Object Oriented Programming

Lab Manual 09

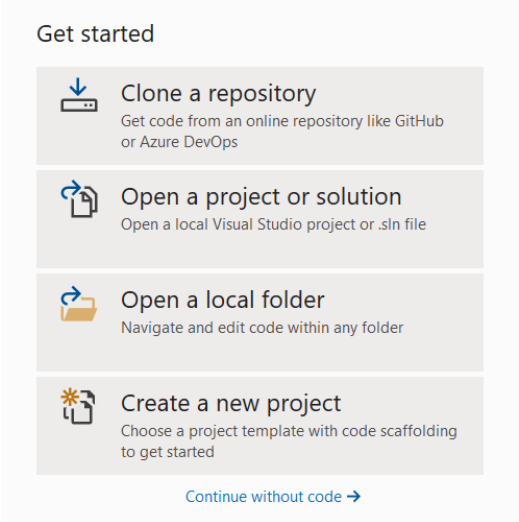
Introduction

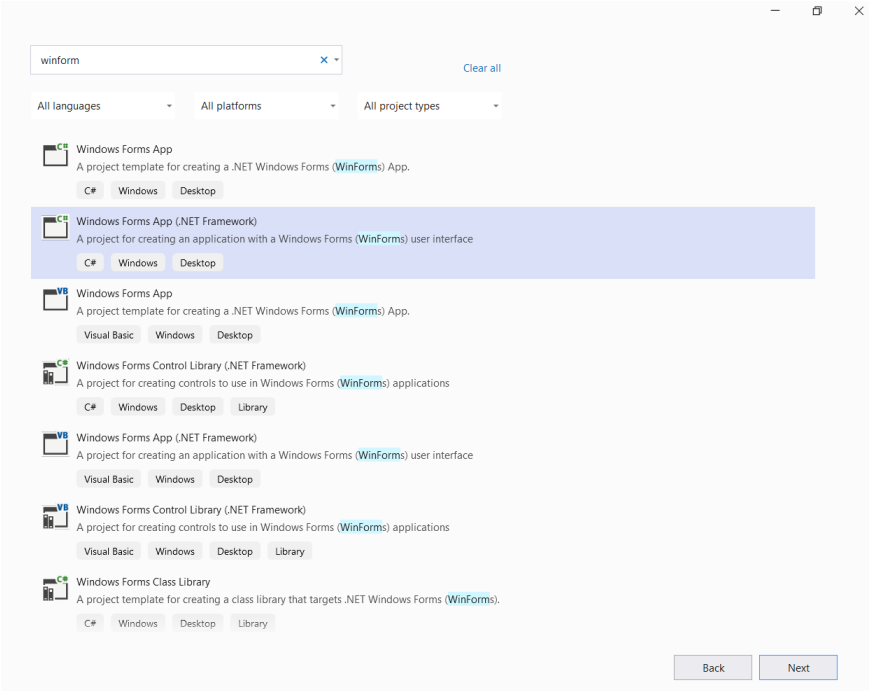
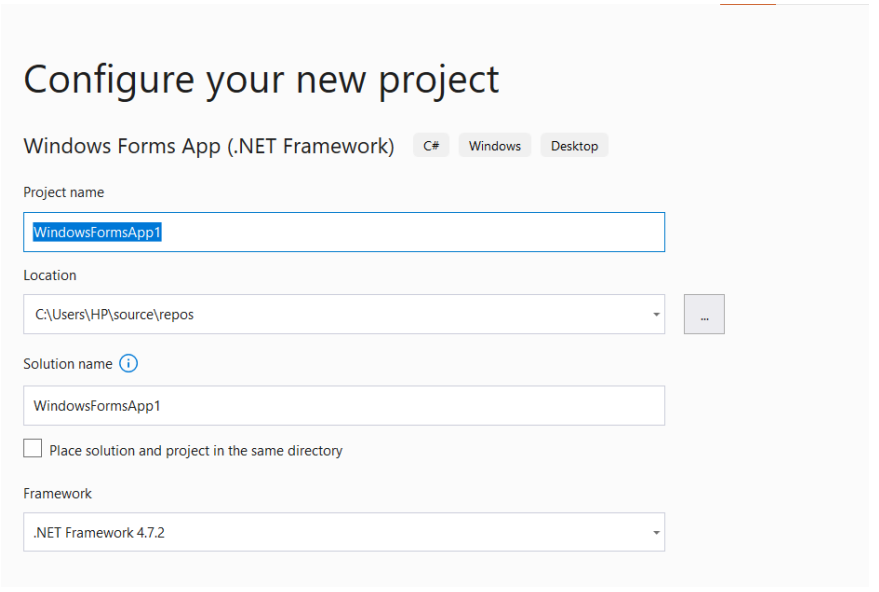
After a week of rigorous coding, Welcome back!

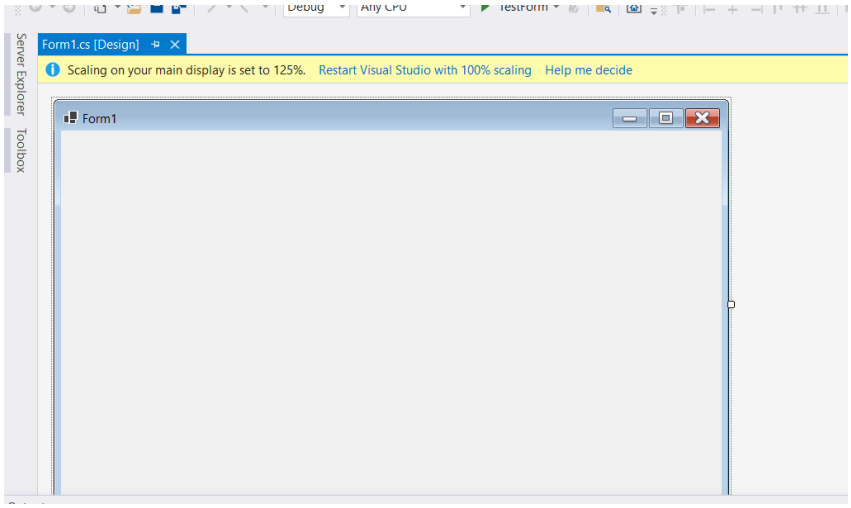
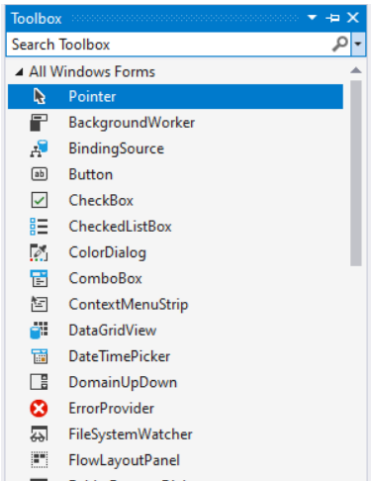
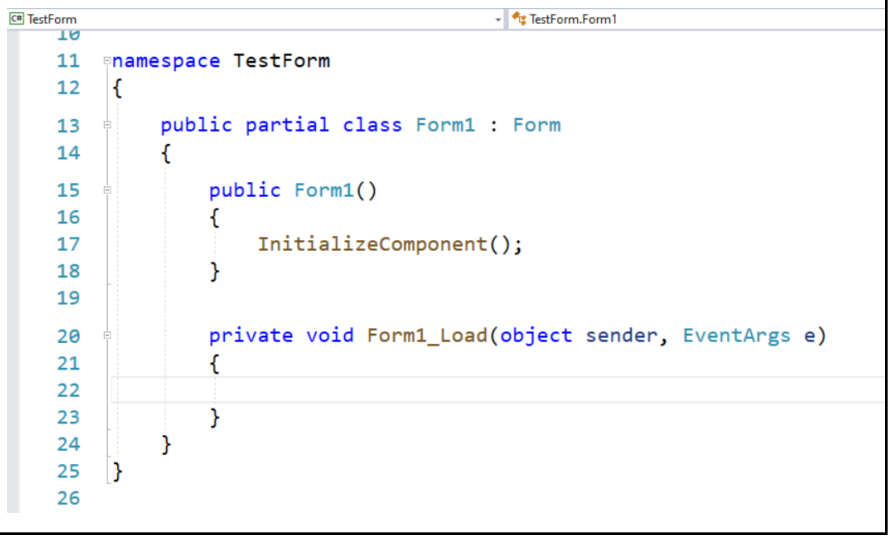
You have learned all about the C#, .NET Framework, and Object-Oriented Programming in the previous lab manuals. Let's move on to the next, new, and exciting concepts.

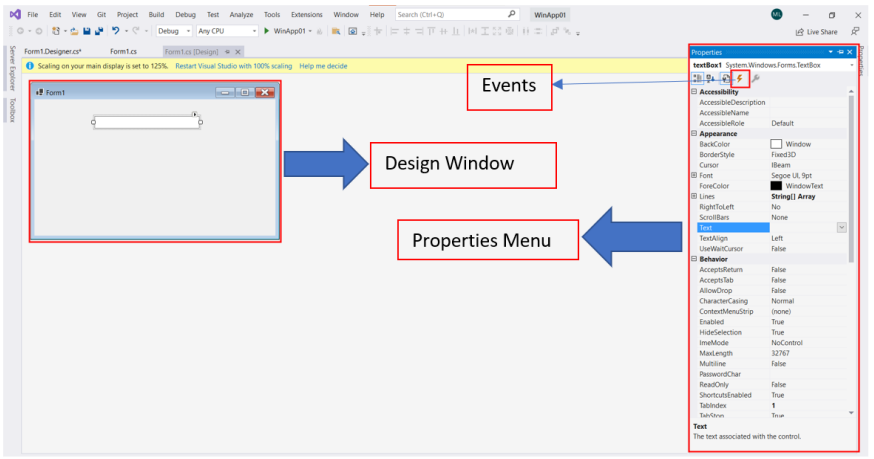
Students, in contrast to Object-Oriented Programming, there is another kind of programming paradigm that is known as **Event-Driven Programming**. Event-driven programming is a programming paradigm in which the flow of program execution is determined by events - for example, a user action such as a mouse click, keypress, or a message from the operating system or another program.
In this Lab, we will learn about the **Event-Driven Application Program** by incorporating the existing knowledge that we have learned so far.

Let's jump right into it.

S #	Description	Snapshot
1.	• Open Microsoft Visual Studio 2019. • Click on "**Create a new project**".	

2.	• Search "**Winform**" in the search bar. • Select the "**Windows Forms App (.NET Framework)**" from the list of options available. • Click next.	
3.	Rename your project	
Let's get familiar with different components of forms.		

5.	This is the "**Design Window**" that provides the graphical layout of the form.	
5(a).	It also includes the "toolbox" option on the left side of the windows that includes multiple draggable options that are dropped on the form.	
6.	Double click on the "**design view**" to open the "**code view**".	

7.	Each control component has a variety of properties and events that can be defined by using the properties menu. Note: You can access it by clicking on the properties menu on the right side of the window. Or select **view > properties**.	

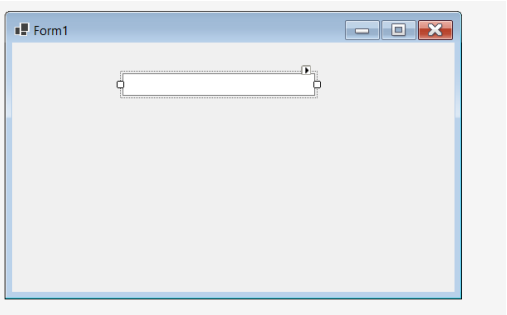
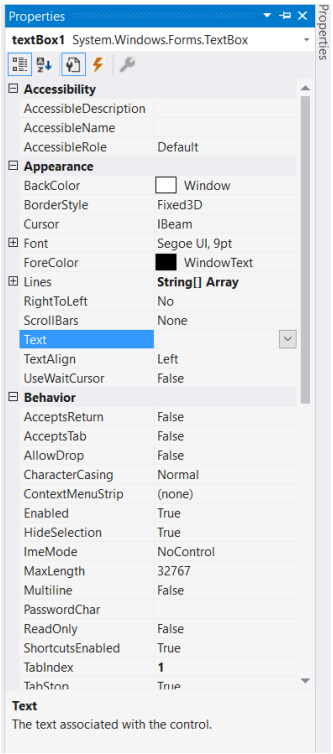
Congratulations !!!!! You have learned about the basic working windows of the forms.

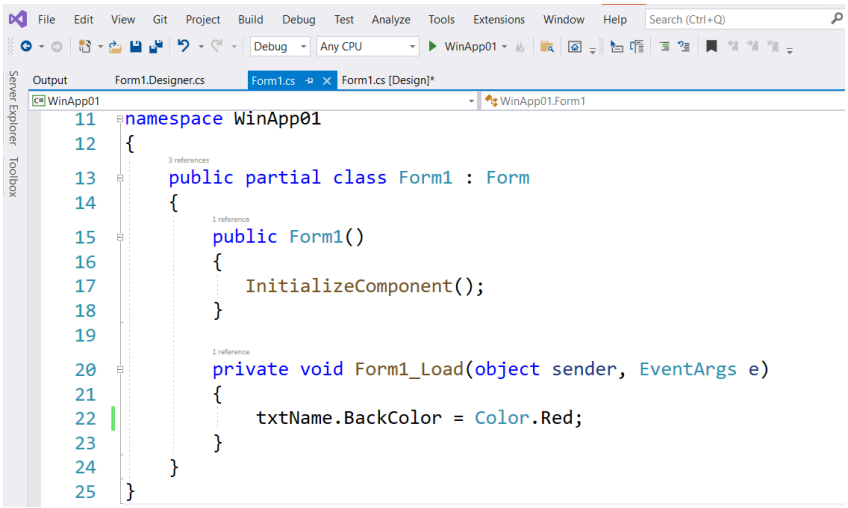
Object Oriented Programming

Lab Manual 09

Let's insert some controls and components on some forms and update their properties.

Task01: Create a textbox and set its properties by using the **properties** menu.

Sr #	Description	Snapshot
1.	Create a new project and insert a **textbox** using the toolbox menu.	
2.	Set the properties such as • Name • Back Color • Font	

3.	You can also set the properties through "**code view**".	 **output:** 
	Note: The properties defined in the design view will be defined as part of the form's design and will be visible even before the execution of the program.	On the other hand, the properties defined through code view will be visible at **run time** and these may overwrite the already existing properties.

Congratulations !!!!! You have successfully learned how to implement control components into your forms.

Now, Let's Attempt the tasks and challenges that are listed on the next page.

Self Assessment Task 01:

A Program that shows Helloworld in the Message PopUp window at a button click.

Self Assessment Task 02:

A Program that Shows HelloWorld in the message PopUp window at form Load

Self Assessment Task 03:

A Program that shows Hello world within the text box on Button Click.

Self Assessment Task 04:

Create a form that asks the user for two names and displays if they are the same or different.

Self Assessment Task 02:

Create a form with two checkboxes with text option 1 and option 2 respectively. Add a button on the form that upon clicking informs the user about the option selected.

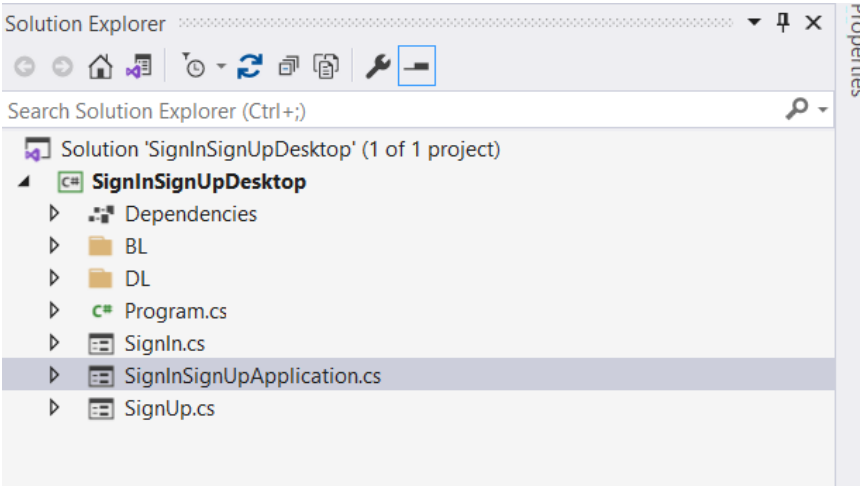
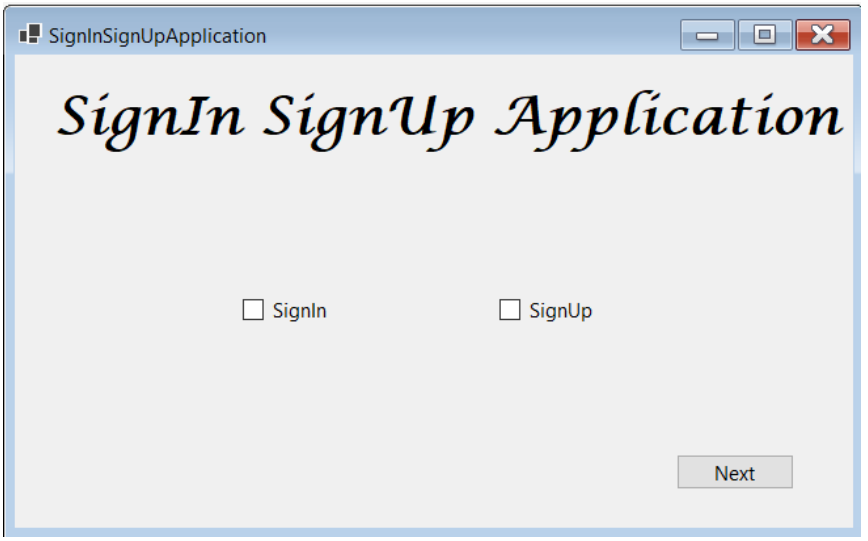
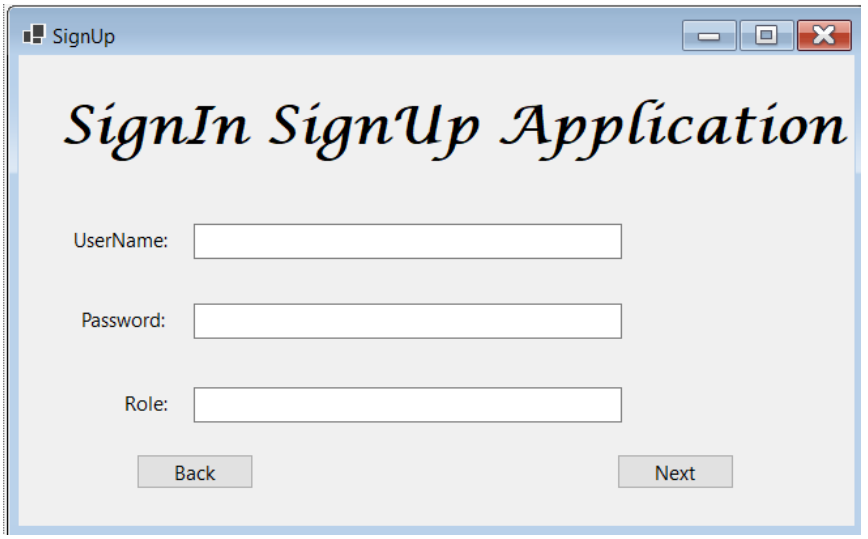
Task 01:

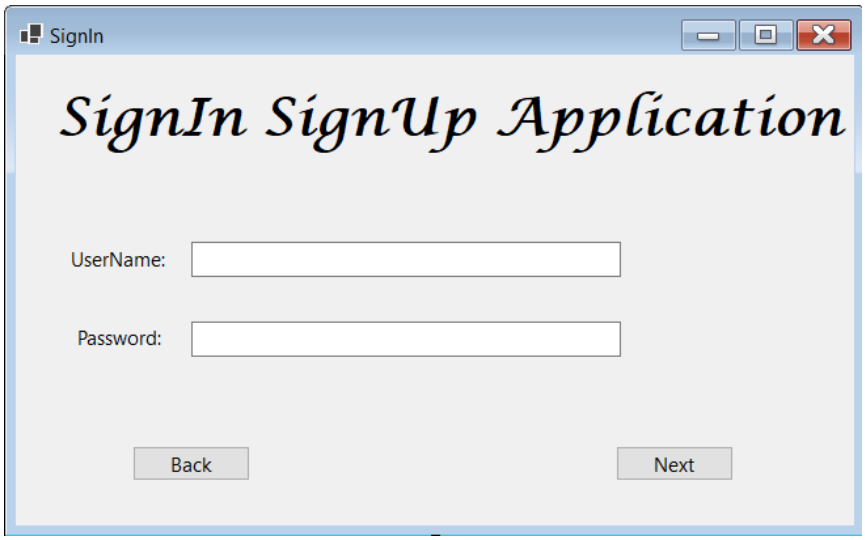
Use the learned knowledge to Convert the SignUpSignIn Application.

Sr #	Description	Snapshot

Object Oriented Programming

Lab Manual 09

1.	Create three forms by using the same method that we used to create new classes.	
2.	The first form asks the user to choose to either signIn or signUp. - Insert and Update the label - Insert and Update the check box - Insert and Update the button	
3.	Create a **SignUp** form. - Insert and Update the labels. - Insert the text boxes. - Insert and Update buttons.	

4.	Create a **signIn** form. - Insert and Update the labels. - Insert the text boxes. - Insert and Update the buttons.	
Let's Implement the back-end functionality of these forms now.		

5.	Define the MUser Class (BL)
	- Define Constructors
	- Define getters functions.
	- Define isAdmin function.
		16 references
		`class MUser`
		`{`
		`  private string userName;`
		`  private string userPassword;`
		`  private string userRole;`
		
		2 references
		`  public MUser(string userName, string userPassword, string userRole)`
		`  {`
		`  this.userName = userName;`
		`  this.userPassword = userPassword;`
		`  this.userRole = userRole;`
		`  }`
		
		1 reference
		`  public MUser(string userName, string userPassword)`
		`  {`
		`  this.userName = userName;`
		`  this.userPassword = userPassword;`
		`  this.userRole = "NA";`
		`  }`

		3 references
		public string getUserName()
		{
		return userName;
		}
		3 references
		public string getUserPassword()
		{
		return userPassword;
		}
		
		1 reference
		public string getUserRole()
		{
		return userRole;
		}
		0 references
		public bool isAdmin()
		{
		if (userRole == "Admin")
		{
		return true;
		}
		else
		{
		return false;
		}
		}

6.	Define the MUser Class (DL)
	- Define the associated functions.
		0 references
		class MUserDL
		{
		private static List<MUser> usersList = new List<MUser>();
		
		2 references
		public static void addUserIntoList(MUser user)
		{
		usersList.Add(user);
		}
		
		1 reference
		public static MUser SignIn(MUser user)
		{
		foreach (MUser storedUser in usersList)
		{
		if (storedUser.getUserName() == user.getUserName() && storedUser.getUserPassword() == user.getUserPassword())
		{
		return storedUser;
		}
		}
		return null;
		}
		
		3 references
		public static string parseData(string record, int field)
		{
		int comma = 1;
		string item = "";
		for (int x = 0; x < record.Length; x++)
		{
		if (record[x] == ',')
		{
		comma++;
		}
		else if (comma == field)
		{
		item = item + record[x];
		}
		}
		return item;
		}

```
public static bool readDataFromFile(string path)
{
    if (File.Exists(path))
    {
        StreamReader fileVariable = new StreamReader(path);
        string record;
        while ((record = fileVariable.ReadLine()) != null)
        {
            string userName = parseData(record, 1);
            string userPassword = parseData(record, 2);
            string userRole = parseData(record, 3);
            MUser user = new MUser(userName, userPassword, userRole);
            addUserIntoList(user);
        }
        fileVariable.Close();
        return true;
    }
    else
        return false;
}
```

```
public static void storeUserIntoFile(MUser user, string path)
{
    StreamWriter file = new StreamWriter(path, true);
    file.WriteLine(user.getUserName() + "," + user.getUserPassword() + "," + user.getUserRole());
    file.Flush();
    file.Close();
}
```

7.

- Read the data from the file as soon as the form starts.
- Additionally, provide the functionality in case of **click event** is triggered.

```
namespace SignInSignUpDesktop
{
    public partial class SignInSignUpApplication : Form
    {
        public SignInSignUpApplication()
        {
            InitializeComponent();
            string path = "data.txt";
            if (MUserDL.readDataFromFile(path))
            {
                MessageBox.Show("Data Loaded From the File");
            }
            else
            {
                MessageBox.Show("Data not loaded");
            }
        }

private void button1_Click(object sender, EventArgs e)
{

    if (SignIn.Checked)
    {
        Form moreForm = new SignInForm();
        moreForm.Show();
        SignIn.Checked = false;
    }
    else if(SignUp.Checked)
    {
        Form moreForm = new SignUpForm();
        moreForm.Show();
        SignUp.Checked = false;
    }

}
```

<table>
<tr>
<td>8.</td>
<td>Implement the code view for the SignUp Form.
- Implement the code for the click events of buttons “next” and “back”.</td>
<td>

```
3 references
public partial class SignUpForm : Form
{
    1 reference
    public SignUpForm()
    {
        InitializeComponent();
    }

    1 reference
    private void ClearDataFromForm()
    {
        usernameText.Text = "";
        passwordText.Text = "";
        roleText.Text = "";
    }

1 reference
private void next_Click(object sender, EventArgs e)
{

    string username = usernameText.Text;
    string password = passwordText.Text;
    string role = roleText.Text;
    string path = "data.txt";
    MUser user = new MUser(username, password, role);
    MUserDL.addUserIntoList(user);
    MUserDL.storeUserIntoFile(user, path);
    MessageBox.Show("User Added Successfully");
    ClearDataFromForm();
}

1 reference
private void back_Click(object sender, EventArgs e)
{
    this.Close();
}
```

</td>
</tr>
<tr>
<td></td>
<td>- Implement the logic in Code View for the associated events in SignIn Form.
- Provide the on-click functionality for events of “next” and “back”.</td>
<td>

```
3 references
public partial class SignInForm : Form
{
    1 reference
    public SignInForm()
    {
        InitializeComponent();
    }

    1 reference
    private void ClearDataFromForm()
    {
        usernameText.Text = "";
        passwordText.Text = "";
    }
```

</td>
</tr>
</table>

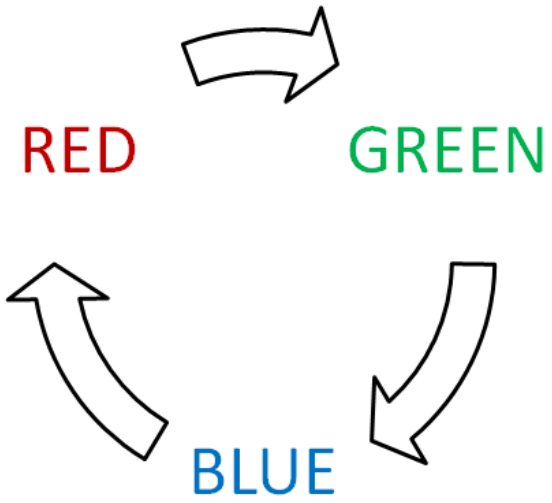
```
1 reference
private void button1_Click(object sender, EventArgs e)
{
    string username = usernameText.Text;
    string password = passwordText.Text;
    MUser user = new MUser(username, password);
    MUser validUser = MUserDL.SignIn(user);
    if (validUser != null)
    {
        MessageBox.Show("User is Valid");
    }
    else
    {
        MessageBox.Show("User is Invalid");
    }
    ClearDataFromForm();
}

1 reference
private void Back_Click(object sender, EventArgs e)
{
    this.Close();
}
```

Congratulations !!!! you have implemented your first complete project by using the windows forms.

Great Work students !!!!

Challenge 01:

Create a graphical interface that shows a text box and two buttons with next and previous labels. When the user clicks on the next button it sets the textbox background color to the next color from the loop as given below and if the user presses the next button again it sets the background color of the text box with the next color. In case the user presses back it sets the previous color to the background of the textbox.



Good Luck and Best Wishes !!

Happy Coding ahead :)