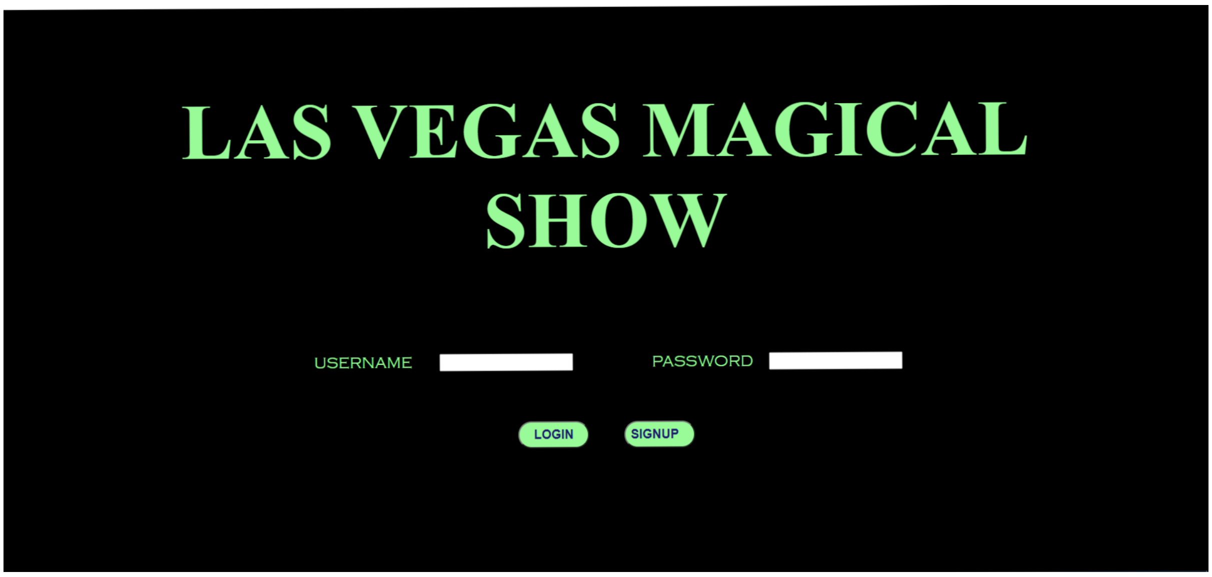
Backend assignment - Las Vegas Magic Show

Mie Grønkjær B. Jørgensen
cph-mj761@cphbusiness.dk

Web Developer



Assignment

Make a Person class.

You decide which information is needed, but id, name and password must be present.

The Person class must be abstract.

Make two subclasses Magician and Staff which inherit Person.

You decide which information is needed, but staff needs a salary.

Magicians need a collection of favourite tricks and must have an AddFavouriteTrick method.

It must be possible to create magicians with or without a single favourite trick.

Make web pages where magicians and staff can enlist by entering the necessary data.

Persons must be created as objects and placed in an ArrayList.

Make a simple login page where existing magicians and staff can login and logout using your own code.

Make pages for displaying and updating magicians and staff including adding magicians favourite tricks.

A person must be able to change own data only.

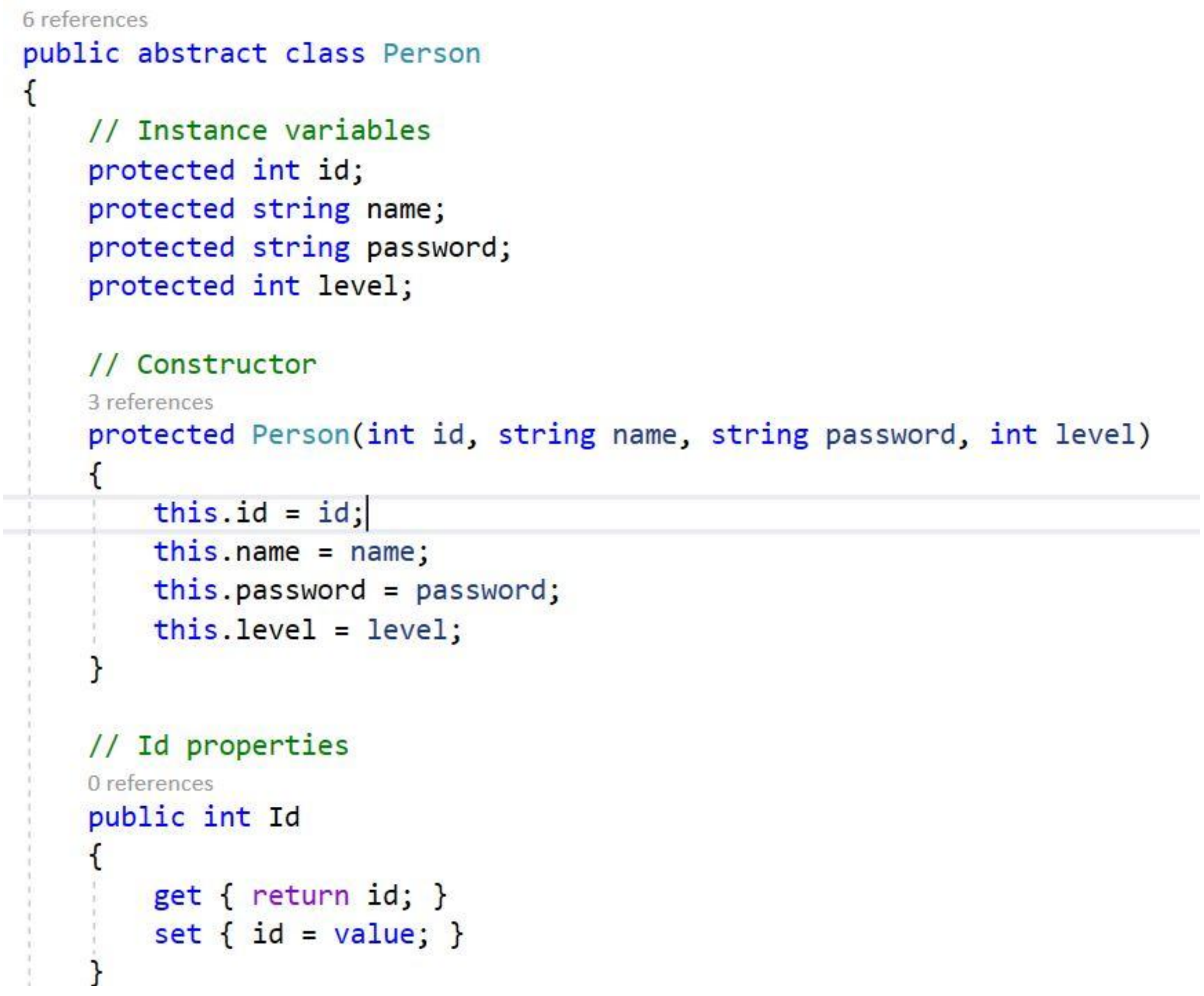
Magicians who have logged in must be able to see a list of all magicians.

Staff who have logged in must be able to see a list of all persons.

The visual appearance of the pages should be professional.

Classes

As the assignment says, I first created a person class which is gonna be the parent class. Which I remembered to create as an abstract as told in the assignment. I did also remember to make it public, so we can connect the magician and staff class later on. Which has id, name, password and I added a level object as well. The level to show if the user is a magician or a staff member.
Then I created a constructor to set up, what properties a new person should contain, as well as property set up.



```
6 references
public abstract class Person
{
    // Instance variables
    protected int id;
    protected string name;
    protected string password;
    protected int level;

    // Constructor
    3 references
    protected Person(int id, string name, string password, int level)
    {
        this.id = id;
        this.name = name;
        this.password = password;
        this.level = level;
    }

    // Id properties
    0 references
    public int Id
    {
        get { return id; }
        set { id = value; }
    }
```

I then ended with creating a ToString() for the person:
public override string ToString()
        {
            return "id: " + id + ", name: " + name + ", password: " + password + ", level: " + level;
        }.

Magician class

I then created the magician subclass which inherits from the Person class. In which I created a list string for the tricks, which the magician should have:

private List<string> fav_tricks = new List<string>();
Then with using the AddFavoritetrick method, I then tell the program that I want the list string I created for the tricks, to be able to count up, or add new when changing the information in the userprofile page, which I create later on.

```
 public class Magicians : Person
  {
      private List<string> fav_tricks = new List<string>();



      // Constructor
      public Magicians(int id, string name, string password, int level) : base(id, name,
password, level)
      {
         this.id = id;
         this.name = name;
         this.password = password;
         this.level = level;
      }

      public void AddFavoriteTrick(string trick)
      {
         if (trick != "")
         {
            if (trick.Contains(','))
            {
               List<string> tricksArr = trick.Split(',').ToList();
               for (int i = 0; i < tricksArr.Count; i++)
               {
                  fav_tricks.Add(tricksArr[i].ToString());
               }
            }
            else
            {
               fav_tricks.Add(trick);
            }
         }
      }
      public List<string> FavTricks
      {
         get { return fav_tricks; }
         // No set
      }

      public string displayTricks()
```

```
        {
            string tricks = "";
            if (fav_tricks.Count > 0)
            {

                return tricks = string.Join(", ", fav_tricks.ToArray());
            }
            else
            {
                return tricks = "-";
            }

        }
        public override string ToString()
        {
            return name + " - Magician";
        }
    }
```

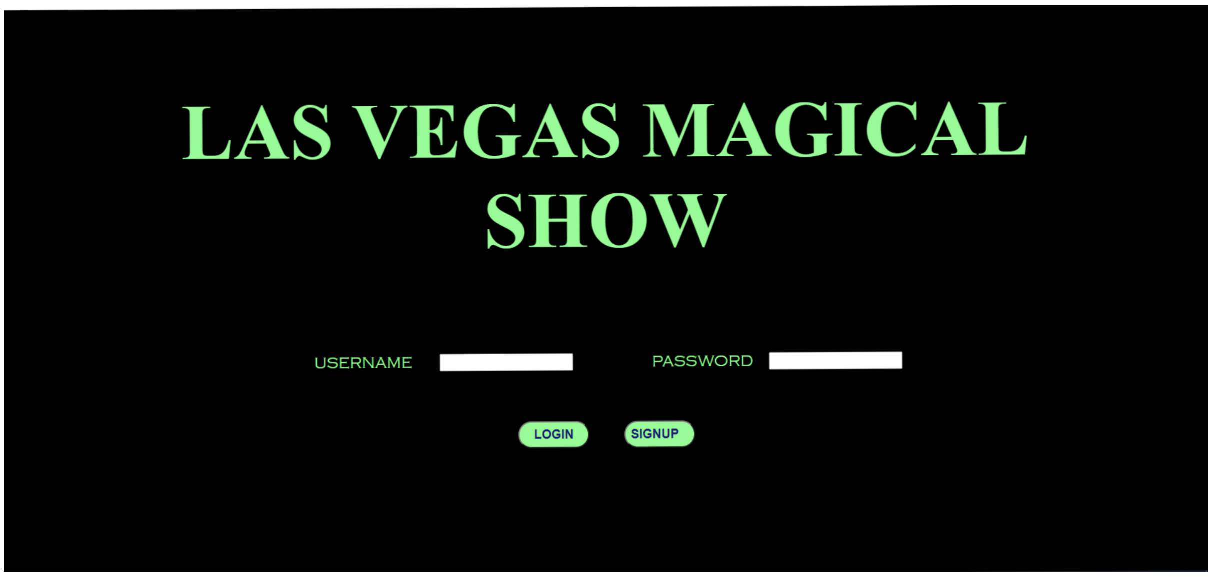
Staff class

I did the same in the Staff subclass which also inherits from Person, but I remembered to give it a salary. I gave the salary a value of 2000.

```
public class Staff : Person
    {
        protected int salary;

        // Constructor
        public Staff(int id, string name, string password, int level, int salary) : base(id,
name, password, level)
        {
            this.salary = salary;
        }

        public Staff(int id, string name, string password, int level) : base(id, name,
password, level)
        {
            salary = 2000; //defines the start salary which can be changed in the userprofile
page.
        }
        public int Salary
        {
            get { return salary; }
            set { salary = value; }
        }

        public override string ToString()
        {
            return name + " - Staff";
        }
    }
```
I remembered to put salary and tricks as protected, in order for the program to be able to inherit from Person.

Web forms

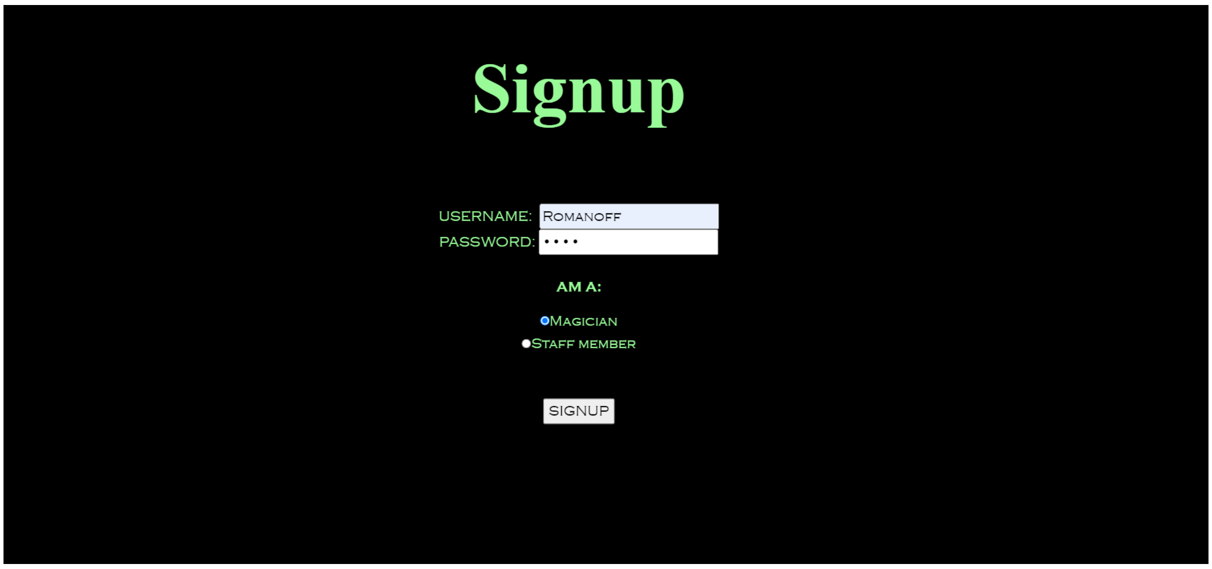
Index.aspx



I then created an Index.aspx file, which is gonna be the login page, I just simply forgot to change the name. However on this page is where a person can either login or press the signup button to go to the signup.aspx page where they can create a new user.
I did, under source, put in some bootstrap code so that I could more easily change the design layout.
In design mode, I put in a big title, and then add a few labels for name and password, together with some text boxes which I named: TextBox_Password and TextBoxName.
and a listbox which is for inputting the arraylist with the next users. It basically checks if the name and password matches with the user that has been created, if it does then you dobbelt click the login button and it will take you to the userprofiles.aspx page.
Then in the Index.aspx.cs file I then put in the code which will check if the input is correct, by using if and else statements.

```
ArrayList localarray;
        object user;
        protected void Page_Load(object sender, EventArgs e)
        {
            localarray = (ArrayList)Application["users"];
            TextBox_password.TextMode = TextBoxMode.Password;
            // If TextBoxUsername.Text exists in arraylist && password match, login and
redirect to userProfiles.aspx
            // Else display "No user with these credentials found."
            if (localarray != null)
            {
                for (int i = 0; i < localarray.Count; i++)
                {
                    ListBox1.Items.Add(localarray[i].ToString());
                }
            }
            else
            {
                ListBox1.Items.Add("No users");
            }
        }

        protected void Button_signup_Click(object sender, EventArgs e)
        {
            Response.Redirect("Signup.aspx"); //Sents you to the signup page
        }

        protected void Button_login_Click(object sender, EventArgs e)
        {
            for (int i = 0; i < localarray.Count; i++) //When pressed, should know the user
and password which was just registered,
                //and then should redirect you to the userProfiles page.

            {
                if (localarray[i].ToString().Contains(TextBox_username.Text) &&
localarray[i].ToString().Contains(TextBox_password.Text))
                {
                    user = localarray[i];
                    Session["currentUser"] = user;
                    Session["currentLevel"] = user.GetType().Name;
                    localarray = (ArrayList)Application["users"];
                    Response.Redirect("userProfiles.aspx");
                    break;
                }
            }
        }
```

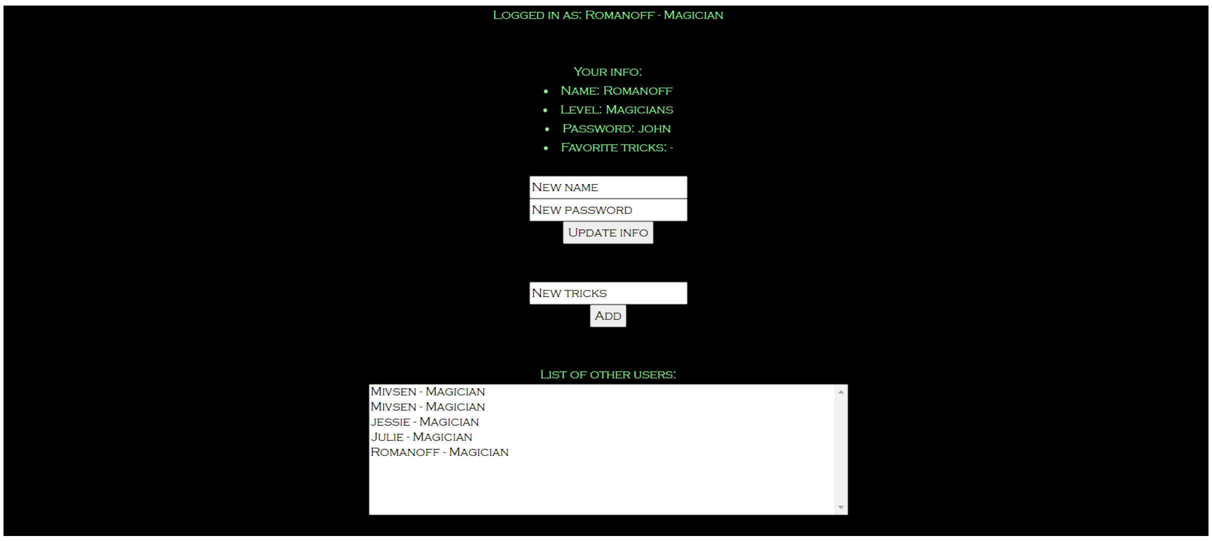
Signup.aspx



In the sign up page I put in as you can see again labels and textboxes, matching name and password. However I also used a radiobuttonlist (id = radiobuttonlist_usertype), which contains the magician or staff option.
so in the Signup.aspx.cs file I then put in the properties of what happens when interacting with these. Having name and password being connected to the textboxes matching their names.
I then for the signup button, put in the information for what happens when clicked. In there I use if, else statements in order to tell the program that if the radio button pressed value is magician or staff it will interact accordingly. The program knows that if magician matches the value then it should put in a new magician inside the arraylist created, and the same for if the value is staff.
I did also create a new arraylist for the users, so that they all will be put into that arraylist, to be shown in the listbox on the userprofiles page. This helps the program to collect the users.

```
public partial class Signup : System.Web.UI.Page
  {
     ArrayList userarray;
     protected void Page_Load(object sender, EventArgs e)
     {
        if (Application["users"] == null)
        {
           userarray = new ArrayList();
           Application["users"] = userarray;
        }
        userarray = (ArrayList)Application["users"];
        TextBox_sp_password.TextMode = TextBoxMode.Password;
     }

     protected void Button_sp_signup_Click(object sender, EventArgs e)
     {
            if (RadioButtonList_usertype.SelectedValue == "Magician")
            {
                Magicians m = new Magicians(Convert.ToInt32(userarray.Count + 1),
TextBox_sp_username.Text, TextBox_sp_password.Text, 1);
                userarray.Add(m);
                Response.Redirect("Index.aspx");
            }
            else
            {
                Staff s = new Staff(Convert.ToInt32(userarray.Count + 1),
TextBox_sp_username.Text, TextBox_sp_password.Text, 2);
                userarray.Add(s);
                Response.Redirect("Index.aspx");
            }
        }
    }
```

UserProfiles.aspx


Logged in as: Romanoff - Magician

Your info:

- Name: Romanoff
- Level: Magicians
- Password: John
- Favorite tricks: -

List of other users:

```
Mivsen - Magician
Mivsen - Magician
Jessie - Magician
Julie - Magician
Romanoff - Magician
```

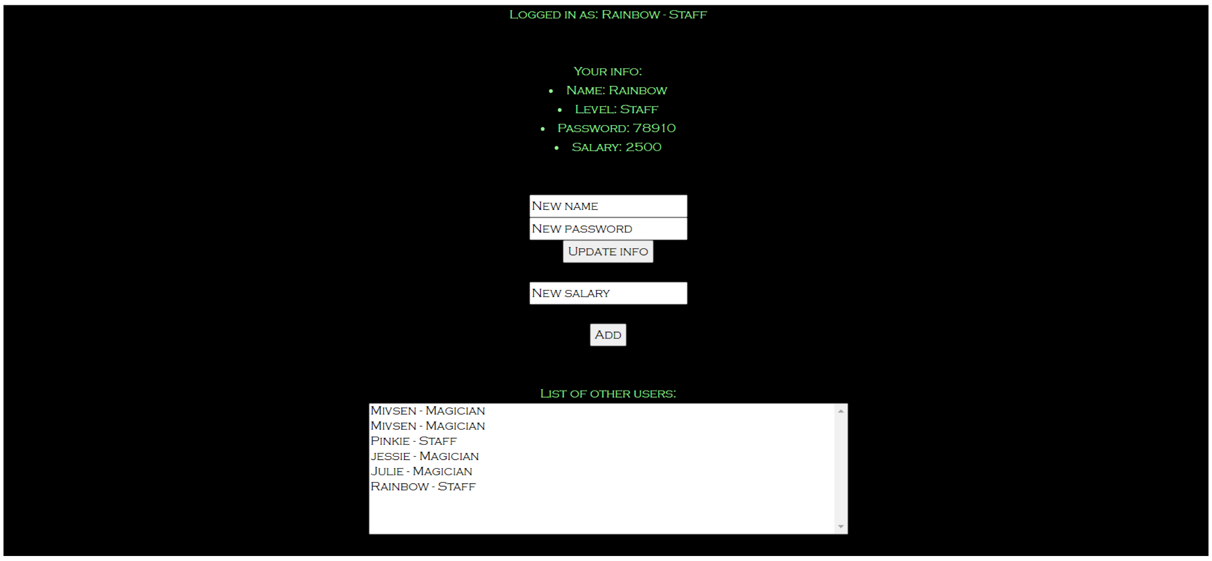
This page is where I can update the profile for the magician or staff member, as you can see in the picture if it is a magician then you can see the 'new tricks' button, where you can put in the tricks the magician does/likes. But as you can see it starts off with no tricks so that you can have a magician without a favorite trick and then add in a trick if needed.
As you can see the magician can only see all the other magician members, however if it had been a staff member then they would be able to see all the users of both magicians as well as staff.
Code from userProfiles.aspx.cs:
object user and userlvl, helps the program to distinguish against the users, as well as what type of user they are, which is the reasoning for the level insert from person. By using arraylists and if, else statements we tell the program what information should be used for the profile depending on the value either 'magician' or 'staff'.

```
 public partial class userProfiles : System.Web.UI.Page
  {
     object user;
     object userlvl;
     ArrayList allusers;
     protected void Page_Load(object sender, EventArgs e)
     {
        allusers = (ArrayList)Application["users"];
        user = Session["currentUser"];
        userlvl = Session["currentLevel"];
        Label_login.Text = "Logged in as: " + user;
        if (userlvl.ToString() == "Magicians")
        {
           Magicians currentm = (Magicians)user;
           TextBoxSal.Visible = false;
           Userinfo.Text = "<li> Name: " + currentm.Name + "</li>" + "<li> Level: " +
userlvl.ToString() + "</li>" + "<li> Password: " + currentm.Password + "</li>" + "<li>
Favorite tricks: " + currentm.displayTricks() + " </ li > ";
           for (int i = 0; i < allusers.Count; i++)
           {
              if (allusers[i].GetType().Name == "Magicians")
              {
                 ListBoxUsers.Items.Add(allusers[i].ToString());
              }
           }
        }
        else
        {
           Staff currents = (Staff)user;
           TextBoxFavTricks.Visible = false;
           Userinfo.Text = "<li> Name: " + currents.Name + "</li>" + "<li> Level: " +
userlvl.ToString() + "</li>" + "<li> Password: " + currents.Password + "</li>" + "<li>
Salary: " + currents.Salary + "</li>";

           foreach (var user in allusers)
           {
              ListBoxUsers.Items.Add(user.ToString());
           }
        }
     }

     protected void ButtonUpdate_Click(object sender, EventArgs e)
     {
        for (int i = 0; i < allusers.Count; i++)
```

```
    {
        if (allusers[i].ToString() == user.ToString())
        {
            if (userlvl.ToString() == "Magicians")
            {
                Magicians m = (Magicians)allusers[i];
                m.Name = TextBoxName.Text;
                m.Password = TextBoxPass.Text;
                m.AddFavoriteTrick(TextBoxFavTricks.Text);
                user = m;
                Session["currentUser"] = user;
                Response.Redirect("userProfiles.aspx");
            }
            else
            {
                Staff s = (Staff)allusers[i];
                s.Name = TextBoxName.Text;
                s.Password = TextBoxPass.Text;
                s.Salary = Convert.ToInt32(value: TextBoxSal.Text);
                user = s;
                Session["currentUser"] = user;
                Response.Redirect("userProfiles.aspx");
            }

        }
    }
}

protected void Button1_Click(object sender, EventArgs e)
{
    for (int i = 0; i < allusers.Count; i++)
    {
        if (allusers[i].ToString() == user.ToString())
        {
            if (userlvl.ToString() == "Magicians")
            {
                Magicians m = (Magicians)allusers[i];
                m.AddFavoriteTrick(TextBoxFavTricks.Text);
                user = m;
                Session["currentUser"] = user;
                Response.Redirect("userProfiles.aspx");
            }
            else
            {
```

```
                        Staff s = (Staff)allusers[i];
                        s.Salary = Convert.ToInt32(value: TextBoxSal.Text);
                        user = s;
                        Session["currentUser"] = user;
                        Response.Redirect("userProfiles.aspx");
                    }

                }
            }
        }

    }
```

An example for the userprofiles page with a staff member instead of magician:

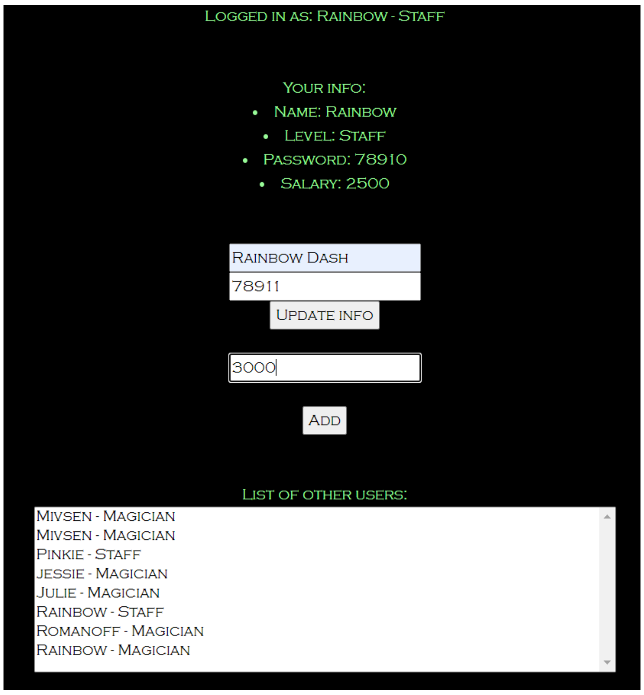

Logged in as: Rainbow - Staff

Your info:
- Name: Rainbow
- Level: Staff
- Password: 78910
- Salary: 2500

New name
New password
Update info
New salary
Add

List of other users:
```
Mivsen - Magician
Mivsen - Magician
Pinkie - Staff
Jessie - Magician
Julie - Magician
Rainbow - Staff
```

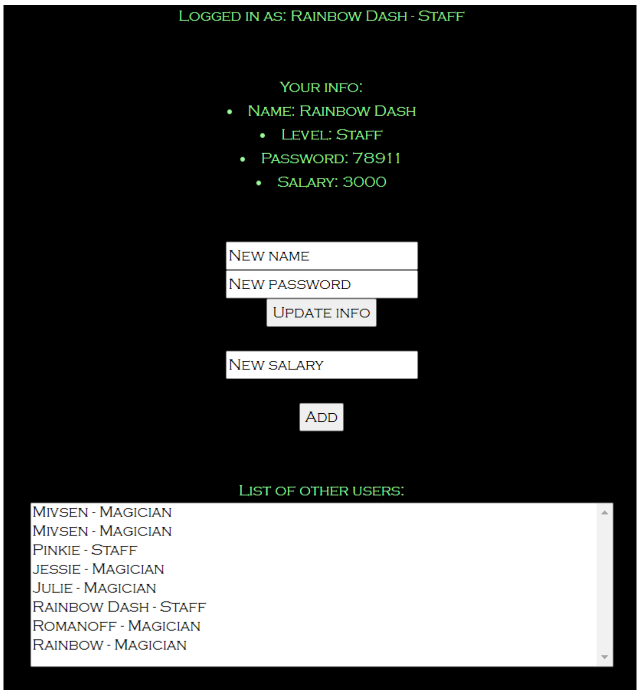
Examples of user profile update system:


Logged in as: Rainbow - Staff

Your info:
- Name: Rainbow
- Level: Staff
- Password: 78910
- Salary: 2500

Rainbow Dash
78911
Update info
3000
Add

List of other users:
```
Mivsen - Magician
Mivsen - Magician
Pinkie - Staff
Jessie - Magician
Julie - Magician
Rainbow - Staff
Romanoff - Magician
Rainbow - Magician
```


Logged in as: Rainbow Dash - Staff

Your info:
- Name: Rainbow Dash
- Level: Staff
- Password: 78911
- Salary: 3000

New name
New password
Update info
New salary
Add

List of other users:
```
Mivsen - Magician
Mivsen - Magician
Pinkie - Staff
Jessie - Magician
Julie - Magician
Rainbow Dash - Staff
Romanoff - Magician
Rainbow - Magician
```

Design

Style.css

So for the styling of the page, I created a CSS file to style the HTML (aspx) pages. Then I remembered to link to the stylesheet in all the pages (Index, Signup & userProfiles), to make sure all the pages would look similar.
I had already, with the bootstrap code, created <div>'s with a container class so that everything would turn more to the center. If something of the form didn't go to the center I would just use the ID or class name of the object and style it inside the style.css file.
There was not a lot of styling in here, since it wasn't my biggest priority, so I chose a black background, to add to a more mysterious vibe. I saw a lot of other people using this lime green color, which I really liked to use for the text. Together with the black background, it helps it stand out.

```
body {
   background-color: black;
   color: palegreen;
   font-family: 'Copperplate Gothic';
   font-size: 20px;
}

#LabelTitle, #LabelTitle1 {
   font-family: 'Emilys Candy';
   font-weight: 900;
   text-align: center;
   font-size: 100px;
}

#loginbBox {
   display: flex;
   align-items: center;
   justify-content: center;
   margin-top: 100px;
}

#Button_login, #Button_signup {
   justify-content: center;
   width: 200px;
   height: 35px;
   border-radius: 50px;
   margin-top: 30px;
   font-size: medium;
   font-weight: bold;
   color: darkblue;
   background-color: palegreen;
}

#LabelPassword {
   margin-left: 100px;
}

#TextBoxPassword, #TextBoxLoginName {
   background-color: lightblue;
   height: 35px;
   border-radius: 15px;
}

#all-div {
   margin-top: 100px;
}

#rightSide, #leftSide {
   flex: 50%;
}

.container {
   text-align: center;
}
#RadioButtonList_usertype {
   display: flex;
   justify-content: center;
}
#big{
   font-weight:bold;
}
```