Programmed Solution to a Problem - Developmental Testing

Porth-y-waen Silver Band Management System

Nia Hawkins: 7183
The Maelor School: 68146

Contents

Developmental testing

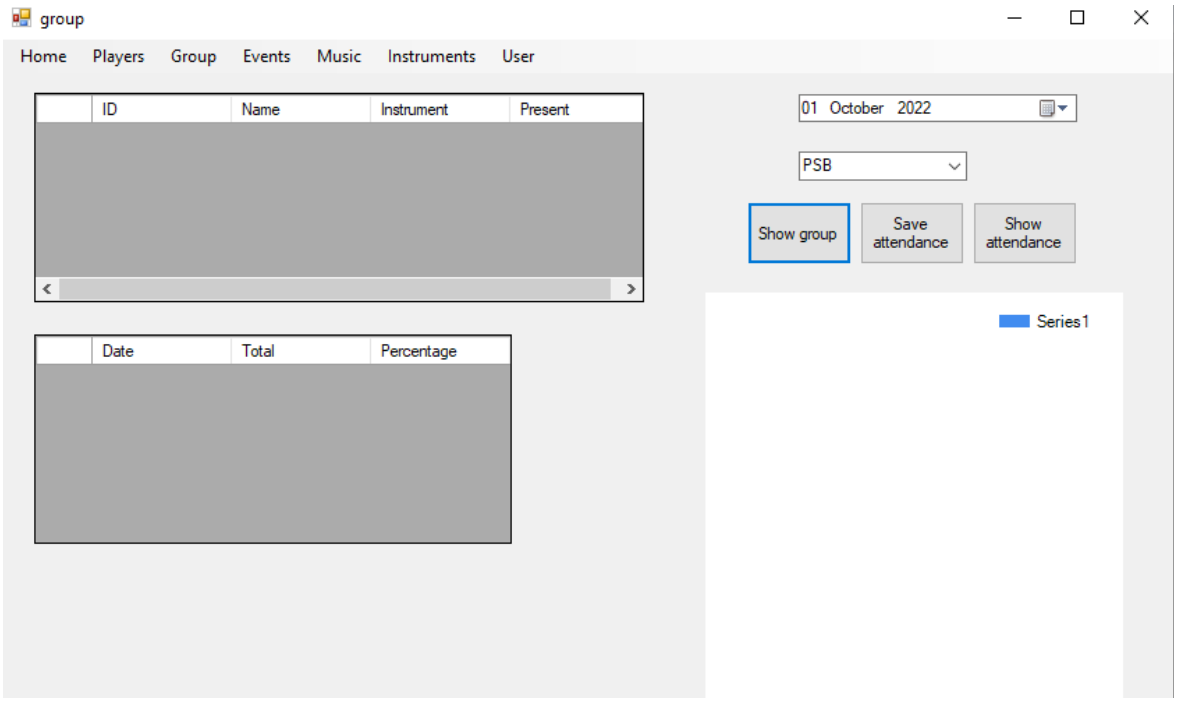
During the development of the program, I encountered many errors when testing, some more significant and complex to fix than others.

Not displaying members from a selected group

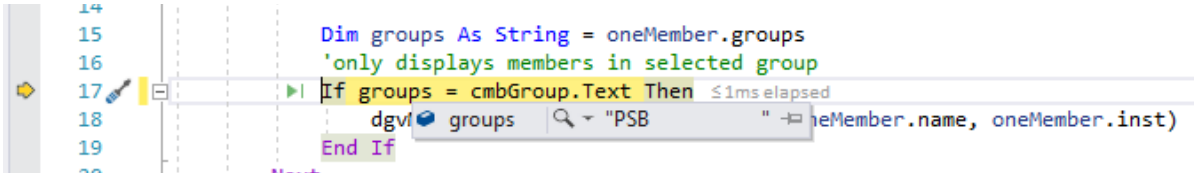
When btnShow is clicked in the groups screen, the following code will be executed to display the members of the group chosen in cmbGroup.

```
Private Sub btnShow_Click(sender As Object, e As EventArgs) Handles btnShow.Click
        Dim index As Integer
        Dim oneMember As memberInfo

        dgvMembers.Rows.Clear()
        FileOpen(1, "members.dat", OpenMode.Random,,, Len(oneMember))

        Dim totalRecords As Integer = LOF(1) / Len(oneMember)
        For index = 1 To totalRecords
                FileGet(1, oneMember)

                Dim groups As String = oneMember.groups
                If groups = cmbGroup.Text Then
                        dgvMembers.Rows.Add(oneMember.id, oneMember.name,
oneMember.inst)
                End If
        Next
        FileClose(1)
        If cmbGroup.Text = "Dep" Then
                dgvTotal.Enabled = False
                Chart1.Enabled = False
        End If
End Sub
```

When the code is run and the button is clicked, no data is displayed in dgvMembers.

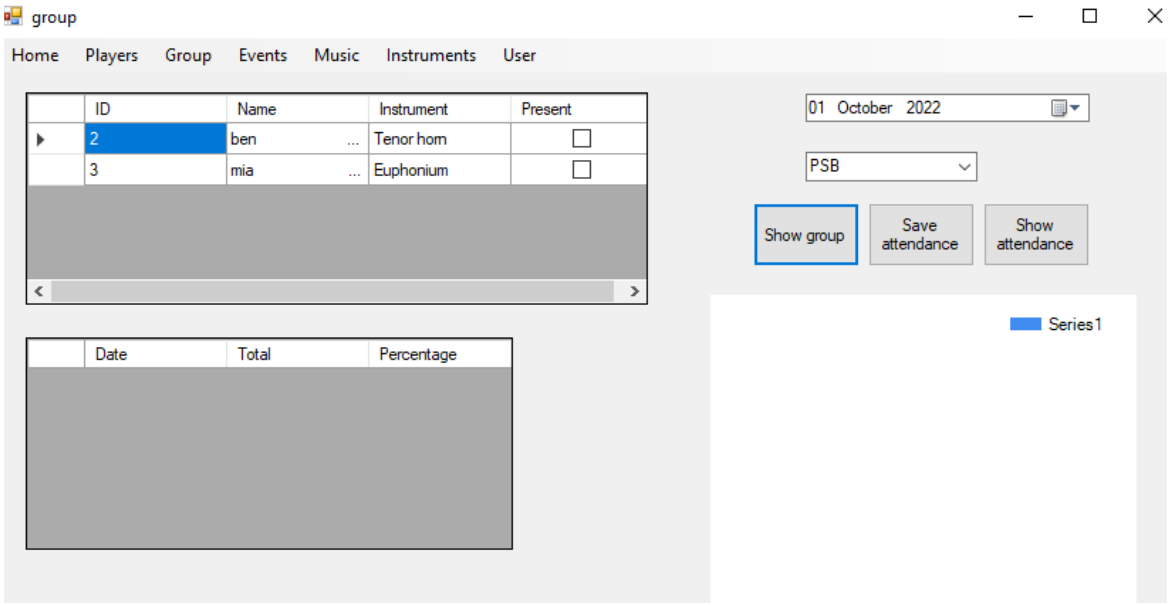


After inserting a breakpoint and stepping through the code, I discovered that *groups* has extra whitespace than *cmbGroups* so they will not equal each other. I fixed this by using *.contains* instead of =.




```
Private Sub btnShow_Click(sender As Object, e As EventArgs) Handles btnShow.Click
        Dim index As Integer
        Dim oneMember As memberInfo

        dgvMembers.Rows.Clear()
        FileOpen(1, "members.dat", OpenMode.Random,,, Len(oneMember))

        Dim totalRecords As Integer = LOF(1) / Len(oneMember)
        For index = 1 To totalRecords
            FileGet(1, oneMember)

            Dim groups As String = oneMember.groups
            If groups.Contains (cmbGroup.Text) Then
                dgvMembers.Rows.Add(oneMember.id, oneMember.name,
oneMember.inst)
            End If
        Next
        FileClose(1)
```

```
            If cmbGroup.Text = "Dep" Then
                dgvTotal.Enabled = False
                Chart1.Enabled = False
                End If
        End Sub
```



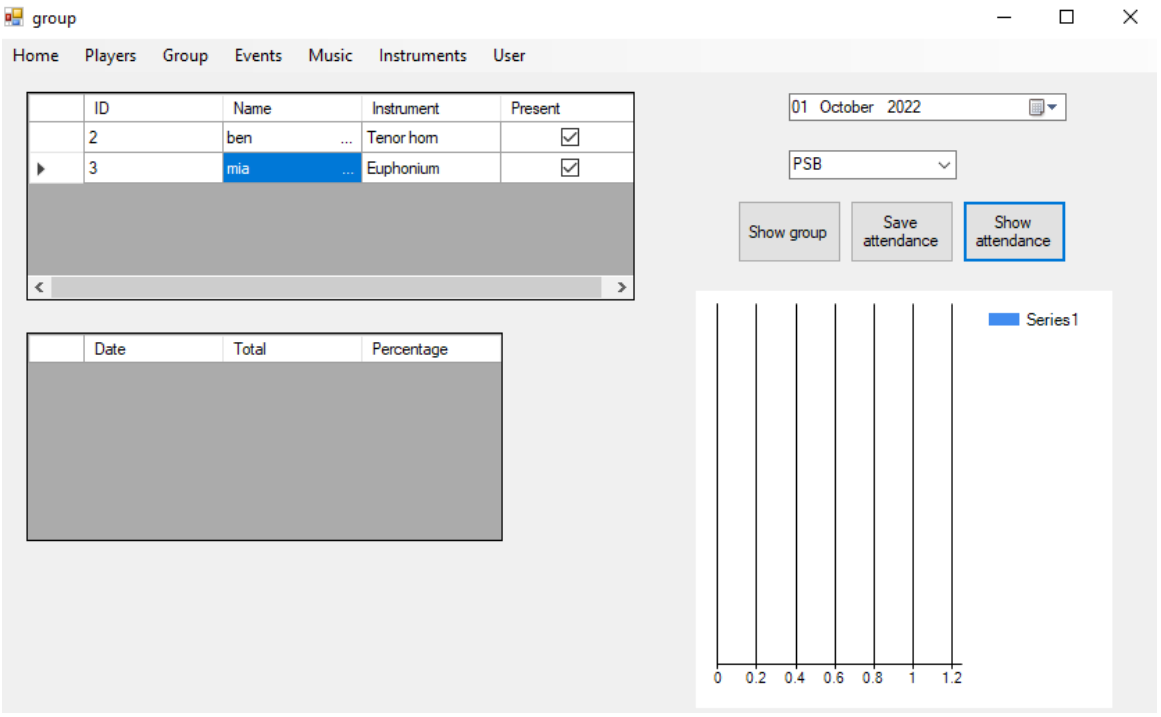
Displaying attendance data in a chart

The code below is executed when btnAttendance is clicked. It should calculate the total percentage of attendance for the date selected and display it in a chart.

```
Private Sub BtnAttendance_Click(sender As Object, e As EventArgs) Handles btnAttendance.Click
        'displaying attendance in a chart
        Dim oneMark As playerAttendance
        Dim dates As String
        Dim count As Integer
        Dim percentage As Decimal
        Dim currentDate As Date = dtpMarkDate.Text
        Chart1.Series(0).ChartType = DataVisualization.Charting.SeriesChartType.Column
        Chart1.Series(0).Points.Clear()

        FileOpen(1, "attendance.dat", OpenMode.Random,,, Len(oneMark))
        Dim recordCount As Integer = LOF(1) / Len(oneMark)

        For i = 0 To recordCount - 1
                FileGet(1, oneMark)
                If oneMark.markDate = (currentDate) Then
                        dates = dates + oneMark.markDate & ","
                ElseIf oneMark.markDate = (currentDate.AddDays(-7)) Then
                        dates = dates + oneMark.markDate
                End If
        Next
        FileClose(1)

        Dim dateArray As Array = dates.Split(",")
        FileOpen(1, "attendance.dat", OpenMode.Random,,, Len(oneMark))
        For i = 0 To recordCount - 1
                FileGet(1, oneMark)
                For Each week In dateArray
                        If week <> "" Then
                                If oneMark.markDate = week and oneMark.mark.StartsWith("True")
Then
                                        count += 1
                                        percentage = count / recordCount
                                        Chart1.Series(0).Points.AddXY(percentage, week)
                                End If
                        End If
                Next
        Next
        FileClose(1)
```

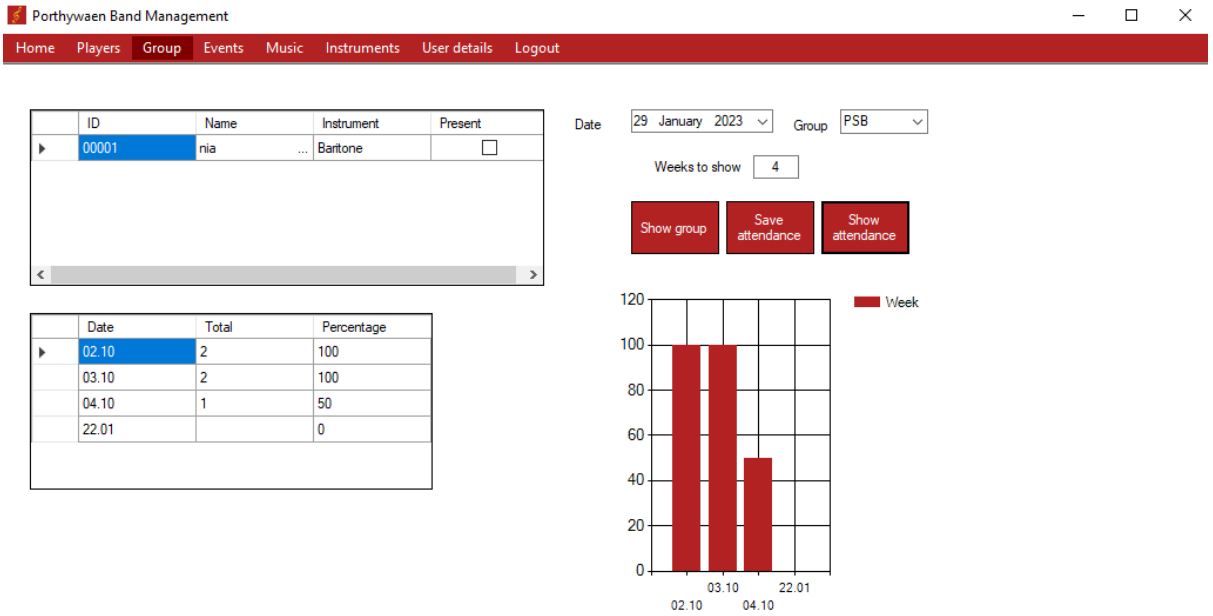
```
        End Sub
```

The code does not work as intended as no data is displayed in the chart and it displays the wrong numbers on the axis.



This issue took a long time to fix as there is little information on charts and the way the data is stored is very complex. One issue was that the charts did not recognise the data format as a date, so was reading them as fractions and converting them to decimals when displayed. This results in every date being formatted to be in a decimal format.

In the event of any errors, all the data that is added to the chart is displayed in a DataGridView so the user can see the results of the calculations.



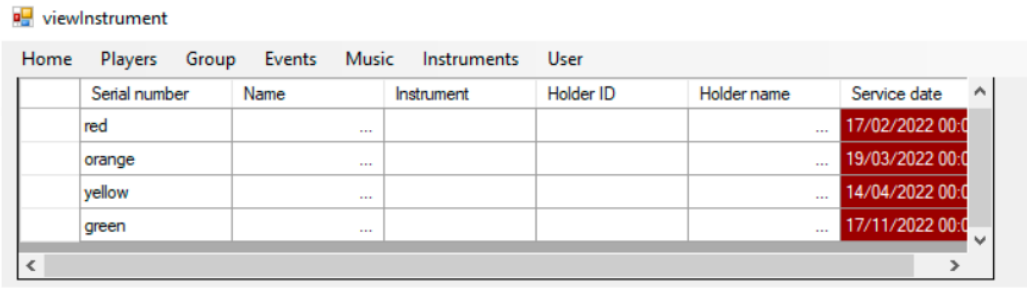
The following pseudocode correctly produces the chart.

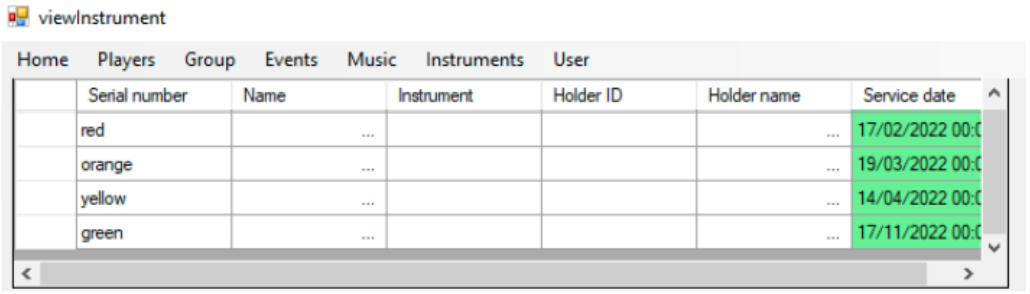
```
Private Sub BtnAttendance_Click(sender As Object, e As EventArgs) Handles btnAttendance.Click
'displaying attendance in a chart
        oneMark isplayerAttendance 'pointer
        weeks isInteger = txtWeeks.Text - 1
        percentage is Decimal
        readDate is String
        dates(5) is String
        counts(5) As String
        totalRead(5) As String

        dgvTotal.Rows.Clear()

        Chart1.Series.Clear()      'setting up the chart
        Chart1.Series.Add("Week")
        Chart1.Series("Week").ChartType = DataVisualization.Charting.SeriesChartType.Column
        Chart1.Series("Week").Points.Clear()

        FileOpen(1, "attendance.dat", OpenMode.Random,,, Len(oneMark))        'getting data
        recordCount As Integer = LOF(1) / Len(oneMark)
        For i = 0 To recordCount - 1
                FileGet(1, oneMark)
                readDate = oneMark.markDate.ToString(Format(oneMark.markDate, "dd.MM"))
'change date format so it is shown
                For j = 0 To 5     'for each possible date stored in the array
                        If dates(j) = readDate Then     'if date found at location, +1 to count
                                totalRead(j) += 1
                                If oneMark.mark.Contains("True") Then
                                        counts(j) += 1
```

```
                                End If
                                Exit For
                        ElseIf dates(j) = "" Then       'if date is not in array, add it to array and +1 to
count if true
                                totalRead(j) += 1
                                dates(j) = readDate
                                If oneMark.mark.Contains("True") Then
                                        counts(j) += 1
                                End If
                                Exit For
                        End If
                Next
        Next
        FileClose(1)

        For i = 0 To weeks      'calculate percentage
                If dates(i) <> "" Then
                        percentage = (counts(i) / totalRead(i)) * 100
                        dgvTotal.Rows.Add(dates(i), counts(i), percentage)
                End If
        Next
        For i = 0 To dgvTotal.Rows.Count - 1        'show data in chart
                Chart1.Series(0).Points.AddXY(dgvTotal.Item(0, i).Value, dgvTotal.Item(2, i).Value)
        Next
End Sub
```

Service date colours

As discussed in the post prototype refinement, service dates for instruments are coloured to indicate if an instrument is due a service based on the time since its last repairs. This feature led to many issues. Firstly, I found it difficult to create comparisons that would show the correct colours. While testing different methods to achieve this, often all outputted dates would show red or green, despite having dates that should be different colours stored in the file. The data shown below should be coloured as indicated in the serial number column.





The code below is the first working solution that correctly coloured each date.

```
Dim serviceMonth = oneInstrument.serviceDate.ToString("MM")
Dim red = Date.Today.ToString("MM") - 11
Dim yellow = Date.Today.ToString("MM") - 9
Dim orange = Date.Today.ToString("MM") - 10

If serviceMonth = red Then
        dgvInstruments.Rows(index - 1).Cells(5).Style.BackColor = Color.DarkRed
        dgvInstruments.Rows(index - 1).Cells(5).Style.ForeColor = Color.White
ElseIf serviceMonth = yellow Then
        dgvInstruments.Rows(index - 1).Cells(5).Style.BackColor = Color.Gold
        dgvInstruments.Rows(index - 1).Cells(5).Style.ForeColor = Color.Black
ElseIf serviceMonth = orange Then
        dgvInstruments.Rows(index - 1).Cells(5).Style.BackColor = Color.Orange
        dgvInstruments.Rows(index - 1).Cells(5).Style.ForeColor = Color.Black
Else
        dgvInstruments.Rows(index - 1).Cells(5).Style.BackColor = Color.LightGreen
        dgvInstruments.Rows(index - 1).Cells(5).Style.ForeColor = Color.Black
End If
```

After finding the solution above, I later discovered when testing other areas that it no longer worked. After looking at the code again, I found that if the current date and the service date are in the same year the code produced the correct results, but if not, the year was not considered.

I looked at the logic of the code and discovered the method of formatting the dates to just be the month means a date could be green if it was serviced 2 years ago. I researched other ways of subtracting months from a date and found the AddMonths method. This allows the whole date to be compared, not just the month.

The updated pseudocode for the variables that are compared to the service date is shown below.

red as date = Date.Today.AddMonths(-11)
Yellow as date = Date.Today.AddMonths(-9)
orange as date = Date.Today.AddMonths(-10)

	Instrument ID	Serial number	Name	Instrument	Holder ID	Holder name	Service date
	00003	JP6816	JP171SW ...	Comet			14/02/2022 00:0...
	00002	BE81A	Soverign ...	Comet	00001	nia ...	16/03/2022 00:0...
	00004	BE8041	International ...	Euphonium			14/04/2022 00:0...
▶	00001	YA825	Neo ...	Tenor horn	00001	nia ...	21/01/2023 00:0...

Login and password reset

I created a login form that asks the user for their name and password. When a player is first added, their password will be their player ID, which is also what the password is set to when the reset password button is clicked.

In order to reset the password, the user enters their name and clicks reset and a search is carried out to find the user and updates the password to their ID. When the button was clicked it would produce the following error.

```
    Dim oneMember As memberInfo 'pointer to structure
    FileOpen(1, "players.dat", OpenMode.Random,,, Len(oneMember))
    Dim totalRecords As Integer = LOF(1) / Len(oneMember)
    For index = 1 To totalRecords
        FileGet(1, oneMember)
        If Trim(oneMember.name) = txtName.Text Then
            oneMember.password = oneMember.id   'save password
            FilePut(1, oneMember, userRecord)  'update file
            FileClose(1)
            MsgBox("Password reset to ID", vbInformation)
        End If
    Next
    FileClose(1)
End Sub
```

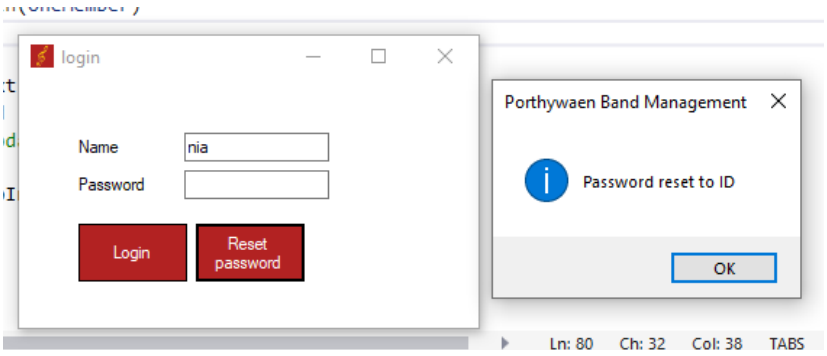
Exception Unhandled

System.ArgumentException: 'Argument 'RecordNumber' is not a valid value.'

View Details | Copy Details | Start Live Share session...

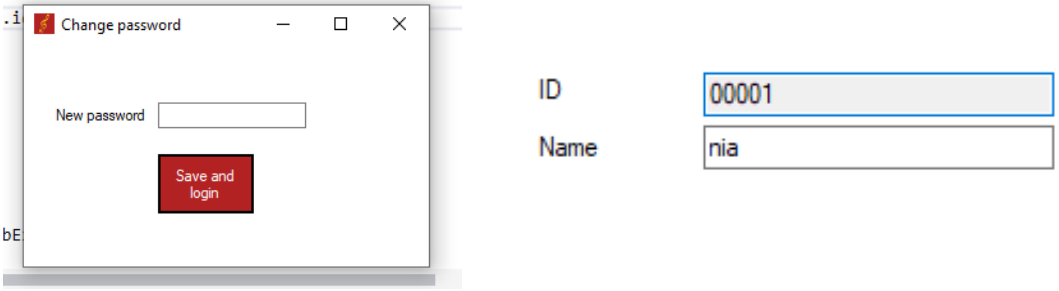
Exception Settings

This occurred as I had used the wrong variable as the record number. userRecord has not been assigned a value at this point as it is assigned a value when the login is successful. To fix this error, I replaced the variable userRecords with index.



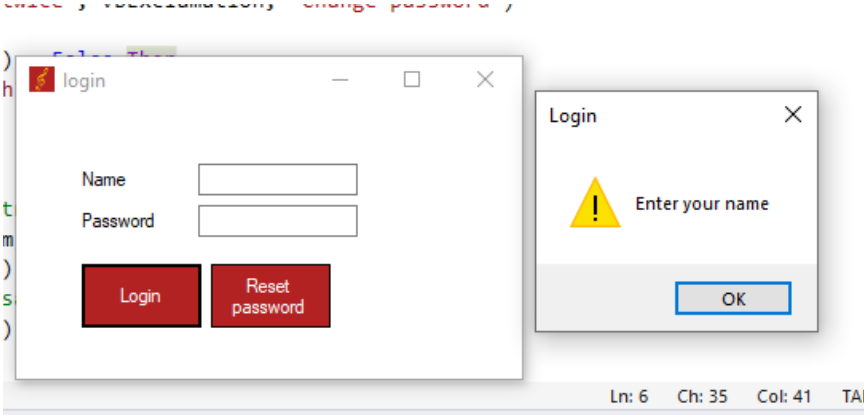
The program now worked as expected.

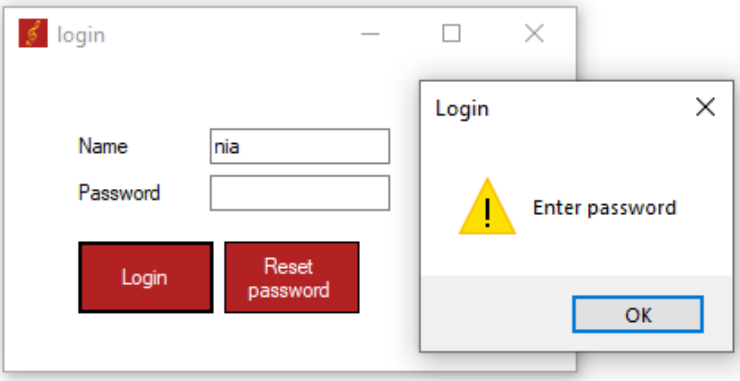
I also discovered while testing the login if the user clicked reset password and left txtName empty, nothing would happen. Similarly, if btnLogin was clicked and no text had been entered, the change password form would be shown, which was not expected. If data was entered into this textbox to change the password, the user would be logged in. When I looked at the user details from, I discovered the program had just taken the first player stored in the system.

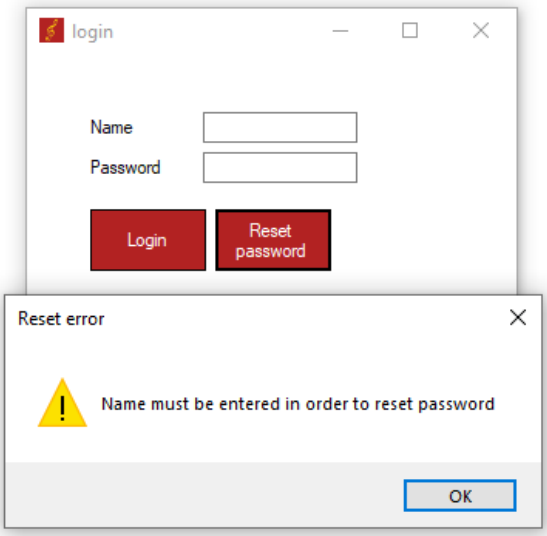


This was because I had not added any validation to these textboxes. To ensure this does not happen again, I added validation to all of these textboxes as this could be a significant issue as an unauthorised person could easily get into the system.

The following messages are now outputted if any textbox is blank.







Code running after warning message shown

After adding the validation and displaying the message, the home page was still shown. I found after adding a breakpoint and stepping through the code that the message being shown did not stop the rest of the code from being run. In order to stop this from happening, I added the line *"Exit sub"* after the messages were displayed, which stops the sub from being run any further. I noticed this issue occurs throughout the whole program so after any warning messages are displayed, the sub is now exited.

btnClear not working on frmUserDetails

This form displays the details of the user so that they are able to update their own details. btnClear should empty all textboxes on the form. However, when the button was clicked, nothing happened. This was because the subroutine that cleared the players form had been called to clear the text boxes as it has the same input fields as userDetails. I had done this in an attempt to save time and reduce the number of lines of code in the files. This will not clear the textboxes on the userDetails form. To fix this, I removed the call for the subroutine and copied the code from the subroutine into the userDetails code.

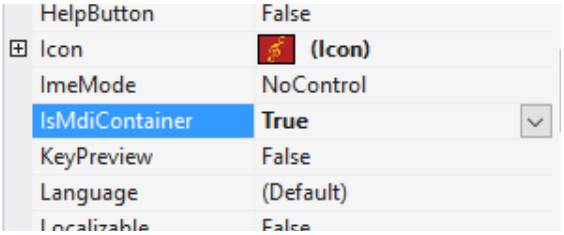
```
Private Sub btnClear_Click(sender As Object, e As EventArgs) Handles btnClear.Click
        'clear all inputs
        txtName.Clear()
        dtpDOB.Text = Today
        txtEmail.Clear()
        txtPhone.Clear()
        cmbInstrument.ResetText()
        cmbLevel.ResetText()
```

```
        chkPhotoPerm.CheckState = False
        cmbRole.ResetText()
        chkPSB.CheckState = False
        chkPYTB.CheckState = False
        chkPBB.CheckState = False
        chkStarters.CheckState = False
        txtContName.Clear()
        txtContPhone.Clear()
End Sub
```

Forms moving around the screen

Each different area of functionality; such as players or events; are in different forms. When a button or menu bar is clicked, the current form closes and the relevant form opens. When this happens, the new form does not open in the same place as the previous one.

To try to fix this, I set the sizes of all forms to be the same and gave them the same starting position. However, the problem still persisted when I tested it.



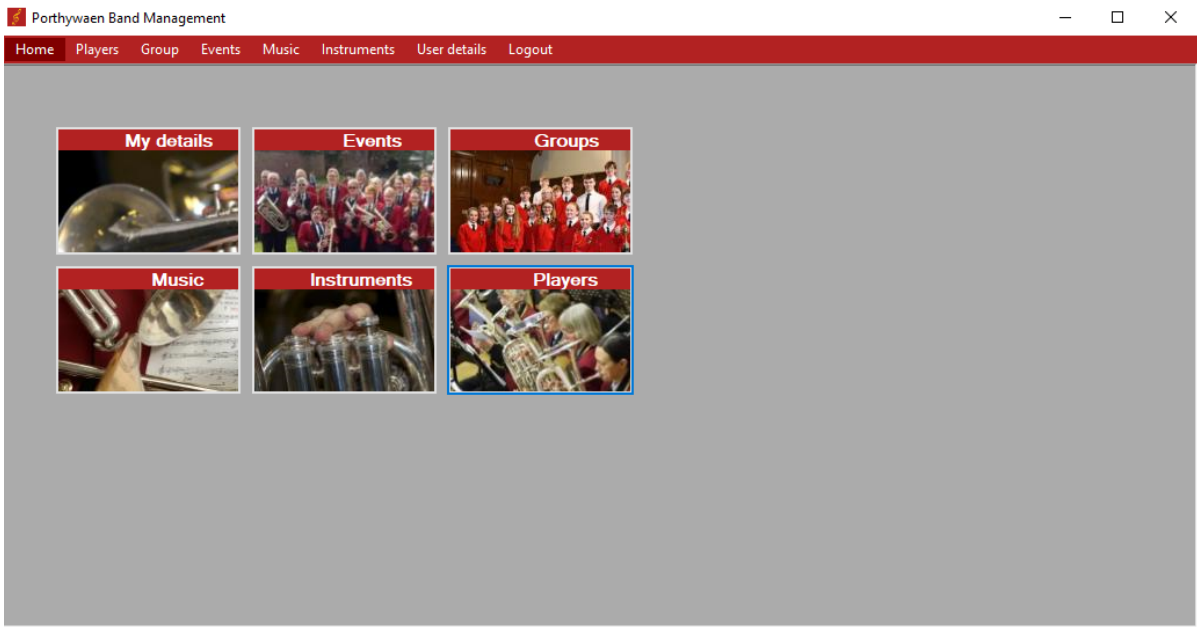
I decided to have one parent form that contains all of the other forms. This also eliminated the need for copies of the menu bar on each form. I did this by setting frmHome IsMdiContainer property to true. The following pseudocode subroutine is called every time the menu bar or a button is clicked to open a form

```
Sub SetupFormsAndMenuItems()
        frmPlayers.Hide()
        frmEvents.Hide()
        frmGroup.Hide()
        frmInstrument.Hide()
        frmMusic.Hide()
        frmUserDetails.Hide()

        btnPlayers.Hide()
        btnEvents.Hide()
        btnGroups.Hide()
        btnInstruments.Hide()
        btnMusic.Hide()
        btnUserDetails.Hide()

        HomeToolStripMenuItem.BackColor = System.Drawing.Color.Firebrick
        PlayersToolStripMenuItem.BackColor = System.Drawing.Color.Firebrick
        EventsToolStripMenuItem.BackColor = System.Drawing.Color.Firebrick
        GroupToolStripMenuItem.BackColor = System.Drawing.Color.Firebrick
        InstrumentsToolStripMenuItem.BackColor = System.Drawing.Color.Firebrick
        MusicToolStripMenuItem.BackColor = System.Drawing.Color.Firebrick
        UserToolStripMenuItem.BackColor = System.Drawing.Color.Firebrick
End sub
```

In each form, the following pseudocode is used to set the form opened within the MDI form.

```
Sub frmLoad()
        Me.MdiParent = frmHome
        Me.Dock = DockStyle.Fill
        Me.FormBorderStyle = FormBorderStyle.None
End sub
```

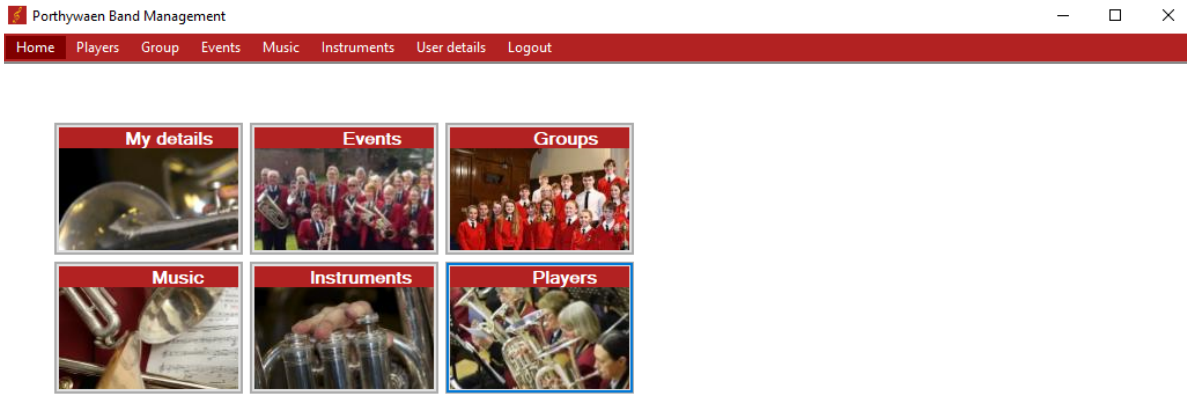
This feature introduced a new problem; the background colour of frmHome was grey, even though the background was set as white like the other forms. Nothing in the code could have done this, but the only changes made to this form was making it a MDI container.



After some research, I discovered, by default, MDI containers background colours are grey. To ensure that the background colour was what it was set as in the properties I added the following code to undo the default colour. It checks if each control in the form is a MDI, and if it is it sets the back colour as the backcolor in its properties.

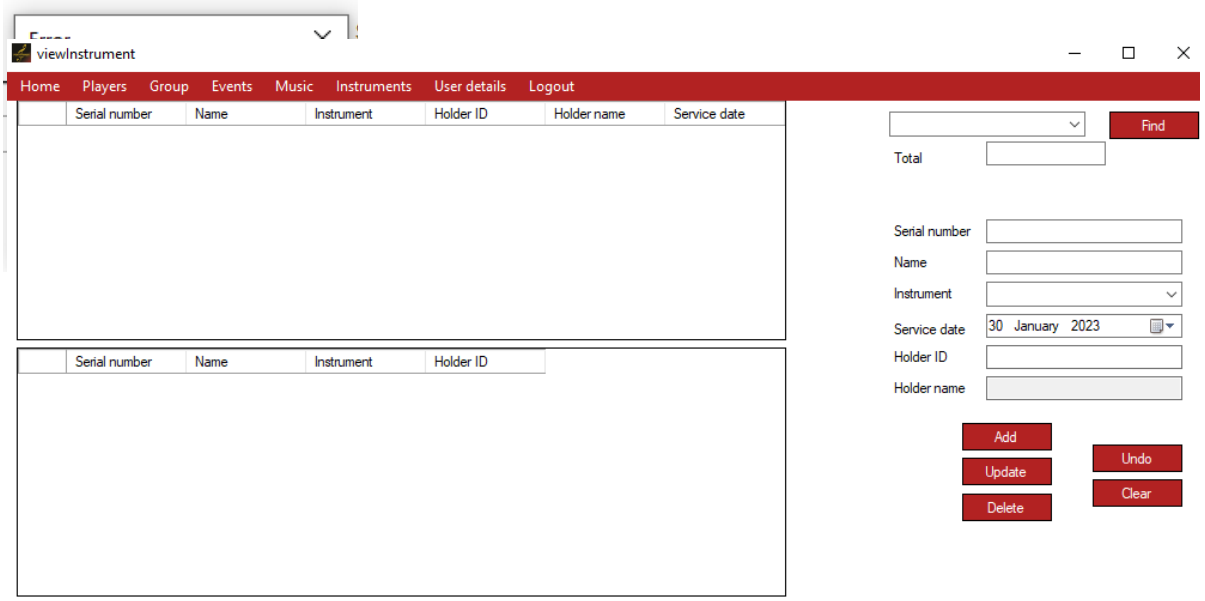
```
For Each ctl As Control In Me.Controls  'set the backColor to be what is specified
        If TypeOf ctl Is MdiClient Then
                ctl.BackColor = Me.BackColor
        End If
Next ctl
```

This checks each control in the form and if it is a MDI, it sets the back colour as the backcolor in its properties.

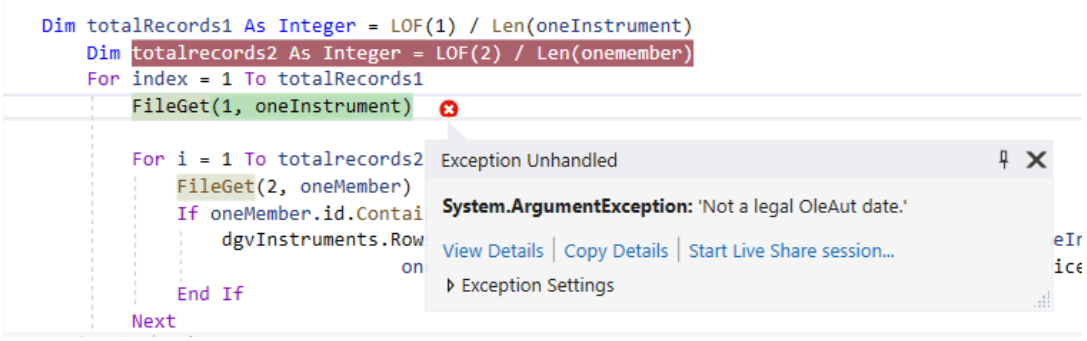


Instruments not displaying

When the user clicks on the button on the homescreen or the menubar to view the band's instruments, this message is displayed. When retry is clicked, the message appears again, showing there is something wrong with the code. When cancel is clicked, the form is shown but no instruments are displayed, despite some being stored in the database.

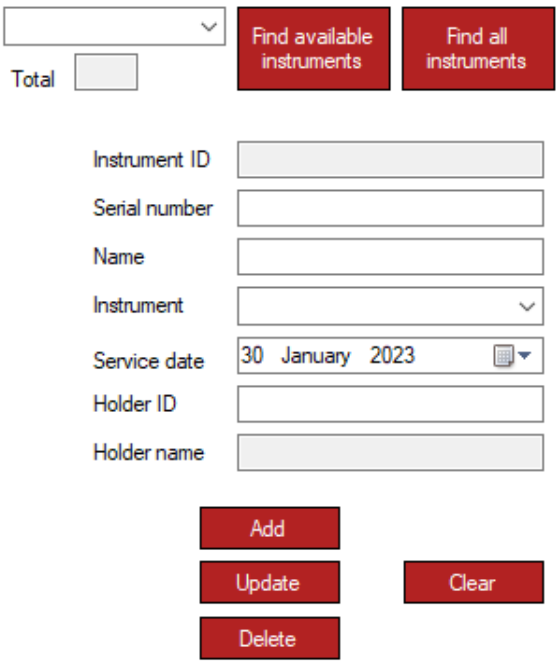


The message is shown when an exception error occurs so in order to quickly find what the error is and where it happens, I commented out the "Try, Catch" in the code. When the program is run again, the following error occurs.



After researching the error, it appeared it can occur for many reasons when FileGet is used, and it was unclear exactly what part of the code causes it. I carefully read back though the code and analysed the logic and realised that although serial numbers on instruments are unique, they do not follow the required consecutive format the FileGet method needs to read files. Therefore, a new primary key needed to be stored for each instrument. An updated data structure table is shown below.

Field Name	Data Type	Description	Length	Example	Validation
instrumentID *PK	string	The unique identification number of the instrument	5	00382	**Length check** - must be 5 characters long.
serialNumber	string	The serial number of the instrument	10	AP84729JP8	**Presence check** - data must be entered.
name	String	The name of the instrument	30	Besson Sovereign	**Presence check** - data must be entered.
instrument	String	The type of instrument	14	Cornet	**Lookup table -** shows a list of all possible instruments **Presence check** - data must be entered.
holderID *FK	String	The unique identification number of the player that has the instrument	5	00033	**Length check** - must be 5 characters long, but only if data is present
serviceDate	Date	When the instrument was last serviced	22	23/07/2021	**Presence check** - data must be entered. **Type check** - data entered must be a date



The use of the serial number as a primary key also later affects the deletion, updating and displaying of instruments, as these processes all rely on a consecutive ID. Therefore the code for these processes had been checked to ensure an ID is used, not the serial number. This ID follows the same formatting as all other IDs, and the automatic ID generation code has been added to the code when btnAdd is clicked. The designs have also been updated to show the additional read-only textbox that shows the instrument ID.

Inputting holder name and ID for an instrument

When an instrument is stored, the system allows the name of the person who has the instrument to be recorded. This data could be inputted into a textbox. However, this could cause issues as the user was required to find or remember the ID and name of the player that has the instrument, which could result in errors. To reduce the chase of errors, I added a combobox to the form, which loads all players stored into the system into it. When a player is selected, the players ID and name are in[puttend into the separate text boxes so that these details can be stored correctly in the database.

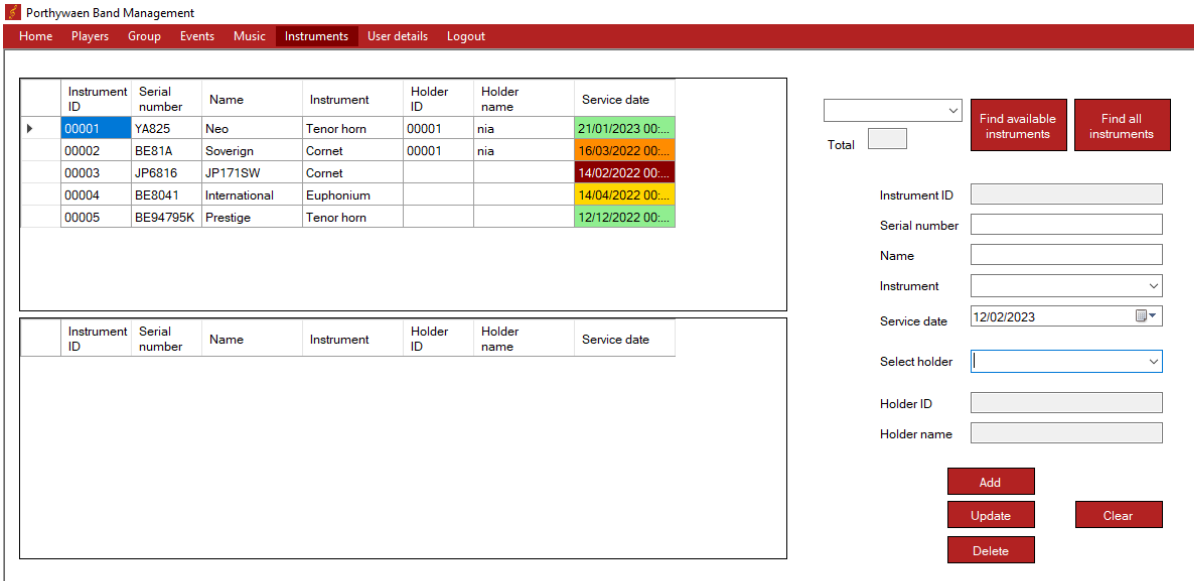
Pseudocode executed when form loads:

```
'add data to cmbHolder
        index is Integer
        oneMember is memberInfo 'pointer to structure

        FileOpen(1, "players.dat", OpenMode.Random,,, Len(oneMember))      'open file

        totalRecords is Integer = LOF(1) / Len(oneMember)      'add all details in file to dataGridView
        For index = 1 To totalRecords
                FileGet(1, oneMember)
                cmbHolder.Items.Add(oneMember.id.Trim() & ", " & oneMember.name.Trim())
        Next
        FileClose(1)
```

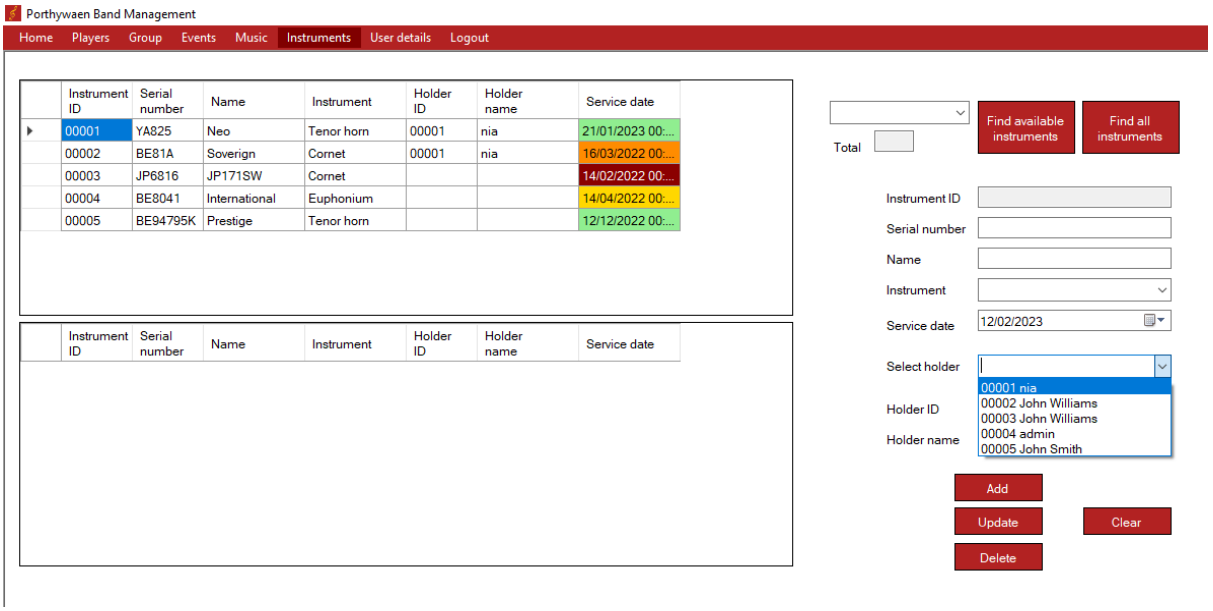
Programmed Solution to a Problem - Developmental Testing
Nia Hawkins: 7183 - The Maelor School: 68146

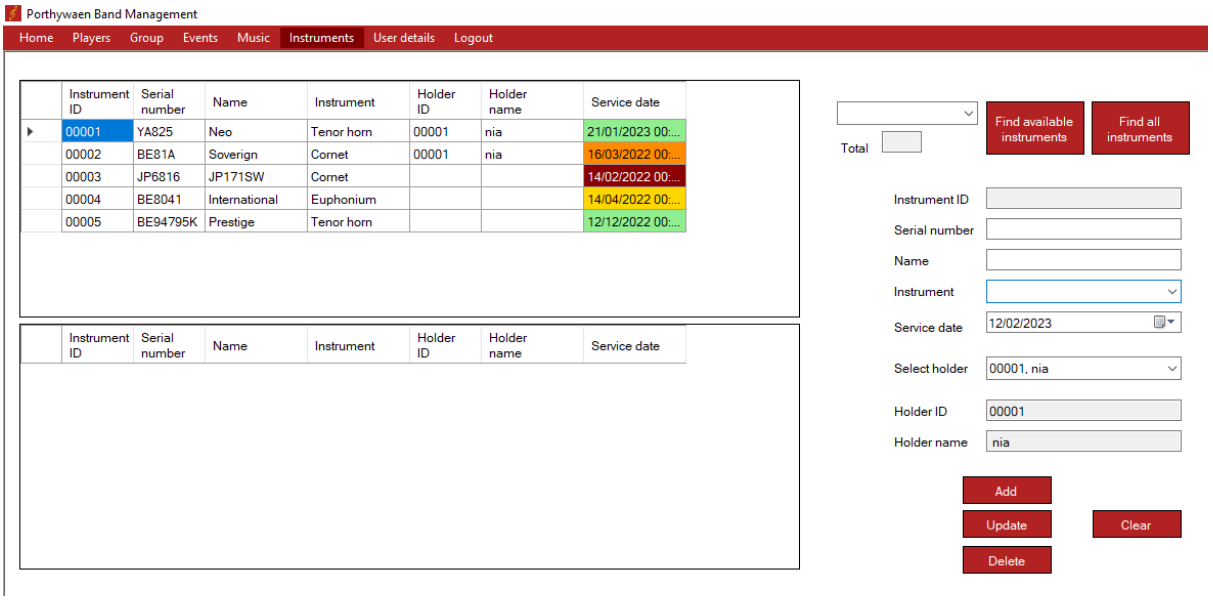
When ComboBox text changed:
'display the holderID and holderName selected in the combobox in the textboxes
    holderSelected is string = cmbHolder.Text.Split(",")
    txtHolderID.Text = holderSelected(0)
    txtHolderName.Text = holderSelected(1)

This is the new instruments form.



The user is able to select a player in the combobox and their ID and name are shown in the separate textboxes.





Attendance and event availability not updated correctly

Both of these features use a very similar algorithm that was written for one and copied into the other and changed for the data inputted and the file where it is stored. This resulted in the same issue occurring in both aspects of the system.

When btnSaveResponse was clicked in the events screen, the following code was executed.

```
Private Sub btnSaveResponse_Click(sender As Object, e As EventArgs) Handles
btnSaveResponse.Click
        Dim index As Integer
        Dim oneEvent As calendarEvent 'pointer to structure
        Dim oneCustomer As customer
        FileOpen(1, "eventsCalendar.dat", OpenMode.Random,,, Len(oneEvent))

        Dim responses As String
        Dim totalRecords As Integer = LOF(1) / Len(oneEvent)
        Dim checked As Boolean

        For index = 0 To dgvDay.Rows.Count - 1
                For j = 0 To totalRecords
                        FileGet(1, oneEvent, index)
                        If oneEvent.eventID = dgvDay.Rows(index).Cells(0).Value Then
                                responses = oneEvent.playing
                                If TypeOf dgvDay.Rows(index).Cells(3) Is DataGridViewCheckBoxCell
Then
                                        'store the mark
                                        checked = dgvDay.Rows(index).Cells(5).Value
                                        oneEvent.playing = oneEvent.playing & login.userID & ", "
                                End If
                                FilePut(1, oneEvent, totalRecords + 1)
                        End If
                Next
        Next
        FileClose(1)
        MessageBox.Show("Changes saved")
End Sub
```

When a user changed their availability or attendance at a rehearsal and clicked the relevant button to record the change, the data would be added again to the file. This happened because the code did not allow for changes. The way the data is stored in the file also made it challenging to determine if the data inputted was to be added to the file or to update existing data.

After further consideration, I identified four different possible scenarios that could occur when clicking save on the events or group screens.

1. The same data has already been recorded for the player or user, so the user must be informed that this has occurred and no changes are made to the file.
2. Different data has been recorded for the player or user. If this occurs, the program must check if the user wants to change the data stored or not.
3. No data has been recorded for the rehearsal date or event booking, so the data can be simply added to the file.
4. No data for the player or user has been stored, but there is data recorded for the rehearsal date or event booking. The data must be read from the file, and the new data must be appended to the existing data.

I also found a different way to store the data. Instead of writing all the data in the structure to the file each time data needs to be written, the new mark or response is appended to any other marks or responses. This makes it easier to determine which of the four above scenarios is true.

The code below allows a user to add and update attendance.

```
Private Sub BtnSave_Click(sender As Object, e As EventArgs) Handles btnSave.Click
        Dim oneMark As playerAttendance 'pointer to structure
        Dim recorded As Boolean
        Dim readMarks As String
        Dim mark As Boolean
        Dim marks As String()
        Dim recordNumber As Integer
        Dim found As Boolean

        If cmbGroup.Text = "" Then
                MsgBox("A group must be selected to save attendance", vbExclamation, "Group")
                Exit Sub
        End If

        Dim totalRecords As Integer

        For rowNumber = 0 To dgvMembers.Rows.Count - 1  'for each row in dgv
                FileClose(1)
                FileOpen(1, "attendance.dat", OpenMode.Random,,, Len(oneMark))
                totalRecords = LOF(1) / Len(oneMark)
                'find record and get all data
                For records = 1 To totalRecords
                        FileGet(1, oneMark)
                        If oneMark.markDate = (dtpMarkDate.Text) Then
                                If oneMark.group.Trim() = cmbGroup.Text Then
                                        readMarks = oneMark.attendance.Trim()
```

```
                                    marks = readMarks.Split(";")
                                    recordNumber = records
                                    found = True
                                    Exit For
                                End If
                            End If
                        Next

                        'saving mark in variable
                        If dgvMembers.Rows(rowNumber).Cells(4).Value = Nothing Then
                            mark = False
                        Else
                            mark = True
                        End If

                        'saving mark to array
                        If marks IsNot Nothing Then
                            For index = 0 To marks.Length - 1
                                recorded = False
                                If marks(index) <> "" Then
                                    If found = True Then
                                        If
marks(index).Contains(dgvMembers.Rows(rowNumber).Cells(0).Value & "," & mark) = True Then
                                            'check if same mark already recorded
                                            MsgBox("Attendance has already been
recorded for " & dtpMarkDate.Text & " for " &
dgvMembers.Rows(rowNumber).Cells(1).Value.Trim(","), vbExclamation, "Group")
                                            recorded = True
                                            Exit For

                                        ElseIf
marks(index).Contains(dgvMembers.Rows(rowNumber).Cells(0).Value & ",") = True Then
                                            'check if mark recorded, but different
                                            If MsgBox("Attendance for " &
oneMark.markDate & " for " & dgvMembers.Rows(rowNumber).Cells(1).Value.Trim(",") & " is
different to inputted reponse. Would you like to update it?", vbYesNo + vbInformation, "Group") =
vbYes Then
                                                'update mark
                                                marks(index) =
(dgvMembers.Rows(rowNumber).Cells(0).Value & "," & mark)

                                                Dim newMark As String = ""
                                                For i = 0 To marks.Length - 1
```

```
                                                    newMark = newMark &
marks(i) & ";"
                                                Next

                                                oneMark.attendance = newMark
                                                FilePut(1, oneMark, recordNumber)
                                                MsgBox("Attendance updated",
vbInformation, "Group")
                                                recorded = True
                                                Exit For
                                            Else 'don't want to update attendance
                                                recorded = True
                                                Exit For
                                            End If
                                        End If
                                    End If
                                End If
                            Next
                        End If

                        If recorded = False Then
                            If found = False Then    'no attendance has been recorded for this date yet so
add to file
                                oneMark.markDate = dtpMarkDate.Text
                                oneMark.group = cmbGroup.Text
                                oneMark.attendance =
(dgvMembers.Rows(rowNumber).Cells(0).Value & "," & mark & ";")
                                FilePut(1, oneMark, totalRecords + 1)

                            ElseIf found = True Then 'if it is a new attendance mark for a recorded date
                                Dim marksString As String = ""
                                'put all array elements into a string so they can be stored
                                For marksElement = 0 To marks.Length - 1
                                    If marks(marksElement) <> "" Then
                                        If marksString = Nothing Then
                                            marksString = marks(marksElement) & ";"
                                        Else
                                            marksString = marksString &
marks(marksElement) & ";"
                                        End If
                                    End If
                                Next
                                'add the new attendance to the string
```

```
                                    marksString = marksString &
dgvMembers.Rows(rowNumber).Cells(0).Value & "," & mark & ";"
                                    'store the attendance
                                    oneMark.attendance = marksString
                                    FilePut(1, oneMark, recordNumber)
                                    recorded = True
                                End If
                            End If
                        Next
                        FileClose(1)
                        MsgBox("Changes saved", vbInformation, "Group")
End Sub
```

Sorting dates

In the group screen, a graph of the attendance can be viewed by inputting a number of dates to show and clicking *show attendance*. Bubble sorts are used to ensure that when a number of rehearsals to show is inputted, the rehearsals shown are the most recent and outputted in chronological order.

The following algorithm was written to sort the dates in decreasing order.

```
'bubble sort on dates so they are in descending order
Dim swapped As Boolean = True
Dim temp As String
Dim n As Integer = recordCount - 1
Dim index As Integer

While swapped = True
        swapped = False
        For index = 0 To n - 1
        'if a date is greater than the following date, swap dates, counts and totalRead at the index
                If dates(index) > dates(index +1)
                        temp = dates(index)
                        dates(index) = dates(index + 1)
                        dates(index + 1) = temp

                        temp = counts(index)
                        counts(index) = counts(index + 1)
                        counts(index + 1) = temp

                        temp = totalRead(index)
                        totalRead(index) = totalRead(index + 1)
                        totalRead(index + 1) = temp
                        swapped = True
                End If
        Next
End While
```

However, the bubble sorting algorithm would not sort the data. After stepping through the code, I discovered that the comparison between the two dates was not working correctly as they were dates not integers. After extensive research and the testing of many different methods, I found the code that allowed the comparison of the dates.

```
If DateDiff(DateInterval.DayOfYear, CDate(dates(index)), CDate(dates(index + 1))) > 0 Then
```

This line of code subtracts the dates at dates(index) and dates(index +1). If the first date is larger than the next, the result will be greater than zero and they will be swapped.