# WinForms UI - Continue

- Multiple Forms
- Passing Data Between Forms
- Logic Layer Separation
- Events

# Multiple Forms

- Create Form2:

  Right-click → Add → Form

- Open it:

```
Form2 f = new Form2();
f.Show(); // non-modal (main form still usable)
f.ShowDialog(); // modal (main form cannot
                //  be used until Form2 closes)
```

# Passing Data Between Forms

- Constructor Injection
- Public Properties (not recommended)
- Public Methods

# Passing Data Between Forms: Constructor Injection

- Pass data when creating the form
- Example:

  public Form2(string name) { lblUser.Text = name; }
- Usage:

  var f = new Form2(textBox1.Text); f.Show();

# Passing Data Between Forms: Public Properties (not recommended)

- Set a property after creating the form
- Example:

```
Form2 f = new Form2();
f.UserName = textBox1.Text;
f.Show();
```

# Passing Data Between Forms: Public Methods

- Call a method to pass data explicitly
- Example:

```
Form2 f = new Form2();
f.SetUserName(textBox1.Text);
f.Show();
```

# Logic Layer Separation

- Do NOT put logic in UI
- Create logic layer classes:
  - For example create a new logic layer class:
    - class Calculator { int Add(a,b); }
- UI calls logic layer

# Events in WinForms (MouseDown & MouseUp Example)

- **What are Events?**
- Events are **actions triggered by the user or system**
- Common events for controls:
  - **Click** → single click
  - **MouseDown / MouseUp** → mouse button pressed/released
  - **MouseEnter / MouseLeave** → cursor enters/leaves control
  - **KeyPress / KeyDown** → keyboard actions

# Events in WinForms (MouseDown & MouseUp Example)

- **How to Attach Events**
- **Select the control in Design View**
- Open **Properties window (F4)**
- Click the **Events icon (⚡)**
- Double-click next to **MouseDown** or **MouseUp** to generate handlers

# Events in WinForms (MouseDown & MouseUp Example)

- MouseDown & MouseUp Example:
- Goal: Change the button color when pressed, then restore the original color when released.

```
// Step 1: Store the original color
Color originalColor;

public Form1()
{
   InitializeComponent();
   originalColor = btnHello.BackColor; // save design-time color
}

// Step 2: Change color on MouseDown
private void btnHello_MouseDown(object sender, MouseEventArgs e)
{
   btnHello.BackColor = Color.DarkBlue; // pressed effect
}

// Step 3: Restore color on MouseUp
private void btnHello_MouseUp(object sender, MouseEventArgs e)
{
   btnHello.BackColor = originalColor; // restore original color
}
```

# Sending Email

- MailMessage message = new MailMessage(fromEmail, toEmail, subject, body);
- SmtpClient smtp = new SmtpClient("smtp.gmail.com", 587);//587, 465, 25, 2525
- smtp.EnableSsl = true;
- smtp.Credentials = new NetworkCredential(fromEmail, fromPassword);
- smtp.Send(message);

- **Explanation:**
- NetworkCredential is a class that **holds a username and password** for authentication.
- fromEmail → your Gmail email address
- fromPassword → **App Password** for Gmail (not your regular password if 2FA is enabled)
- This tells Gmail **you are allowed to send email from this account**.
- ⚠ Important: For Gmail, you **must use an App Password** if 2FA is on:

https://myaccount.google.com/apppasswords