BUDGETAPP

A Budget Management System

Giorgio Plescia, Oleksii Redko

A Personal Budget Application

- **Keep Track Of Spending**
  - Manage a better Income
    - Plan out investments
      - Cut out useless funds
        - Create a better future

Project Overview

In Depth Features

Challenges and Solutions

- ## Import Scanner Image to DB

- ## Export Scanner Image from DB

```csharp
public void AddPictures(Byte[] p, int Id)
{

    SqlCommand insertCOmmand = new SqlCommand("UPDATE Records SET Document = @Document WHERE RecordId = @RecordId", conn);
    SqlParameter binParam = new SqlParameter("Document", SqlDbType.VarBinary);
    binParam.Value = p;
    insertCOmmand.Parameters.Add(binParam);
    insertCOmmand.Parameters.Add(new SqlParameter("RecordId", Id));
    insertCOmmand.ExecuteNonQuery();


}

1 reference | BH210-13\ipd, 5 hours ago | 1 author, 1 change
public byte[] GetPicture(int id)
{

    var cmd = new SqlCommand("SELECT Document FROM Records WHERE RecordId = @ID", conn);

    cmd.Parameters.AddWithValue("@ID", id);
    return cmd.ExecuteScalar() as byte[];
```

```csharp
private void OpenImage_Click(object sender, EventArgs e)
{
    db = new Database();
    Image x = (Bitmap)((new ImageConverter().ConvertFrom(db.GetPicture(77))));
    pictureBox1.Image = x;


}
```

```csharp
private void SaveImage_Click(object sender, EventArgs e)
{
    OpenFileDialog dlg = new OpenFileDialog();

    if (dlg.ShowDialog() == DialogResult.OK)
    {
        pictureBox1.Image = Image.FromFile(dlg.FileName);
    }

    db = new Database();
    System.Drawing.Image scanpic = pictureBox1.Image;

    // System.Drawing.Image image;
    System.IO.MemoryStream imageStream;
    byte[] imageBytes;

    imageStream = new System.IO.MemoryStream();
    scanpic.Save(imageStream, System.Drawing.Imaging.ImageFormat.Jpeg);
    imageBytes = imageStream.ToArray();
    //  VARBINARY(MAX)
    db.AddPictures(imageBytes, 77);
```
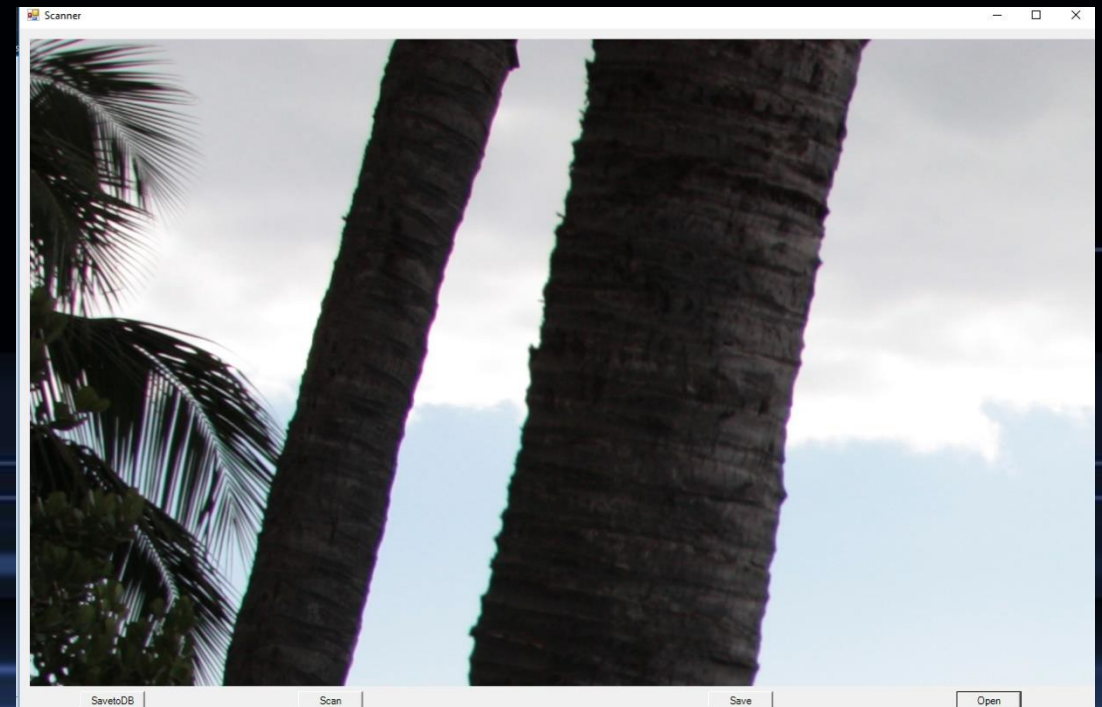
Challenges and Solutions

- Chart creation and Linking



```csharp
public void ShowAccountCart()
{
    Func<ChartPoint, string> labelPoint = chartPoint =>
        string.Format(" {0} ({1:P})", chartPoint.Y, chartPoint.Participat

    foreach (var entry in db.getBalance())
    {
        pcDebitAccouts.Series.Add(new PieSeries
        {
            Title = entry.Key,
            Values = new ChartValues<double> { entry.Value },
            DataLabels = true,
            LabelPoint = labelPoint

        });
        pcDebitAccouts.LegendLocation = LegendLocation.Bottom;
    }
}
```

```csharp
Record rc = new Record();

for (rc.Amount = db.GetBalanceById(77); rc.Amount < MaxX; rc.Amount += 0.1)
{
    chart.Series["Chart"].Points.AddXY(rc.Amount, Math.Sin(rc.Amount) / rc.Amount);

}
chart.Series["Chart"].LegendText = "Projected Increase";

/   legend
chart.Legends.Add("MyLegend");
chart.Legends["MyLegend"].BorderColor = Color.Red;
Controls.Add(this.chart);
```
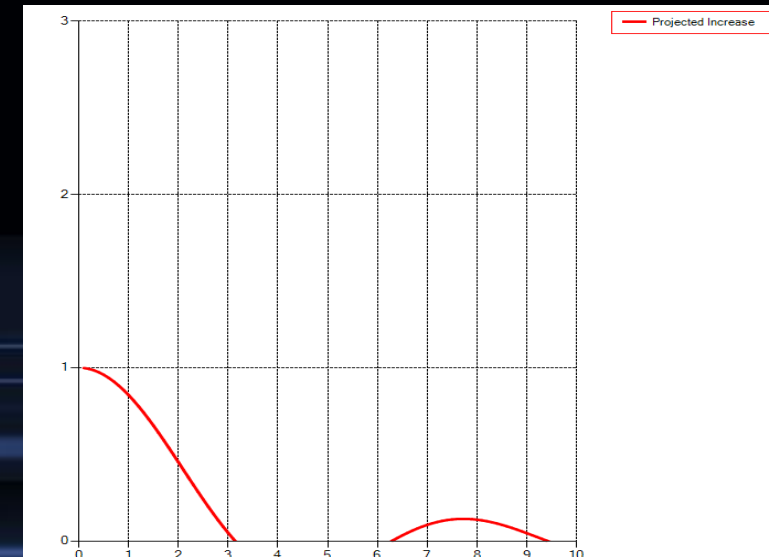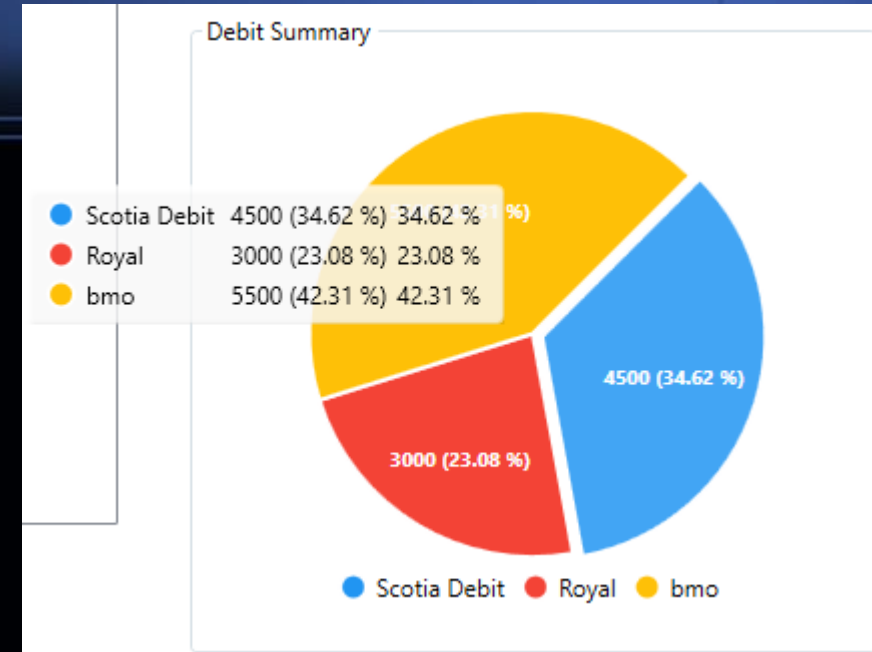
Challenges and Solutions

- Functional Api's

```csharp
string url = "";
if (txtSymbol1.Text != "") url += txtSymbol1.Text + "+";
if (txtSymbol2.Text != "") url += txtSymbol2.Text + "+";
if (txtSymbol3.Text != "") url += txtSymbol3.Text + "+";
if (txtSymbol4.Text != "") url += txtSymbol4.Text + "+";
if (url != "")
{

    url = url.Substring(0, url.Length - 1);
    const string base_url =
        "http://download.finance.yahoo.com/d/quotes.csv?s=@&f=sl1d1t1c1";
    url = base_url.Replace("@", url);
    try
    {

        string result = GetWebResponse(url);
        Console.WriteLine(result.Replace("\\r\\n", "\r\n"));

        string[] lines = result.Split(
            new char[] { '\r', '\n' },
            StringSplitOptions.RemoveEmptyEntries);
        lbStock1.Content = decimal.Parse(lines[0].Split(',')[1]).ToString("C3");
        lbStock2.Content = decimal.Parse(lines[1].Split(',')[1]).ToString("C3");
        lbStock3.Content = decimal.Parse(lines[2].Split(',')[1]).ToString("C3");
        lbStock4.Content = decimal.Parse(lines[3].Split(',')[1]).ToString("C3");
    }
```
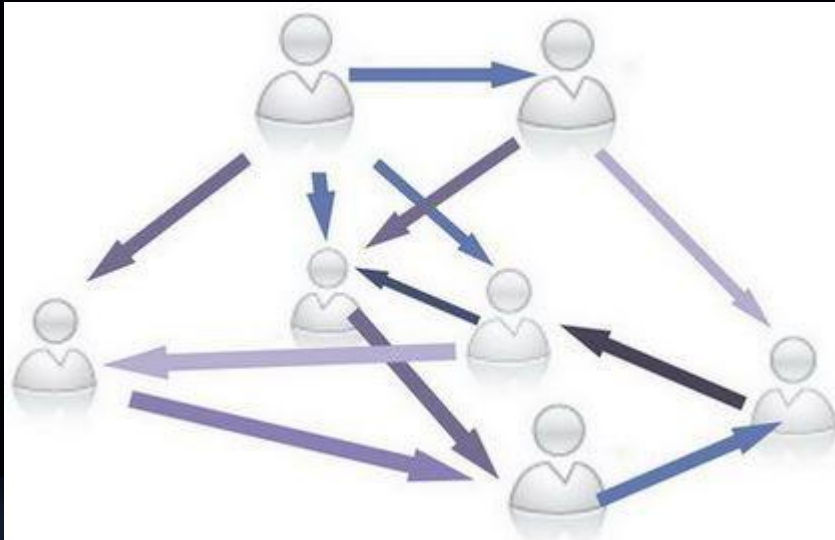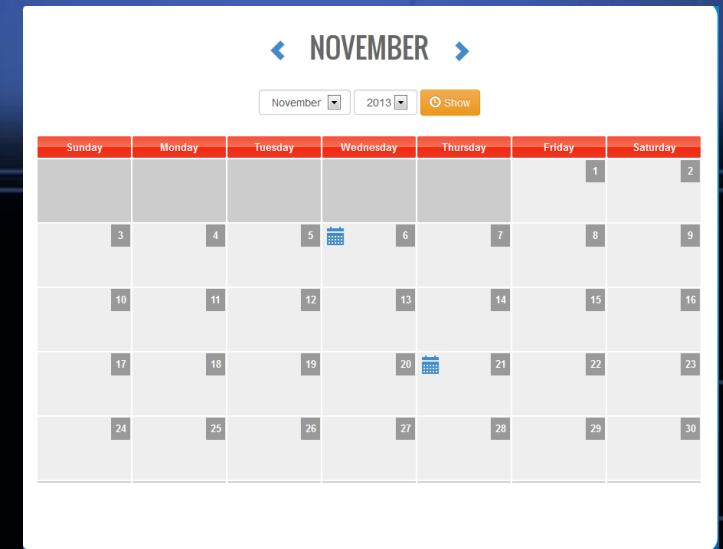
Challenges and Solutions

- SQL Many to Many Implementation

Future Work

- Advanced Search
- ASP.NET Event Calendar
- Improved Charts
- Nicer User Interface

Summary

- Our Completed Goals
- Our Experience
- Conclusion

Thank you!