Graphical user interface, text

Description automatically generatedPublic Class Form1

'AUTHOR: Thomas Tran

'TITLE: Stock Portfolio Application (Menu)

'VERSION: v1.0

'OTHER INFO:

'--this program was developed using the 4.6.1 framework

'--it relies on a YahooFinanceAPI and other packages

'--button hides this form and switches to basic form

Private Sub btnBasicForm\_Click(sender As Object, e As EventArgs) Handles btnBasicForm.Click

Form2.Show()

Me.Hide()

End Sub

'--button hides this form and switches to advanced form

Private Sub btnAdvancedForm\_Click(sender As Object, e As EventArgs) Handles btnAdvancedForm.Click

Form3.Show()

Me.Hide()

End Sub

'--button hides this form and switches to sorting form

Private Sub btnSortingForm\_Click(sender As Object, e As EventArgs) Handles btnSortingForm.Click

Form4.Show()

Me.Hide()

End Sub

'--button hides this form and switches to the future value form

Private Sub btnFutureValueForm\_Click(sender As Object, e As EventArgs) Handles btnFutureValueForm.Click

Form5.Show()

Me.Hide()

End Sub

'--button hides this form and switches to the present value form

Private Sub btnPresentValueForm\_Click(sender As Object, e As EventArgs) Handles btnPresentValueForm.Click

Form6.Show()

Me.Hide()

End Sub

'--exit button to close down the application, with a confirmation message

Private Sub btnExit\_Click(sender As Object, e As EventArgs) Handles btnExit.Click

Dim confirm\_msg As Integer

confirm\_msg = MessageBox.Show("Are you sure you want to exit?", "Exit and Close",

MessageBoxButtons.YesNo, MessageBoxIcon.Information)

If confirm\_msg = vbYes Then

Me.Close()

Else

Exit Sub

End If

End Sub

End Class

Graphical user interface, text, application, email

Description automatically generatedOption Strict On

'--accesses API that imports yahoo finance data (yahoo disabled their own service years ago)

Imports YahooFinanceApi

Public Class Form2

'AUTHOR: Thomas Tran

'TITLE: Stock Portfolio Application (Basic)

'VERSION: v1.0

Dim blnInputDataOK As Boolean = False

'--validates existence of symbols, type/range check not suitable as some stocks include numbers (e.g. A2M.AX)

Private Sub DoInputValidation()

'--existence check

If (String.IsNullOrEmpty(txtTickers.Text)) Then

MessageBox.Show("Enter a symbol", "Error")

txtTickers.Focus()

Else

blnInputDataOK = True

End If

End Sub

Private Async Sub btnGetPrices\_Click(sender As Object, e As EventArgs) Handles btnGetPrices.Click

'--run validation subroutine

blnInputDataOK = False

DoInputValidation()

If blnInputDataOK = True Then

'--send an async query so program doesn't have to wait for internet response

Dim securities = Await Yahoo.

Symbols(txtTickers.Text.Split(","c)).

Fields(Field.Symbol, Field.RegularMarketPrice, Field.RegularMarketTime, Field.Currency, Field.LongName).

QueryAsync()

DisplayResults(securities)

End If

End Sub

Private Sub DisplayResults(securities As IReadOnlyDictionary(Of String, Security))

'--adds a blank line to ease reading the results

If lstResults.Items.Count > 0 Then lstResults.Items.Add(String.Empty)

'--loops through the results

For Each objSecurity As Security In securities.Values

Dim strCurrency As String

Try

'--the currency is not always available, so this catches exceptions

strCurrency = objSecurity.Currency

Catch

strCurrency = "(unknown)"

End Try

Dim strLongName As String

Try

strLongName = objSecurity.LongName

Catch

strLongName = "(unknown)"

End Try

Dim dtLatestPrice As DateTime = UnixTimeStampToDateTime(objSecurity.RegularMarketTime)

lstResults.Items.Add($"{objSecurity.Symbol} @ {objSecurity.RegularMarketPrice} {strCurrency} - {dtLatestPrice} - {strLongName}")

Next

lstResults.SelectedIndex = lstResults.Items.Count - 1

End Sub

Public Shared Function UnixTimeStampToDateTime(ByVal unixTimeStamp As Long) As DateTime

'--unix timestamp is seconds past epoch

Dim dtDateTime As DateTime = New DateTime(1970, 1, 1, 0, 0, 0, 0, DateTimeKind.Utc)

dtDateTime = dtDateTime.AddSeconds(unixTimeStamp).ToLocalTime()

Return dtDateTime

End Function

Private Sub btnMenu\_Click(sender As Object, e As EventArgs) Handles btnMenu.Click

Form1.Show()

Me.Hide()

End Sub

End Class

Graphical user interface, text, application, email

Description automatically generatedImports System.IO

Imports System.Text

'--accesses API that imports yahoo finance data (yahoo disabled their own service years ago)

Imports YahooFinanceApi

Public Class Form3

'--AUTHOR: Thomas Tran

'--TITLE: Stock Portfolio Application (Advanced)

'--VERSION: v1.0

Dim blnInputDataOK As Boolean = False

Private Sub Form3\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'--example tickers for testing

txtTickers.Text = "DHHF.AX,VTS.AX,NDQ.AX,VAS.AX,TSLA,AAPL,GOOG,ETH-USD,BTC-USD,MSFT"

'--add columns To listview

lsvStocks.Columns.Add("Symbol", 60, HorizontalAlignment.Left) 'Column 0

lsvStocks.Columns.Add("Price", 100, HorizontalAlignment.Left) 'Column 1

lsvStocks.Columns.Add("Change", 80, HorizontalAlignment.Left) 'Column 2

lsvStocks.Columns.Add("Change %", 80, HorizontalAlignment.Left) 'Column 3

lsvStocks.Columns.Add("Time", 150, HorizontalAlignment.Left) 'Column 4 - used in Function GetAvg

lsvStocks.Columns.Add("Volume", 80, HorizontalAlignment.Left) 'Column 5

lsvStocks.Columns.Add("52 Week Low", 90, HorizontalAlignment.Left) 'Column 6

lsvStocks.Columns.Add("52 Week High", 90, HorizontalAlignment.Left) 'Column 7

lsvStocks.Columns.Add("200 Day Average", 100, HorizontalAlignment.Left) 'Column 8

lsvStocks.Columns.Add("200 Day Change", 100, HorizontalAlignment.Left) 'Column 9

lsvStocks.Columns.Add("200 Day Change %", 110, HorizontalAlignment.Left) 'Column 10

lsvStocks.Columns.Add("Long Name", 250, HorizontalAlignment.Left) 'Column 11

End Sub

'--validates existence of symbols, type/range check not suitable as some stocks include numbers (e.g. A2M.AX)

Private Sub DoInputValidation()

'--existence check

If (String.IsNullOrEmpty(txtTickers.Text)) Then

MessageBox.Show("Enter a symbol", "Error")

txtTickers.Focus()

Else

blnInputDataOK = True

End If

End Sub

'--retrives stock data from the internet after validation, then runs Private Sub DisplayResults to display

Private Async Sub btnGetPrices\_Click(sender As Object, e As EventArgs) Handles btnGetPrices.Click

'--runs validation subroutine

blnInputDataOK = False

DoInputValidation()

If blnInputDataOK = True Then

'--send an async query so program doesn't have to wait for internet response

Dim securities = Await Yahoo.

Symbols(txtTickers.Text.Split(","c)).

Fields(Field.Symbol,

Field.RegularMarketPrice,

Field.Currency,

Field.RegularMarketChange,

Field.RegularMarketChangePercent,

Field.RegularMarketTime,

Field.RegularMarketVolume,

Field.TwoHundredDayAverage,

Field.TwoHundredDayAverageChange,

Field.TwoHundredDayAverageChangePercent,

Field.FiftyTwoWeekLow,

Field.FiftyTwoWeekHigh,

Field.LongName).

QueryAsync()

DisplayResults(securities)

End If

End Sub

'--outputs stock data to listview as part of button press and saves values for sorting

Private Sub DisplayResults(securities As IReadOnlyDictionary(Of String, Security))

'--clears old tickers (prevents duplicates when sorting later)

lsvStocks.Items.Clear()

'--creates a textfile for only ticker and price for later use

Dim PricesFile As New IO.StreamWriter("stonkprices.txt")

'--loops through the results

For Each objSecurity As Security In securities.Values

Dim strCurrency As String

Try

'--the currency is not always available, so this catches exceptions

strCurrency = objSecurity.Currency

Catch

strCurrency = "(unavailable)"

End Try

Dim strLongName As String

'--the long name is not available for cryptocurrencies, so this catches exceptions

Try

strLongName = objSecurity.LongName

Catch

strLongName = "(unavailable)"

End Try

Dim dtLatestPrice As DateTime = UnixTimeStampToDateTime(objSecurity.RegularMarketTime)

'--adds stock information to listview

lsvStocks.Items.Add(New ListViewItem(New String() {

$"{objSecurity.Symbol}",

$"{objSecurity.RegularMarketPrice} {strCurrency}",

$"{objSecurity.RegularMarketChange}",

$"{objSecurity.RegularMarketChangePercent}",

$"{dtLatestPrice}",

$"{objSecurity.RegularMarketVolume}",

$"{objSecurity.FiftyTwoWeekLow}",

$"{objSecurity.FiftyTwoWeekHigh}",

$"{objSecurity.TwoHundredDayAverage}",

$"{objSecurity.TwoHundredDayAverageChange}",

$"{objSecurity.TwoHundredDayAverageChangePercent}",

$"{strLongName}"

}))

'--writes symbol and price to textfile for use in sorting form

PricesFile.WriteLine(objSecurity.Symbol)

PricesFile.WriteLine(objSecurity.RegularMarketPrice)

Next

PricesFile.Close()

End Sub

'--converts time from API for calculations

Public Shared Function UnixTimeStampToDateTime(ByVal unixTimeStamp As Long) As DateTime

'--unix timestamp is seconds past epoch

Dim dtDateTime As DateTime = New DateTime(1970, 1, 1, 0, 0, 0, 0, DateTimeKind.Utc)

dtDateTime = dtDateTime.AddSeconds(unixTimeStamp).ToLocalTime()

Return dtDateTime

End Function

'--function to get average of a listview column

Private Function GetAvg(ByVal lv As ListView, ByVal col As Integer) As Double

Dim total As Double

For Each itm As ListViewItem In lsvStocks.Items

total += CDbl(itm.SubItems(col).Text)

Next

GetAvg = total / lsvStocks.Items.Count

End Function

'--outputs total average day change % with date and performance

Private Sub btnCalculate\_Click(sender As Object, e As EventArgs) Handles btnCalculate.Click

'--clear old values if any

txtDayChange.Clear()

txtPerformance.Clear()

txtDate.Clear()

'--display current date

txtDate.Text = DateTime.Now.ToString("dd/MM/yyyy")

'--calculate average day change % from listview column using private function

Dim result As Double

result = GetAvg(lsvStocks, 3)

txtDayChange.Text = (result & "%")

'--comment if positive or negative

If result > 0 Then

txtPerformance.Text = "Positive Daily Change"

Else

txtPerformance.Text = "Negative Daily Change"

End If

End Sub

'--exports listview data to textfile

Private Sub btnExport\_Click(sender As Object, e As EventArgs) Handles btnExport.Click

'--creates textfile and cycles through listview to add each value

Using LVStream As New IO.StreamWriter("stonks.txt")

For Each LVi As ListViewItem In lsvStocks.Items

Dim LVRow As String = ""

For Each LVCell As ListViewItem.ListViewSubItem \_

In LVi.SubItems

LVRow &= LVCell.Text & ","

Next

LVRow = LVRow.Remove(LVRow.Length - 1, 1)

LVStream.WriteLine(LVRow)

Next

MessageBox.Show("Stock ListView data has been added to the textfile")

End Using

End Sub

'--returns to menu form

Private Sub btnMenu\_Click(sender As Object, e As EventArgs) Handles btnMenu.Click

Form1.Show()

Me.Hide()

End Sub

'--clears input fields and display

Private Sub btnReset\_Click(sender As Object, e As EventArgs)

txtDayChange.Clear()

txtPerformance.Clear()

txtDate.Clear()

txtTickers.Clear()

lsvStocks.Items.Clear()

End Sub

'--add back example tickers for testing

Private Sub btnRefill\_Click(sender As Object, e As EventArgs) Handles btnRefill.Click

txtTickers.Clear()

txtTickers.Text = "DHHF.AX,VTS.AX,NDQ.AX,VAS.AX,TSLA,AAPL,GOOG,ETH-USD,BTC-USD,MSFT"

End Sub

End Class

A picture containing graphical user interface

Description automatically generatedImports System.IO

Imports System.Text

Public Class Form4

'AUTHOR: Thomas Tran

'TITLE: Stock Portfolio Application (Sorting)

'VERSION: v1.0

'--outputs unsorted array from textfile

Private Sub Form4\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'--load up the Listboxes on the Form with a Heading

lstUnsorted.Items.Add("--------------------------")

lstUnsorted.Items.Add("Unsorted Array")

lstUnsorted.Items.Add("--------------------------")

lstSorted.Items.Add("--------------------------")

lstSorted.Items.Add("Sorted Array")

lstSorted.Items.Add("--------------------------")

End Sub

'--outputs textfile contents into listview

Private Sub btnStandard\_Click(sender As Object, e As EventArgs) Handles btnStandard.Click

lstContents.Items.Clear()

Dim StocksFile As StreamReader = File.OpenText("stonks.txt")

Dim Stock As String

Do While StocksFile.Peek <> -1

Stock = StocksFile.ReadLine()

lstContents.Items.Add(Stock)

Loop

StocksFile.Close()

End Sub

'--conducts selection sort for highest stock price

Private Sub btnSort\_Click(sender As Object, e As EventArgs) Handles btnSort.Click

'--load data from textfile into arrays

Try

Dim Stocks(9) As String

Dim Prices(9) As Integer

Dim StonkSortFile As IO.StreamReader

'--open the Input File to be read using the VB StreamReader

StonkSortFile = New IO.StreamReader("stonkprices.txt")

'--use a Loop to read through all of the Lines in the Input File

Dim ItemCounter As Integer = 0

Do While Not StonkSortFile.EndOfStream

Stocks(ItemCounter) = StonkSortFile.ReadLine

Prices(ItemCounter) = StonkSortFile.ReadLine

ItemCounter = ItemCounter + 1

Loop

'--close the file after finishing loop

StonkSortFile.Close()

'--display items in listbox

lstUnsorted.Items.Clear()

lstUnsorted.Items.Add("--------------------------")

lstUnsorted.Items.Add("Unsorted Array")

lstUnsorted.Items.Add("--------------------------")

Dim p As Integer

For p = 0 To ItemCounter - 1

lstUnsorted.Items.Add(Stocks(p) & ": " & Prices(p))

Next

'--apply the Selection Sort Algorithm to sort the Array into Ascending Order.

Dim lowerbound As Integer = LBound(Prices) '--start index of array

Dim upperbound As Integer = UBound(Prices) '--end Index of Array

Dim i As Integer = 0 '--Main Loop Counter for array processing

Dim j As Integer = 0 '--Index counter Value to shuffle along 1 index step

Dim intMinValueIndex As Integer '--Index to move Minimum Value to

Dim intSwapValueIndex As Integer '--Index Value that Min Value is found at

Dim intSwapValue As Integer '--Actual value of the found Minimum

Dim strSwapStock As String

'--loop to go through the array doing Selection Sort Passes.

For i = lowerbound To upperbound

intMinValueIndex = i

'--reset the saved Swap Values from the last Pass of the Array

intSwapValue = 0

intSwapValueIndex = 0

For j = i + 1 To upperbound

'--search and find the lowest value remaining in the array

If Prices(j) < Prices(intMinValueIndex) Then

'--First check for first time finding a smaller value,

'--then check if smallest so far.

If intSwapValue = 0 Then

intSwapValue = Prices(j)

strSwapStock = Stocks(j)

intSwapValueIndex = j

Else

If Prices(j) < intSwapValue Then

intSwapValue = Prices(j)

strSwapStock = Stocks(j)

intSwapValueIndex = j

End If

End If

End If

Next j

If intSwapValue <> 0 Then

Prices(intSwapValueIndex) = Prices(intMinValueIndex)

Stocks(intSwapValueIndex) = Stocks(intMinValueIndex)

Prices(intMinValueIndex) = intSwapValue

Stocks(intMinValueIndex) = strSwapStock

End If

'--Display Load of Sorted Listbox for the current Pass

'--First clear out the previous Pass display data

lstSorted.Items.Clear()

Dim strCurrentPassInfo As String = "Array Pass i = " & CStr(i)

lstSorted.Items.Add("--------------------------")

lstSorted.Items.Add(strCurrentPassInfo)

lstSorted.Items.Add("--------------------------")

For k = 0 To Stocks.Count - 1

lstSorted.Items.Add(Prices(k) & ": " & Stocks(k))

Next k

MsgBox("Click OK to continue")

Next i

'--The Array should now be fully sorted into ascending order.

'--Load up the Sorted Array into the Sorted List Box to display on the Form.

'--First clear out the previous Pass display data

lstSorted.Items.Clear()

'--Load up the Sorted Listbox on the Form with a Heading followed the Array Data.

lstSorted.Items.Add("--------------------------")

lstSorted.Items.Add("Final Sorted Array")

lstSorted.Items.Add("--------------------------")

For c = 0 To Prices.Count - 1

lstSorted.Items.Add(Prices(c) & ": " & Stocks(c))

Next

Catch ex As Exception

MsgBox("Please ensure you exported a list of exactly 10 stocks from the advanced view for this to function (or change the default examples)")

End Try

End Sub

'--returns to menu form

Private Sub btnMenu\_Click(sender As Object, e As EventArgs) Handles btnMenu.Click

Form1.Show()

Me.Hide()

End Sub

End Class

Graphical user interface

Description automatically generatedPublic Class Form5

'AUTHOR: Thomas Tran

'TITLE: Stock Portfolio Application (Future)

'VERSION: v1.0

'--sets Validation boolean as false by default

Dim blnInputDataOK = False

'--when button is clicked, runs DoValidation subroutine, then DoActions subroutine if data is correct

Private Sub btnCalculate\_Click(sender As Object, e As EventArgs) Handles btnCalculate.Click

blnInputDataOK = False

DoValidation()

'--checks if IF statements in DoValidation subroutine are fulfilled and runs DoActions

If blnInputDataOK = True Then

DoActions()

End If

End Sub

'--validation subroutine via IF statement

Private Sub DoValidation()

'--existence checks

If (String.IsNullOrEmpty(txtPresent.Text)) Then

MessageBox.Show("Enter a present value", "Error")

txtPresent.Focus()

ElseIf (String.IsNullOrEmpty(txtInterest.Text)) Then

MessageBox.Show("Enter an interest rate", "Error")

txtInterest.Focus()

ElseIf (String.IsNullOrEmpty(txtYears.Text)) Then

MessageBox.Show("Enter a number of years", "Error")

txtYears.Focus()

'--type checks

ElseIf (IsNumeric(txtPresent.Text)) = False Then

MessageBox.Show("Input numbers for present value", "Error")

txtPresent.Focus()

ElseIf (IsNumeric(txtInterest.Text)) = False Then

MessageBox.Show("Input numbers for interest", "Error")

txtInterest.Focus()

ElseIf (IsNumeric(txtYears.Text)) = False Then

MessageBox.Show("Input numbers for years", "Error")

txtYears.Focus()

'--range checks

ElseIf (txtPresent.Text <= 0) Then

MessageBox.Show("Input a value above 0 for present value", "Error")

txtPresent.Focus()

ElseIf (txtInterest.Text <= 0) Then

MessageBox.Show("Input a value above 0 for interest", "Error")

txtInterest.Focus()

ElseIf (txtYears.Text <= 0) Then

MessageBox.Show("Input a value above 0 for years", "Error")

txtYears.Focus()

'--makes InputDataOK True if all validations are correct

Else

blnInputDataOK = True

End If

End Sub

'--formula to calculate Future Value(FV)

Private Sub DoActions()

Dim FV As Double

Dim PV As Double = txtPresent.Text

Dim i As Double = txtInterest.Text

Dim n As Double = txtYears.Text

FV = PV \* (1 + i / 100) ^ n

'--rounds to 2 decimal places and outputs

txtFuture.Text = ("$" & Math.Round(FV, 2))

End Sub

'--returns to menu form

Private Sub btnMenu\_Click(sender As Object, e As EventArgs) Handles btnMenu.Click

Form1.Show()

Me.Hide()

End Sub

End Class

Graphical user interface

Description automatically generatedPublic Class Form6

'AUTHOR: Thomas Tran

'TITLE: Stock Portfolio Application (Present)

'VERSION: v1.0

'--sets Validation boolean as false by default

Dim blnInputDataOK = False

'--when button is clicked, runs DoValidation subroutine, then DoActions subroutine if data is correct

Private Sub btnCalculate\_Click(sender As Object, e As EventArgs) Handles btnCalculate.Click

blnInputDataOK = False

DoValidation()

'--checks if IF statements in DoValidation subroutine are fulfilled and runs DoActions

If blnInputDataOK = True Then

DoActions()

End If

End Sub

'--validation subroutine via IF statement

Private Sub DoValidation()

'--existence checks

If (String.IsNullOrEmpty(txtFuture.Text)) Then

MessageBox.Show("Enter a given name", "Error")

txtFuture.Focus()

ElseIf (String.IsNullOrEmpty(txtInterest.Text)) Then

MessageBox.Show("Enter an interest rate", "Error")

txtInterest.Focus()

ElseIf (String.IsNullOrEmpty(txtYears.Text)) Then

MessageBox.Show("Enter a number of years", "Error")

txtYears.Focus()

'--type checks

ElseIf (IsNumeric(txtFuture.Text)) = False Then

MessageBox.Show("Input numbers for present value", "Error")

txtFuture.Focus()

ElseIf (IsNumeric(txtInterest.Text)) = False Then

MessageBox.Show("Input numbers for interest", "Error")

txtInterest.Focus()

ElseIf (IsNumeric(txtYears.Text)) = False Then

MessageBox.Show("Input numbers for years", "Error")

txtYears.Focus()

'--range checks

ElseIf (txtFuture.Text <= 0) Then

MessageBox.Show("Input a value above 0 for present value", "Error")

txtFuture.Focus()

ElseIf (txtInterest.Text <= 0) Then

MessageBox.Show("Input a value above 0 for interest", "Error")

txtInterest.Focus()

ElseIf (txtYears.Text <= 0) Then

MessageBox.Show("Input a value above 0 for years", "Error")

txtYears.Focus()

'--makes InputDataOK True if all validations are correct

Else

blnInputDataOK = True

End If

End Sub

'--formula to calculate Present Value(PV)

Private Sub DoActions()

Dim FV As Double = txtFuture.Text

Dim PV As Double

Dim i As Double = Val(txtInterest.Text) / 100

Dim n As Double = txtYears.Text

PV = FV / (1 + i) ^ n

'--rounds to 2 decimal places and outputs

txtPresent.Text = ("$" & Math.Round(PV, 2))

End Sub

'--returns to menu form

Private Sub btnMenu\_Click(sender As Object, e As EventArgs) Handles btnMenu.Click

Form1.Show()

Me.Hide()

End Sub

End Class