], reader[names[i]]);
                            table.Add(row);
                        }
                    }
                }catch(Exception E)
                {
                    MessageBox.Show(E.Message);
                }
                finally
                {
                    conn.Close();
                }
            }
            return table;
        }
        public bool table_not_exists(string tablename)
        {
            string sql = "IF (NOT EXISTS(SELECT * FROM INFORMATION_SCHEMA.TABLES
      WHERE TABLE_SCHEMA = 'dbo' AND  TABLE_NAME = '" + tablename + "')) BEGIN SELECT 'no'
      as result; END";
            List<Dictionary<string, object>> r = execute(sql, true);
            if (r.Count > 0)
                if (r[0]["result"].ToString() == "no")
                    return true;
                else
                    return false;
            else
                return false;
        }
        public void add_topic(string name)
        {
            string sql = "INSERT INTO topics(topic) VALUES('" + name + "')";
            execute(sql);
        }
        public void add_user(string name, string username, string password, int role, string
speciality="")
        {
            string sql = "INSERT INTO users(Name, username, psword, user_role) VALUES('" + name
+ "', '" + username + "', '" + password + "', " + role + ")";
            execute(sql);
            if (role != 0)
            {
                string[] specialities = speciality.Split("; ");
                sql = "SELECT userID FROM users WHERE username='" + username + "'";
                List<Dictionary<string, object>> r = execute(sql, true);
                int uid = Convert.ToInt32(r[0]["userID"]);
                foreach (string special in specialities)
                {
                    if (special != "")
                    {
                        sql = "SELECT topicID FROM topics WHERE topic = '" + special + "'";
                        r = execute(sql, true);
                        int tid = Convert.ToInt32(r[0]["topicID"]);
                        sql = "INSERT INTO reviewers_speciality(userID, topicID) VALUES(" + uid + "," + tid
+ ")";
                        execute(sql);
                    }
                }
            }
        }
        public void update_user(int uid, string name, string username, string password, int role, string
speciality = "")
        {
            string sql = "UPDATE users SET Name='" + name + "', username='" + username + "',
psword='" + password + "' WHERE userID=" + uid;
            execute(sql);

            if (role != 0)
            {
                string[] specialities = speciality.Split("; ");
                foreach (string special in specialities)
                {
                    if (special != "")
                    {
                        sql = "SELECT topicID FROM topics WHERE topic = '" + special + "'";
                        List<Dictionary<string, object>> r = execute(sql, true);
                        int tid = Convert.ToInt32(r[0]["topicID"]);
                        sql = "INSERT INTO reviewers_speciality(userID, topicID) VALUES(" + uid + "," + tid
+ ")";
                        execute(sql);
                    }
                }
            }
        }
        public void add_post(int userId, string topic, string caption, string fileName, byte[] document,
    int crlimit)
        {
            string sql = "SELECT topicID FROM topics WHERE topic='" + topic + "'";
            List<Dictionary<string, object>> r = execute(sql, true);
            int tid = Convert.ToInt32(r[0]["topicID"]);
            SqlCommand sc = new SqlCommand();
            sc.CommandText = "INSERT INTO posts(userID, topicID, caption, document_name,
    document, creditLimit) VALUES(@uid,@tid,@cpt,@dnm,@doc,@crl)";
            sc.Parameters.Add("@uid", System.Data.SqlDbType.Int).Value = userId;
            sc.Parameters.Add("@tid", System.Data.SqlDbType.Int).Value = tid;
            sc.Parameters.Add("@cpt", System.Data.SqlDbType.VarChar).Value = caption;
            sc.Parameters.Add("@dnm", System.Data.SqlDbType.VarChar).Value = fileName;
            sc.Parameters.Add("@doc", System.Data.SqlDbType.VarBinary).Value = document;
            sc.Parameters.Add("@crl", System.Data.SqlDbType.Int).Value = crlimit;
            conn.Open();
            sc.Connection = conn;
            sc.ExecuteNonQuery();
            conn.Close();
        }
        public void add_review(int uid, int postID, string content)
        {
            SqlCommand sc = new SqlCommand();
            sc.CommandText = "INSERT INTO reviews(postID, userID, content)
    VALUES(@pid,@uid,@content)";
            sc.Parameters.Add("@pid", System.Data.SqlDbType.Int).Value = postID;
            sc.Parameters.Add("@uid", System.Data.SqlDbType.Int).Value = uid;
            sc.Parameters.Add("@content", System.Data.SqlDbType.VarChar).Value = content;
            conn.Open();
            sc.Connection = conn;
            sc.ExecuteNonQuery();
            conn.Close();
        }
        public void delete_post(int postID)
        {
            string sql = "DELETE FROM posts WHERE postID=" + postID;
            execute(sql);
        }
        public void delete_review(int reviewID)
        {
            string sql = "DELETE FROM reviews WHERE reviewID=" + reviewID;
            execute(sql);
        }
        public void delete_user(int userID)
        {
            string sql = "DELETE FROM users WHERE userID=" + userID;
            execute(sql);
        }
      }
    }
```

- Add Post.cs

```
    using System;
    using System.Collections.Generic;
    using System.IO;
    using System.Windows.Forms;
```

```
namespace rating.Frames
{
    public partial class Add_Post : Form
    {
        public int uid;
        public Add_Post(int uid)
        {
            InitializeComponent();
            this.uid = uid;
            DBManager dbm = new DBManager();
            string sql = "SELECT topic from topics";
            List<Dictionary<string, object>> r = dbm.execute(sql);
            int i = 0;
            foreach(Dictionary<string, object> row in r)
            {
                comboBox1.Items.Insert(i, row["topic"].ToString());
                i++;
            }
        }

        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            comboBox2.Items.Clear();
            DBManager dbm = new DBManager();
            string sql = "SELECT topicID from topics WHERE topic='" + comboBox1.Text.Trim() + "'";
            List<Dictionary<string, object>> r = dbm.execute(sql);
            int topicID = Convert.ToInt32(r[0]["topicID"]);
            sql = "SELECT MAX(AVGrate) as maxrate FROM reviewers_speciality WHERE topicID=" +
topicID + " AND userID!="+uid;
            r = dbm.execute(sql);
            if (r.Count == 0 )
            {
                MessageBox.Show("Sorry, No reviewer available for this topic yet.");
                return;
            }
            try
            {
                Convert.ToInt32((r[0]["maxrate"]));
            }catch(Exception)
            {
                MessageBox.Show("Sorry, No reviewer available for this topic yet.");
                return;
            }
            int max = Convert.ToInt32(r[0]["maxrate"]);
            for (int i = 0; i <= max; i++)
                comboBox2.Items.Insert(i, i.ToString());
        }

        private void button2_Click(object sender, EventArgs e)
        {
            OpenFileDialog dialog = new OpenFileDialog();
            dialog.Filter = "PDF Documents | *.pdf";
            dialog.Multiselect = false;
            if (dialog.ShowDialog() == DialogResult.OK)
            {
                String path = dialog.FileName;
                textBox2.Text = path;
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
            string caption = textBox1.Text.Trim();
            string topic = comboBox1.Text.Trim();
            string mincredit = comboBox2.Text.Trim();
            string FilePath = textBox2.Text.Trim();
            if (topic == "")
            {
                MessageBox.Show("Select one topic.");
                comboBox1.Focus();
                return;
            }
            if (mincredit == "")
            {
                MessageBox.Show("Select minimun credit requirement.");
                comboBox2.Focus();
                return;
            }
            int mincred = Convert.ToInt32(mincredit);
            if (FilePath == "")
            {
                MessageBox.Show("File is required.");
                comboBox2.Focus();
                return;
            }
            string filename = Path.GetFileName(FilePath);
            if (filename.Length > 140)
            {
                MessageBox.Show("Filename cannot be longer than 140 characters. Please rename and
try again.");
                comboBox2.Focus();
                return;
            }
            FileStream fs = new FileStream(FilePath, FileMode.Open, FileAccess.Read);
            long length = new System.IO.FileInfo(FilePath).Length;
            if (length > 52428800)
            {
                MessageBox.Show("Filename size cannot be more than 50MB.");
                comboBox2.Focus();
                return;
            }
            decimal len = (decimal)((decimal)length / (decimal)1048576);
            len = Math.Ceiling(len * 100) / 100;

            BinaryReader br = new BinaryReader(fs);
            byte[] document = br.ReadBytes((Int32)fs.Length);
            br.Close();
            fs.Close();
            DBManager dbm = new DBManager();
            filename = filename.Remove(filename.Length - 4);
            dbm.add_post(uid, topic, caption, filename.ToLower().Trim() + " [" + len + "MB].pdf",
document, mincred);
            MessageBox.Show("Posted");
            this.Close();
        }
    }
}
```

- Add Topic.cs

```
using System;
using System.Collections.Generic;
using System.Text.RegularExpressions;
using System.Windows.Forms;

namespace rating.Frames
{
    public partial class Add_Topic : Form
    {
        public Add_Topic()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            string name = textBox1.Text.Trim();
            string namereg = @"^[a-zA-Z][a-zA-Z0-9 ]*$";
            if (!Regex.IsMatch(name, namereg))
            {
                MessageBox.Show("Invalid Topic Name!");
                textBox1.Focus();
                return;
            }
            DBManager dbm = new DBManager();
            if (!dbm.table_not_exists("topics"))
            {
                string sql = "SELECT * FROM topics where topic='" + name + "'";
                List<Dictionary<string, object>> r = dbm.execute(sql, true);
                if (r.Count != 0)
                {
                    MessageBox.Show("Topic Name already available!");
                    textBox1.Focus();
                    return;
                }
            }
            dbm.add_topic(name);
            MessageBox.Show("Added!");
            Close();
        }
    }

}
```

- Admin.cs

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace rating.Frames
{
    public partial class Admin : Form
    {
```

```
public int uid;
public Admin(int uid)
{
    InitializeComponent();

    panel2.HorizontalScroll.Maximum = 0;
    panel2.AutoScroll = false;
    panel2.VerticalScroll.Visible = false;
    panel2.AutoScroll = true;

    this.uid = uid;
    string sql = "SELECT * FROM users WHERE user_role=" + 1;
    DBManager dbm = new DBManager();
    List<Dictionary<string, object>> r = dbm.execute(sql);
    paint(r);
}

private void button1_Click(object sender, EventArgs e)
{
    Register reg = new Register(0);
    reg.Closed += this.button2_Click;
    reg.Show();
}

public void button2_Click(object sender, EventArgs e)
{
    string sql = "SELECT * FROM users WHERE user_role=" + 1;
    DBManager dbm = new DBManager();
    List<Dictionary<string, object>> r = dbm.execute(sql);
    paint(r);
}
public void paint(List<Dictionary<string, object>> r)
{
    panel2.Controls.Clear();
    panel2.Controls.Add(panel3);
    int i = 1;
    foreach (Dictionary<string, object> row in r)
    {
        int userID = Convert.ToInt32(row["userID"]);
        string name = row["Name"].ToString();
        Custom_Widgets.listPanel lp = new Custom_Widgets.listPanel(i, name, userID, this);
        lp.Location = new System.Drawing.Point(8, 55 + (i - 1) * 45);
        panel2.Controls.Add(lp);
        lp.Show();
        i++;
    }
    panel2.Refresh();
}

private void button3_Click(object sender, EventArgs e)
{
    search();
}

private void textBox1_TextChanged(object sender, EventArgs e)
{
    search();
}
public void search()
```

```
        {
            string searchString = textBox1.Text.Trim();
            searchString = searchString.Replace("'", "").Replace("\"", "").Replace("`", "").Replace("%",
"").Replace("?", "").Replace("_", "").Replace("*", "");

            string sql = "SELECT * FROM users WHERE user_role !=0 AND Name LIKE '%" +
searchString + "%'";
            DBManager dbm = new DBManager();
            List<Dictionary<string, object>> r = dbm.execute(sql);
            paint(r);
        }
    }
}
```

- Edit.cs

```
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Windows.Forms;

namespace rating.Frames
{
    public partial class Edit : Form
    {
        public bool isPost;
        public int id;
        public int postID;
        public bool isComment = false;

        public Edit(int id, int postID)
        {
            this.id = id;
            this.postID = postID;
            this.isComment = true;
            InitializeComponent();
            this.Text = "Add Review";
            this.label1.Text = "Add Review";
            this.label1.Location = new System.Drawing.Point(155, 18);
            this.label2.Text = "Review Text (Upto 100000 chars):";
            this.button1.Text = "+ Add";
        }
        public Edit(int id, bool isPost)
        {
            this.isPost = isPost;
            this.id = id;
            InitializeComponent();

            DBManager dbm = new DBManager();
            string sql;
            if(isPost)
                sql = "SELECT caption as content FROM posts WHERE postID=" + id;
            else
                sql = "SELECT content FROM reviews WHERE reviewID=" + id;
            List<Dictionary<string, object>> r = dbm.execute(sql);
            string content = r[0]["content"].ToString();
            textBox1.Text = content;
        }

        private void button1_Click(object sender, System.EventArgs e)
```

```
            {
                DBManager dbm = new DBManager();
                string content = textBox1.Text.Trim();
                if(content.Length == 0)
                {
                    MessageBox.Show("Text Field is empty!");
                    this.Close();
                    return;
                }
                SqlCommand sc = new SqlCommand();
                if (isComment)
                {
                    dbm.add_review(id, postID, content);
                    this.Close();
                    return;
                }

                if (isPost)
                    sc.CommandText = "UPDATE posts SET caption = @content WHERE postID="+id;
                else
                    sc.CommandText = "UPDATE reviews SET content = @content WHERE reviewID=" + id;
                sc.Parameters.Add("@content", System.Data.SqlDbType.VarChar).Value = content;
                dbm.conn.Open();
                sc.Connection = dbm.conn;
                sc.ExecuteNonQuery();
                dbm.conn.Close();
                this.Close();
            }
        }
    }
```

- Home.cs

```
using System;
using System.Collections.Generic;
using System.Runtime.InteropServices;
using System.Windows.Forms;

namespace rating.Frames
{
    public partial class Home : Form
    {
        public int uid;
        public string name;
        public string username;
        private const int EM_GETLINECOUNT = 0xba;
        [DllImport("user32", EntryPoint = "SendMessageA", CharSet = CharSet.Ansi, SetLastError =
true, ExactSpelling = true)]
        private static extern int SendMessage(int hwnd, int wMsg, int wParam, int lParam);
        public Home(int uid)
        {
            InitializeComponent();
            this.uid = uid;
            comboBox1.SelectedIndex = 0;
            comboBox2.SelectedIndex = 0;
            comboBox3.SelectedIndex = 0;
            get_posts();
        }
        private void button2_Click(object sender, EventArgs e)
        {
            Add_Post ap = new Add_Post(uid);
            ap.Show();
            ap.Closed += (s, args) => this.get_posts();

        }
        public void button1_Click(object sender, EventArgs e)
        {
            get_posts();
        }
        private void button6_Click(object sender, EventArgs e)
        {
            Login login = new Login();
            this.Hide();
            login.Closed += (s, args) => this.Close();
            login.Show();
        }
        private void button5_Click(object sender, EventArgs e)
        {
            string sql = "SELECT topics.topic as topic FROM reviewers_speciality INNER JOIN topics
    ON reviewers_speciality.topicID = topics.topicID WHERE userID = " + uid;
            DBManager dbm = new DBManager();
            List<Dictionary<string,object>> r = dbm.execute(sql);
            string speciality = "";
            foreach(Dictionary<string, object> row in r)
            {
                speciality = speciality + row["topic"].ToString() + "; ";
            }
            Register register = new Register(uid, name, username, speciality);
            register.Closed += (s, args) => this.get_posts();
            register.Show();
        }
        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            get_posts();
        }

        private void comboBox3_SelectedIndexChanged(object sender, EventArgs e)
        {
            get_posts();
        }

        private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
        {
            get_posts();
        }

        private void textBox1_TextChanged(object sender, EventArgs e)
        {
            get_posts();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            get_posts();
        }
        public void get_posts()
        {
            string s = "SELECT Name, username FROM users WHERE userID=" + uid;
            DBManager dbm = new DBManager();
            List<Dictionary<string, object>> r = dbm.execute(s);
            name = r[0]["Name"].ToString();
            username = r[0]["username"].ToString();
            label5.Text = name;
            label7.Text = username;

            if (dbm.table_not_exists("posts"))
                return;
            string Search = textBox1.Text.Trim();
            string View = comboBox1.Text.Trim();
            string Type = comboBox3.Text.Trim();
            string Short = comboBox2.Text.Trim();

            string revSrt = "DESC";
            Search = Search.Replace("'", "").Replace("\"", "").Replace("`", "").Replace("%",
"").Replace("?", "").Replace("_", "").Replace("*", "");

            if (Type.Equals("All"))
                Type = "(posts.userID = " + this.uid + " OR a.userID = " + this.uid + ")";
            else if (Type.Equals("My Works"))
                Type = "(posts.userID = " + this.uid + ")";
            else
                Type = "(a.userID = " + this.uid + ")";

            if (View.Equals("All"))
                View = " ";
            else if (View.Equals("Reviewed"))
                View = " AND (reviews.reviewID IS NOT NULL) ";
            else
                View = " AND (reviews.reviewID IS NULL) ";

            if (Short.Equals("Oldest Posts First"))
                Short = " ORDER BY posts.postedON ASC";
            else if(Short.Equals("Oldest Reviews First"))
            {
                Short = "ORDER BY CASE WHEN reviews.postedON IS NULL THEN 1 ELSE 0 END,
reviews.postedON";
                revSrt = "ASC";
            }
            else if (Short.Equals("Newest Posts First"))
                Short = " ORDER BY posts.postedON DESC";
            else
                Short = " ORDER BY reviews.postedON DESC";
            string sql = "SELECT posts.postID as postID, posts.userID as userID, posts.topicID as
topicID, posts.postedON as postedON, posts.caption as caption, posts.document_name as
document_name FROM posts INNER JOIN assignments_posts a ON posts.postID=a.postID LEFT
JOIN reviews ON posts.postID = reviews.postID WHERE " + Type + " AND (posts.caption LIKE '%"
+ Search + "%' OR reviews.content LIKE '%" + Search + "%')" + View + Short;
            List<Dictionary<string, object>> posts = dbm.execute(sql);
            List<int> postID = new List<int>();
            List<Dictionary<string, object>> unique_posts = new List<Dictionary<string, object>>();
            foreach (Dictionary<string, object> row in posts)
            {
                if (!postID.Contains(Convert.ToInt32(row["postID"])))
                {
                    postID.Add(Convert.ToInt32(row["postID"]));
                unique_posts.Add(row);
            }
        }
        paint_posts(unique_posts, revSrt);
    }
    public void paint_posts(List<Dictionary<string, object>> result, string Short)
    {
        this.panel2.Controls.Clear();

        if (result.Count > 0)
        {
            int h = 10;
            foreach (Dictionary<string, object> row in result)
            {
                Custom_Widgets.Post post = new Custom_Widgets.Post(uid, row, Short, this);
                post.Location = new System.Drawing.Point(5, h);
                this.panel2.Controls.Add(post);
                h += post.Height + 5;
            }
        }
        this.panel2.Refresh();
    }
  }
}
```

- Login.cs

```
using System;
using System.Collections.Generic;
using System.Text.RegularExpressions;
using System.Windows.Forms;

namespace rating.Frames
{
    public partial class Login : Form
    {
        public Login()
        {
            InitializeComponent();
        }

        private void checkBox1_CheckedChanged(object sender, EventArgs e)
        {
            if (checkBox1.Checked)
                textBox2.UseSystemPasswordChar = false;
            else
                textBox2.UseSystemPasswordChar = true;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            string username = textBox1.Text.Trim();
            string password = textBox2.Text.Trim();
            string unameRegex = @"^[A-Za-z][A-Za-z0-9@_]*$";
            string passRegex = @".";
            if (username.Length == 0)
            {
                MessageBox.Show("Username cannot be empty!");
                textBox1.Focus();
                return;
            }
            else if (username.Length > 50)
            {
                MessageBox.Show("Username cannot longer than 50 characters!");
                textBox1.Focus();
                return;
            }
            if (!Regex.IsMatch(username, unameRegex))
            {
                MessageBox.Show("Invalid Username!");
                textBox1.Focus();
                return;
            }
            if (password.Length == 0)
            {
                MessageBox.Show("Password cannot be empty!");
                textBox1.Focus();
                return;
            }
            else if (password.Length > 50)
            {
                MessageBox.Show("Password cannot longer than 50 characters!");
                textBox1.Focus();
                return;
            }
            if (!Regex.IsMatch(password, passRegex))
            {
                MessageBox.Show("Invalid Password!");
                textBox1.Focus();
                return;
            }
            DBManager dbm = new DBManager();
            string passhash = Modules.Crypto.Sha256(password);
            string sql = "SELECT userID,user_role FROM users WHERE username='" + username + "'
AND psword='" + passhash + "'";
            List<Dictionary<string, object>> r = dbm.execute(sql, true);
            if (r.Count == 0)
            {
                MessageBox.Show("Username and Password combination incorrect!");
                return;
            }
            int uid = Convert.ToInt32(r[0]["userID"]);
            int role = Convert.ToInt32(r[0]["user_role"]);
            this.Hide();
            if (role == 0)
            {
                var admin = new Admin(uid);
                admin.Closed += (s, args) => this.Close();
                admin.Show();
            }
            else
            {
                var home = new Home(uid);
                home.Closed += (s, args) => this.Close();
                home.Show();
            }
        }
        private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
        {
            this.Hide();
            var register = new Register();
            register.Closed += (s, args) => this.Close();
            register.Show();
        }
    }
}
```

- Register.cs

```
using System;
using System.Collections.Generic;
using System.Text.RegularExpressions;
using System.Windows.Forms;

namespace rating.Frames
{
    public partial class Register : Form
    {
        public int uid;
        public string name;
        public string username;
        public string speciality;
        public bool isUpdateWindow = false;
        public int a = 2;
        public Register()
        {
            InitializeComponent();
            refresh_topic_list();
            button3.Enabled = false;
            button3.Hide();
        }
        public Register(int a)
        {
            this.a = a;
            InitializeComponent();
            refresh_topic_list();
            this.linkLabel1.Hide();
            button3.Enabled = false;
            button3.Hide();
        }

        public Register(int uid, string name, string username, string speciality)
        {
            this.uid = uid;
            this.name = name;
            this.username = username;
            this.speciality = speciality;
            this.isUpdateWindow = true;

            InitializeComponent();
            refresh_topic_list();

            linkLabel1.Hide();
            textBox3.Text = name;
    textBox1.Text = username;
    label1.Text = "Update Account";
    label1.Location = new System.Drawing.Point(88, 9);
    this.Text = "Update Account";
    button1.Text = "Update";
}

private void button2_Click(object sender, EventArgs e)
{
    Add_Topic at = new Add_Topic();
    at.Closed += (s, args) => this.refresh_topic_list();
    at.Show();
}

private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox1.Checked)
        textBox2.UseSystemPasswordChar = false;
    else
        textBox2.UseSystemPasswordChar = true;
}

private void button1_Click(object sender, EventArgs e)
{
    string name = textBox3.Text.Trim();
    string username = textBox1.Text.Trim();
    string password = textBox2.Text.Trim();
    string nameRegex = @"^[A-Za-z][A-Za-z ,.'-]*$";
    string unameRegex = @"^[A-Za-z][A-Za-z0-9@_]*$";
    string passRegex = @".";

    if (name.Length == 0)
    {
        MessageBox.Show("Name cannot be empty!");
        textBox3.Focus();
        return;
    }
    else if (name.Length > 255)
    {
        MessageBox.Show("Name cannot longer than 255 characters!");
        textBox3.Focus();
        return;
    }
    if (!Regex.IsMatch(name, nameRegex))
    {
        MessageBox.Show("Invalid Name!");
        textBox3.Focus();
        return;
    }
    if (username.Length == 0)
    {
        MessageBox.Show("Username cannot be empty!");
        textBox1.Focus();
        return;
    }
    else if (username.Length > 50)
    {
        MessageBox.Show("Username cannot longer than 50 characters!");
        textBox1.Focus();
```

```
                return;
            }
            if (!Regex.IsMatch(username, unameRegex))
            {
                MessageBox.Show("Invalid Username!");
                textBox1.Focus();
                return;
            }
            if (password.Length == 0)
            {
                MessageBox.Show("Password cannot be empty!");
                textBox2.Focus();
                return;
            }
            else if (password.Length > 50)
            {
                MessageBox.Show("Password cannot longer than 50 characters!");
                textBox2.Focus();
                return;
            }
            if (!Regex.IsMatch(password, passRegex))
            {
                MessageBox.Show("Invalid Password!");
                textBox2.Focus();
                return;
            }
            string speciality = "";
            for (int i = 0; i < checkedListBox1.Items.Count; i++)
            {
                if (checkedListBox1.GetItemCheckState(i) == CheckState.Checked)
                {
                    speciality = speciality + checkedListBox1.Items[i].ToString() + "; ";
                }
            }
            if (speciality == "")
            {
                MessageBox.Show("Select atleast one speciality!");
                return;
            }

            if (isUpdateWindow)
            {
                DBManager dbm = new DBManager();
                string sql = "SELECT * FROM users WHERE username='" + username + "' AND
userID!=" + uid;
                List<Dictionary<string, object>> r = dbm.execute(sql, true);
                if (r.Count != 0)
                {
                    MessageBox.Show("Username already taken!");
                    textBox1.Focus();
                    return;
                }

                string passhash = Modules.Crypto.Sha256(password);
                dbm.update_user(uid, name, username, passhash, 2, speciality);
                MessageBox.Show("Account Successfully Updated!");
                this.Close();
            }
            else
```

```
    {
        DBManager dbm = new DBManager();
        if (!dbm.table_not_exists("users"))
        {
            string sql = "SELECT * FROM users WHERE username='" + username + "'";
            List<Dictionary<string, object>> r = dbm.execute(sql, true);
            if (r.Count != 0)
            {
                MessageBox.Show("Username already taken!");
                textBox1.Focus();
                return;
            }
        }

        string passhash = Modules.Crypto.Sha256(password);
        dbm.add_user(name, username, passhash, 1, speciality);
        MessageBox.Show("Account Successfully Created!");
        if (a != 2)
        {
            this.Close();
        }
        else
        {
            this.Hide();
            var login = new Login();
            login.Closed += (s, args) => this.Close();
            login.Show();
        }

    }
}
public void refresh_topic_list()
{
    checkedListBox1.Items.Clear();
    DBManager dbm = new DBManager();
    if (dbm.table_not_exists("topics"))
        return;
    string sql = "SELECT * FROM topics";
    List<Dictionary<string, object>> r = dbm.execute(sql);
    int i = 0;
    foreach (Dictionary<string, object> row in r)
    {
        checkedListBox1.Items.Insert(i, row["topic"].ToString());
        i++;
    }
    if (isUpdateWindow)
    {
        string[] specialities = speciality.Split("; ");
        List<int> lst = new List<int>();
        for (i = checkedListBox1.Items.Count-1; i>=0 ; i--)
        {
            int pos = Array.IndexOf(specialities, checkedListBox1.Items[i].ToString());
            if (pos > -1)
                lst.Add(i);
        }
        foreach (int a in lst)
            checkedListBox1.Items.RemoveAt(a);
    }
}
```

```
        private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
        {
            this.Hide();
            var login = new Login();
            login.Closed += (s, args) => this.Close();
            login.Show();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            DBManager dbm = new DBManager();
            string sql = "SELECT t.topic as topic, r.AVGrate as AVGrate FROM reviewers_speciality r
inner join topics t ON r.topicID=t.topicID WHERE userID = " + uid;
            List<Dictionary<string, object>> tbl = new List<Dictionary<string, object>>();
            tbl = dbm.execute(sql);
            string message = "Your Skill points(average rating) for each speciality are as follow: \n\n";
            foreach(Dictionary<string, object> row in tbl)
            {
                message += row["topic"].ToString() + ": " + Convert.ToInt32(row["AVGrate"]) + "\n";
            }
            MessageBox.Show(message, "Skill Points");
        }
    }
}
```

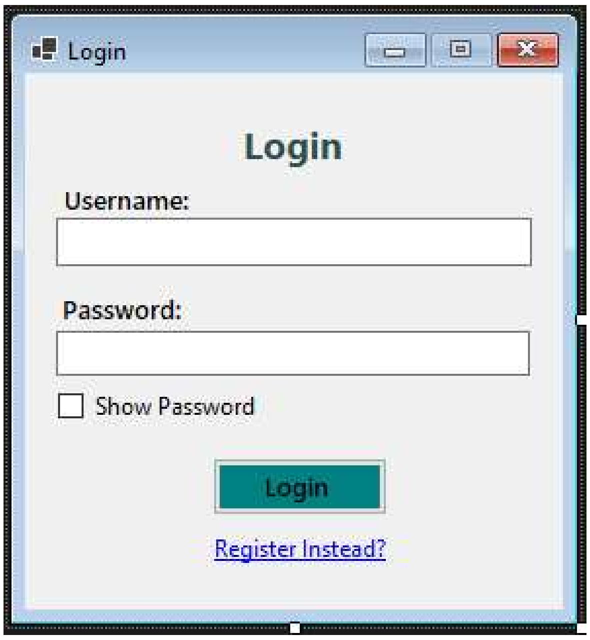

Figure 9: Login Form Window

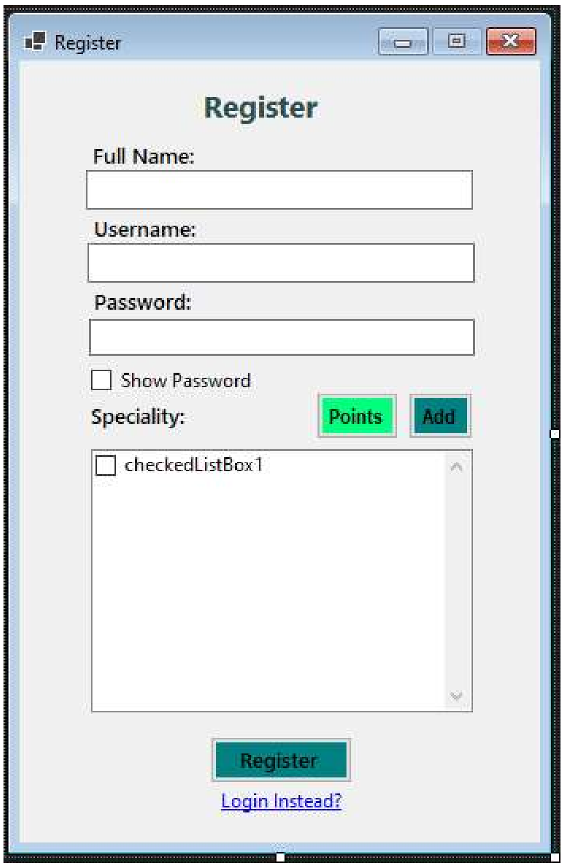

Figure 10: Register Form Window (Resized)

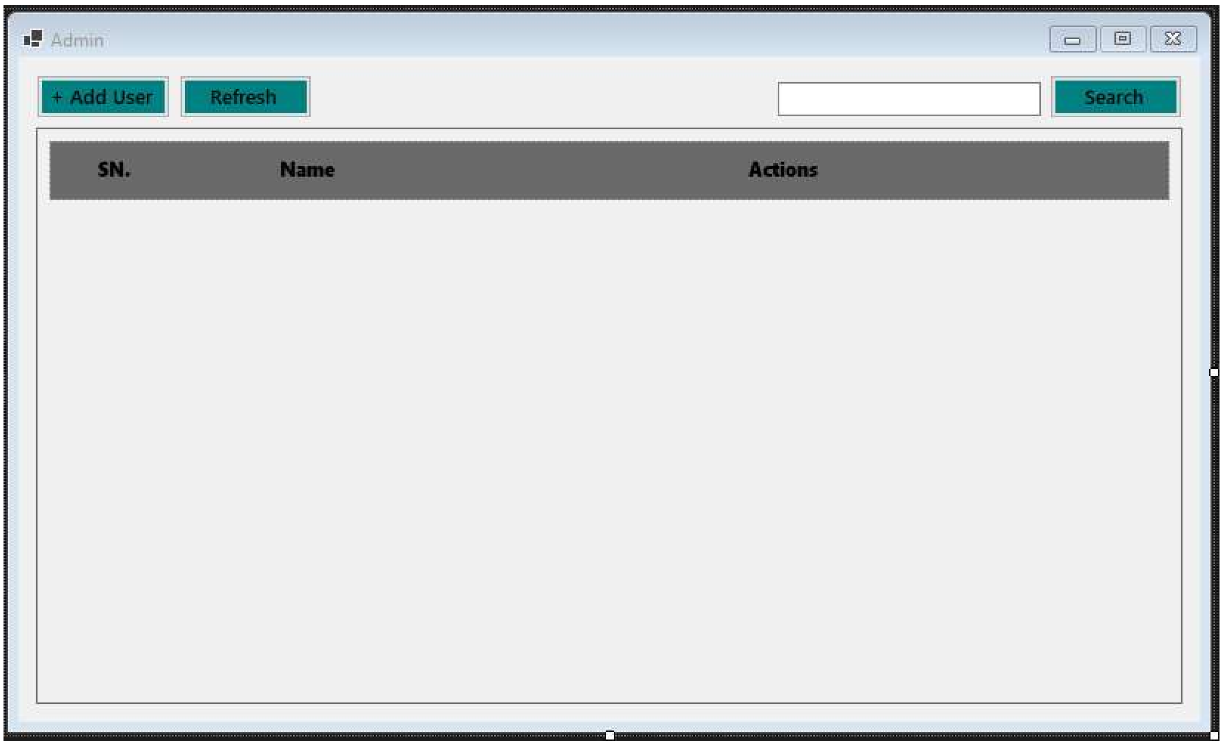

Figure 11: Admin's Main Window (Note: Design View doesn't contain items , panels or views that are build and attached on runtime)

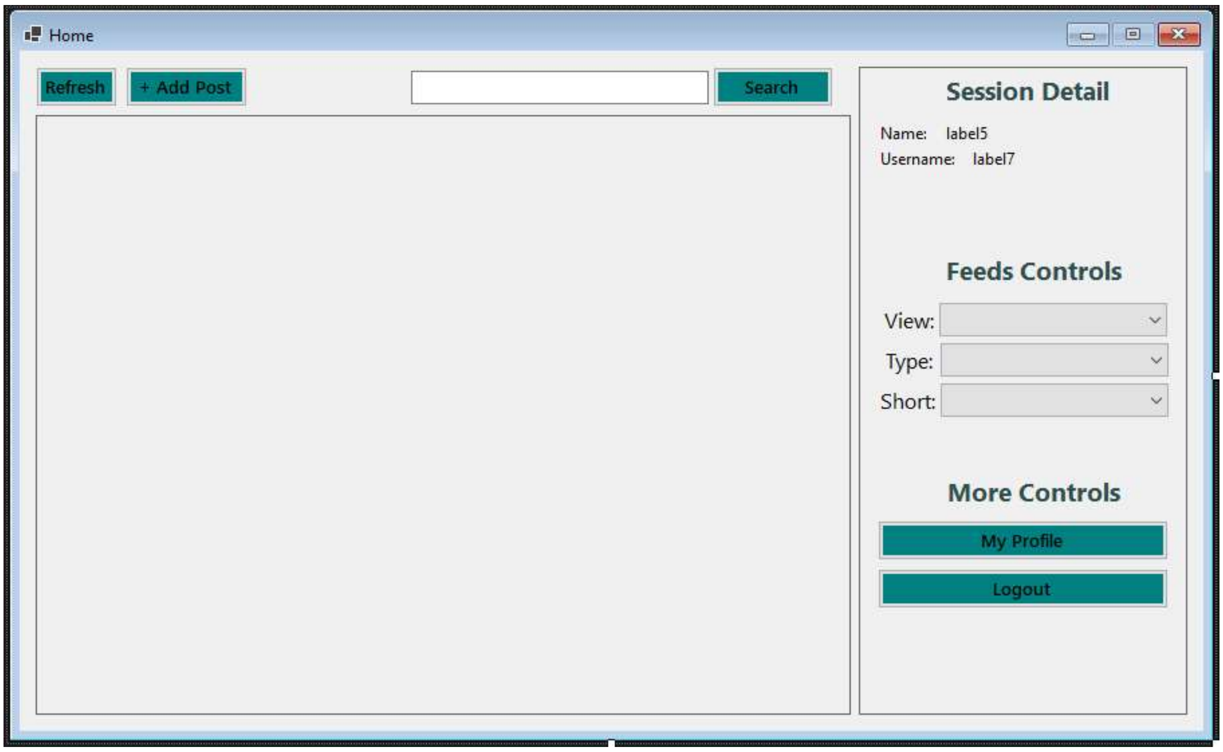

Figure 12: User's Main Window (Note: Design View doesn't contain items , panels or views that are build and attached on runtime)

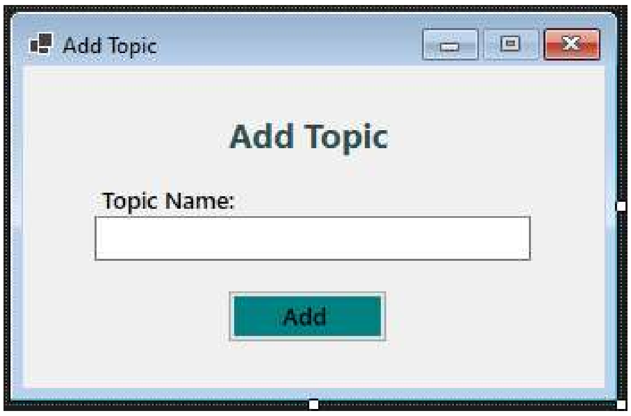

Figure 13: Add Topic(Special skill) Window (Resized)

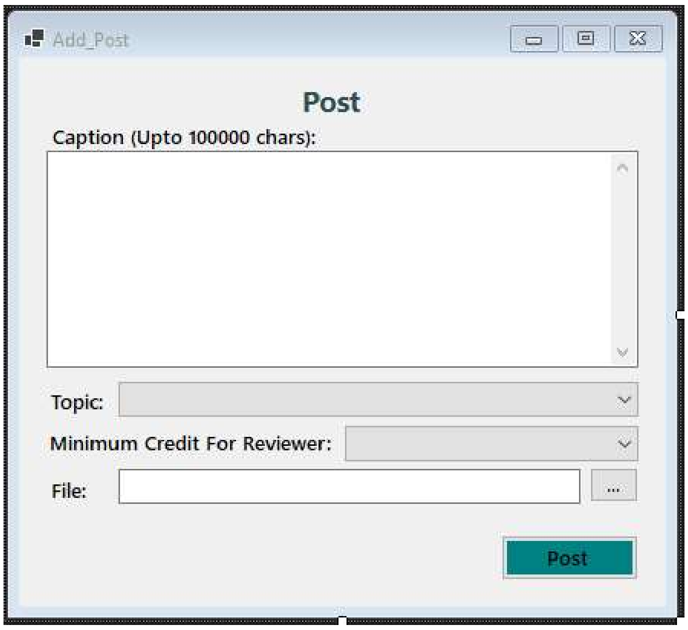

Figure 14: Add Post Window (Resized)

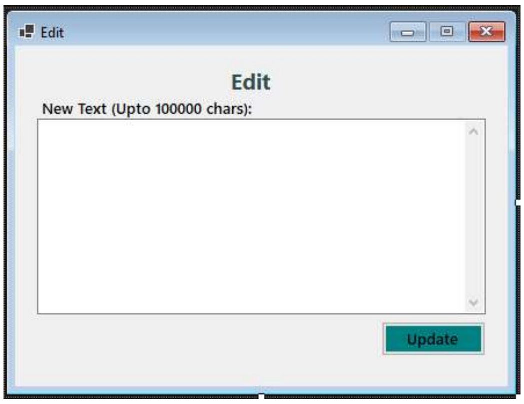

Figure 15: Add/Edit Review Window (Resized)

Task 6 Software quality and maintenance

1. Term design pattern is a repeatable solution for the common problems that occurs while developing a software. In other words, it can be explained as a template or description for correcting the possible problems that can occur during the development of a software system.

   The three families of design patterns are shown below:

   - Behavioral patterns: These pattern helps to describe the interactions between objects with each other.
     Example for Behavioral pattern is shown below:

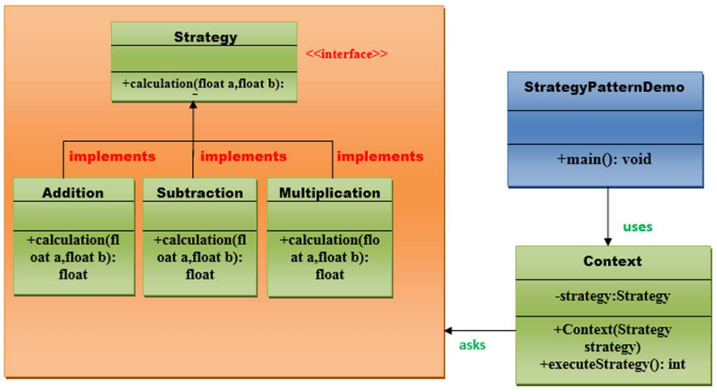

Figure 16: Strategy pattern

   - Creational Patterns: These patterns are used for creating objects to suitable class.
     Example for Creational pattern is shown below:

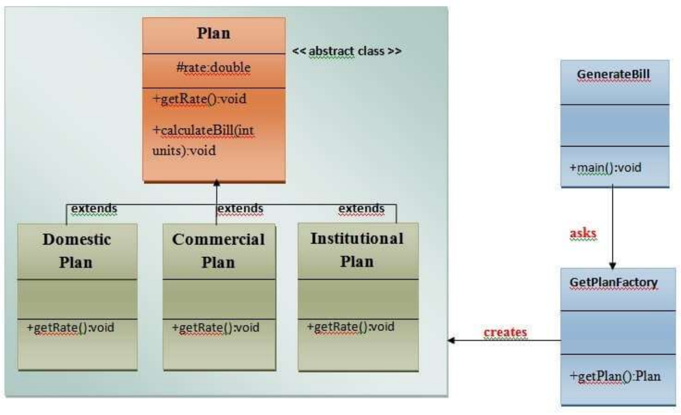

Figure 17: Factory method pattern

- Structural Patterns: These patterns are used for connecting the classes and objects to form a larger system.
  Example for Structural Pattern is shown below:

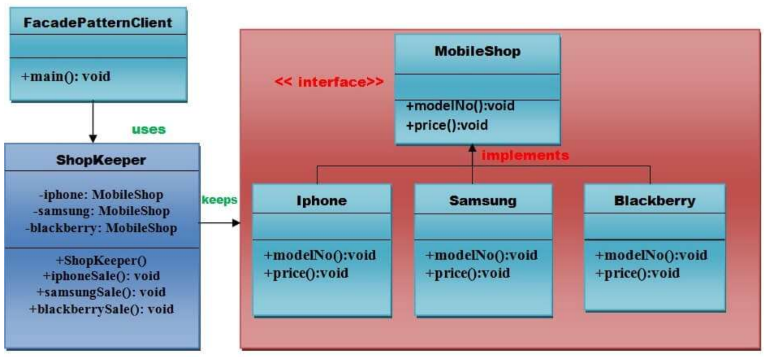

Figure 18: Facade Pattern

2. Refactoring of the software is considered once all the coding is completed to assure that it improves the internal structure of the system code. Refactoring is performed to clean up unnecessary code to overcome the problem of bugs and failures. Design patterns rarely need refactoring because design patterns are itself build with refactoring feature.

Conclusion

Peer review system for the University was designed and developed successfully including all the components that are required for the document to be completed. The principles of ADI (Analysis Design and Implementation) was followed in order to complete the report. Selective class were justified and class diagram was drawn. Furthermore, Activity diagram was also drawn using the features and activities involved in the requirements. In addition to this Use case diagram was also drawn to list all the features of the peer review system for the university. Similarly, in implementation phase, coding was completed as per the requirements of the documents. Lastly, the development of peer review tracking system was successfully made.