Programmed Solution to a Problem - Testing

Porth-y-waen Silver Band Management System

Nia Hawkins: 7183
The Maelor School: 68146

Contents

Testing

I will devise a test plan to test all system functions. I will carry out alpha and black box testing to test the overall system functionality. I will complete white box testing to test that the code will work in every circumstance and the code pathways work as expected. I will test the system with normal, extreme and erroneous data to ensure the code works as expected.

Test plan

Test ID	Description	Type of test	Test data	Expected results
Splash screen				
1	Check that the splash screen is displayed when the program is opened.	Normal	Run the program.	The splash screen is shown.
General tests				
2	Check all navigation.	Normal	Click on menu bars and home screen buttons.	The relevant form is shown.
3	Check input fields can be cleared	Normal	In any form with data contained in input fields, click "Clear".	All input fields are cleared or reset.
4	Check that data is shown in DataGridViews when forms are shown.	Normal	Open any form with a DataGridView or click show on any form.	The data in the database for the form is shown.
5	Check details are displayed when the DataGridView is clicked.	Normal	Click a record in any DataGridView.	The records details are shown in the input fields.
Login and reset password				
6	Check if the user can login to the system.	Normal	Name: John Smith Password: music123! Click "Login".	The home screen is shown with the relevant level of access.
7	Check the name field is required.	Erroneous	Name: [blank] Password: music123! Click "Login".	An error message is shown to notify the user.

Test ID	Description	Type of test	Test data	Expected results
8	Check the password is reset.	Normal	Name: John Smith Click "Reset password".	A message shown informing the user their password has been reset to their ID. The user can login with their ID as their password. The change password form shown.
9	Check a password can be changed.	Normal	Password: music123! Confirm: music123! Click “Save and login”.	A message is shown to notify the user their password has been changed. The user is logged into the system and the home screen is shown with the relevant level of access.
10	Check the new password is what the user intended.	Erroneous	Password: music123! Confirm: [blank] Click “Save and login”.	A message is shown to tell the user these inputs do not match.
11	Check the length of a password.	Extreme	Password: cornet! Confirm: cornet! Click “Save and login”.	A message is shown telling the user that the password is too short.
Events				
12	Check events on a certain date are displayed.	Normal	Click on a date in the calendar.	All event booking on the selected date are shown. A message to inform the user that there are no bookings on that date.

Test ID	Description	Type of test	Test data	Expected results
13	Check an event can be added to the system.	Normal	Event ID: 00020 Address: Ellesmere Town Hall, Ellesmere Postcode: SY12 0AL Date: 04/02/2023 Start time: 1pm Arrival time: 12:30 Groups: PSB Music: tbc Customer ID: 00019 Name: Julia Smith Phone: 07466285991 Email: julia.smith@gmail.com Click “Add event”.	A message is displayed to confirm the event is added. The DataGridView is refreshed to show the event.
14	Check an event can be updated.	Normal	Event ID: 00020 Address: Ellesmere Town Hall, Ellesmere Postcode: SY12 0AL Date: 04/02/2023 Start time: 2pm Arrival time: 1:30 Groups: PSB, PYTB Music: tbc Customer ID: 00019 Name: Julia Smith Phone: 07466285991 Email: julia.smith@gmail.com Click “Add event” or "Update event”.	A message is shown telling the user the event has been updated. The DataGridView is refreshed to show changes.

Test ID	Description	Type of test	Test data	Expected results
15	Check address field is needed.	Erroneous	Event ID: 00020 Address: [blank] Postcode: SY12 0AL Date: 04/02/2023 Start time: 1pm Arrival time: 12:30 Groups: PSB Music: tbc Customer ID: 00019 Name: Julia Smith Phone: 07466285991 Email: julia.smith@gmail.com Click “Add event “or "Update" event”.	A message shown informing the user that they need to enter an address.
16	Check the postcode must be in postcode format and needed.	Erroneous	Event ID: 00020 Address: Ellesmere Town Hall, Ellesmere Postcode: ABCDE Date: 04/02/2023 Start time: 1pm Arrival time: 12:30 Groups: PSB Music: tbc Customer ID: 00019 Name: Julia Smith Phone: 07466285991 Email: julia.smith@gmail.com Click “Add event” or "Update" event”.	A message is shown informing the user that the postcode is in the incorrect format.
17	Check the phone number is the correct length.	Extreme	Event ID: 00020 Address: Ellesmere Town Hall, Ellesmere Postcode: SY12 0AL Date: 04/02/2023 Start time: 2pm Arrival time: 1:30 Groups: PSB, PYTB Music: tbc Customer ID: 00019	A message is shown informing the user that the phone number is an incorrect length.

Test ID	Description	Type of test	Test data	Expected results
			Name: Julia Smith Phone: 0746628599 Email: julia.smith@gmail.com Click “Add event” or "Update" event”.	
18	Check the date is within an acceptable range.	Erroneous	Event ID: 00020 Address: Ellesmere Town Hall, Ellesmere Postcode: SY12 0AL Date: 10/05/1893 Start time: 2pm Arrival time: 1:30 Groups: PSB, PYTB Music: tbc Customer ID: 00019 Name: Julia Smith Phone: 0746628599 Email: julia.smith@gmail.com Click “Add event” or "Update" event”.	A message is shown informing the user that the date is invalid.
19	Check that the date and time selected for the booking is available.	Normal	Event ID: 00020 Address: Ellesmere Town Hall, Ellesmere Postcode: SY12 0AL Date: 04/02/2023 Start time: 2pm Arrival time: 1:30 Groups: PSB, PYTB Music: tbc Customer ID: 00019 Name: Julia Smith Phone: 0746628599 Email: julia.smith@gmail.com Click “Add event”.	If date is available, add the event to the database as normal. If another booking has been added for the same date and time, a message is shown to the user to inform them another booking has been made at the same date and time and ask if they would like to add the booking to the database or not.
20	Check an event can be deleted.	Normal	Click on an event in the DataGridView and click “Delete event”.	A message is shown informing the user that

Test ID	Description	Type of test	Test data	Expected results
				the event has been deleted. The DataGridView is updated to show the changes.
21	Check the events can be printed.	Normal	Click “Print events”.	The print preview shown, allowing the user to print the events.
22	Check the user is able to save their availability for events.	Normal	Check the checkboxes in the DataGridView and click “Save availability”.	A message showed that the user's availability has been saved.
23	Check the user only has one response for each event.	Normal	Check the checkboxes in the DataGridView and click “Save availability”.	A message is shown if the same response has already been stored to notify the user that this response has not been saved.
24	Check if the user can update their availability for an event.	Normal	Check the checkboxes in the DataGridView and click “Save availability”.	If the response is different, a message is shown asking the user if they would like to update their answer.
25	Check users can see the available players for an event.	Normal	Click an event in the DataGridView and then click “View available players”.	Form shown that shows all players that are available for the event.
26	Check that an event must be selected to show player availability.	Erroneous	Click “View available players”.	A message is shown informing the user that they need to select an event to show player availability.
Group				
27	Check a group is shown.	Normal	Select a group in the combo box.	All players in the group are displayed in the DataGridView.

Test ID	Description	Type of test	Test data	Expected results
28	Check attendance can be recorded on a certain date.	Normal	Select a date and check the applicable checkboxes in the DataGridView. Then click "Save attendance".	Message shown confirming the attendance has been saved.
29	Check that attendance can be updated.	Normal	Select a date and check the applicable checkboxes in the DataGridView. Then click "Save attendance".	A message is shown informing the user that a different mark has been recorded for this player, and asks them if they wish to update it or not. If they click yes, the attendance is updated and a message is shown to confirm.
30	Check that attendance can be only recorded once for each player in a group.	Normal	Select a date and check the applicable checkboxes in the DataGridView. Then click "Save attendance".	A message is shown informing the user that this mark has already been recorded for a certain player on the chosen date.
31	Check that a group must be selected to mark and view attendance	Erroneous	Click "Save attendance" or show attendance without selecting a group.	A message is shown informing the user that a group must be selected to show or save attendance.
32	Check attendance can be viewed in a graph.	Normal	Enter the number of weeks to show in the textbox and click show attendance.	Attendance for the number of weeks shown in a DataGridView and as a bar chart.
Players and User details				
33	Check if a player can be added to the database.	Normal	ID: 00001 Name: John Smith DOB: 19/02/1998 Email: johnSmith@gmail.com Phone: 07283266481 Instrument: Cornet Level: 7 Photograph permission: True Groups: PSB, PYTB Role: Player contName: Mary Smith contPhone: 07384736285	A message is displayed confirming the player has been added.

Test ID	Description	Type of test	Test data	Expected results
			Click "Add" .	
34	Check that a player's details can be updated.	Normal	ID: 00001 Name: John Smith DOB: 19/02/1998 Email: johnSmith@gmail.com Phone: 07283266481 Instrument: Cornet Level: 7 Photograph permission: True Groups: PSB, PYTB Role: Player contName: Jack Smith contPhone: 07384736285 Click "Update" .	A message confirming the details have been updated is displayed.
35	Check that the player's name is needed.	Erroneous	ID: 00001 Name: [blank] DOB: 19/02/1998 Email: johnSmith@gmail.com Phone: 07283266481 Instrument: Cornet Level: 7 Photograph permission: True Groups: PSB, PYTB Role: Player contName: Jack Smith contPhone: 0738473628543 Click "Add" or "Update" .	A message is shown informing the user that they need to enter the player's name.
36	Check that the player's email is in the correct format.	Erroneous	ID: 00001 Name: John Smith DOB: 19/02/1998 Email: johnSmithgmailcom Phone: 07283266481 Instrument: Cornet Level: 7 Photograph permission: True Groups: PSB, PYTB Role: Player	A message is shown informing the user that the email entered is in the incorrect format.

Test ID	Description	Type of test	Test data	Expected results
			contName: Jack Smith contPhone: 0738473628543 Click "Add" or "Update" .	
37	Check the player's DOB is within an acceptable range.	Erroneous	ID: 00001 Name: John Smith DOB: 19/02/1834 Email: johnSmith@gmail.com Phone: 07283266481 Instrument: Cornet Level: 7 Photograph permission: True Groups: PSB, PYTB Role: Player contName: Jack Smith contPhone: 073847362854 Click "Add" or "Update" .	A message is shown informing the user that the date is invalid.
38	Check the contPhone number is 11 characters long.	Extreme	ID: 00001 Name: John Smith DOB: 19/02/1998 Email: johnSmith@gmail.com Phone: 07283266481 Instrument: Cornet Level: 7 Photograph permission: True Groups: PSB, PYTB Role: Player contName: Jack Smith contPhone: 073847362854 Click "Add" or "Update" .	A message is shown informing the user that the phone number must be 11 characters long.

Test ID	Description	Type of test	Test data	Expected results
39	Check that at least one group must be selected.	Erroneous	ID: 00001 Name: John Smith DOB: 19/02/1998 Email: johnSmith@gmail.com Phone: 07283266481 Instrument: Cornet Level: 7 Photograph permission: True Groups: [blank] Role: Player contName: Jack Smith contPhone: 07384736285434 Click "Add" or "Update" .	A message is shown informing the user that at least one group needs to be selected.
40	Check that a player can be deleted from the database.	Normal	Click on a player in the DataGridView and click “Delete”.	A message is shown to confirm the player has been deleted. The DataGridView is refreshed to show the changes.
Music				
41	Check that piece can be added to the database.	Normal	ID: 00001 Title: Shine as the Light Composer: Peter Graham Click "Add" .	A message is shown informing the user that the music has been added. The DataGridView is refreshed to show the latest changes.
42	Check that music can be updated.	Normal	ID: 00001 Title: Shine as the Light Composer: Peter Graham arr. Sandy Smith	
43	Check that the title is needed.	Erroneous	ID: 00001 Title: [blank] Composer: Peter Graham Click "Add" or "Update" .	A message is shown informing the user that the title is needed.

Test ID	Description	Type of test	Test data	Expected results
44	Check that a piece can be deleted.	Normal	When a piece is selected, click "Delete" .	A message is shown informing the user that the music has been deleted. The DataGridView is refreshed to show the changes.
45	Check that a piece can be searched for.	Normal	TitleSearch: Shine as the light Click “Search”.	If the music is found it is shown in the DataGridView. If it is not found, a message is shown informing the user that the music is not in the database.
46	Check that title is needed when searching.	Erroneous	TitleSearch: [blank] Click “Search”.	A message is shown informing the user that a title needs to be entered to search for.
Instruments				
47	Check that an instrument can be added.	Normal	InstrumentID: 00001 SerialNumber: BE94795K Name: Prestige Instrument: Tenor horn Service date: 12/12/2022 HolderID: 00005 HolderName: John Smith Click "Add" .	A message is displayed informing the user that the instrument has been added to the database. The DataGridView is refreshed to show the changes.
48	Check that an instrument’s details can be updated	Normal	InstrumentID: 00001 SerialNumber: BE94795K Name: Prestige Instrument: Tenor horn Service date: 12/12/2022 HolderID: [blank] HolderName: [blank] Click "Update" .	A message is displayed informing the user that the instrument has been updated. The DataGridView is refreshed to show the changes.

Test ID	Description	Type of test	Test data	Expected results
49	Check that an instrument can be deleted.	Normal	When an instrument is selected, click "Delete" .	A message is displayed confirming the instrument has been deleted. The DataGridView is refreshed to show the changes.
50	Check that the serial number is required.	Erroneous	InstrumentID: 00001 SerialNumber: [blank] Name: Prestige Instrument: Tenor horn Service date: 12/12/2022 HolderID: 00005 HolderName: John Smith Click "Add" or "Update" .	A message is shown informing the user that the serial number needs to be entered.
51	Check that the service date is a valid date.	Erroneous	InstrumentID: 00001 SerialNumber: BE94795K Name: Prestige Instrument: Tenor horn Service date: 11/06/1834 HolderID: 00005 HolderName: John Smith Click "Add" or "Update" .	A message is shown informing the user that the data is invalid.
52	Check that only players in the system can be instrument holders.	Normal	Open form.	All player IDs and names are added to a combo box.
Logout				
53	Check that the user can logout.	Normal	Click “Logout”.	A message is shown confirming if the user wants to logout. If yes is clicked, the program is closed. If no is clicked, the program stays open.

Testing evidence

Splash screen

Test 1

This test checks that when the program is run the splash screen should be shown for a few seconds before the login screen is displayed.

Figure 1: The splash screen.

General tests

Test 2

This test checks if the user can navigate between different forms in the program. When a menu bar element is clicked, the corresponding screen is shown. The menu bar also indicates which form the user is on by changing the colour of the bar. All forms have also been correctly shown in the parent form, home, so they are shown in the same place in the screen when loaded as the previous form.

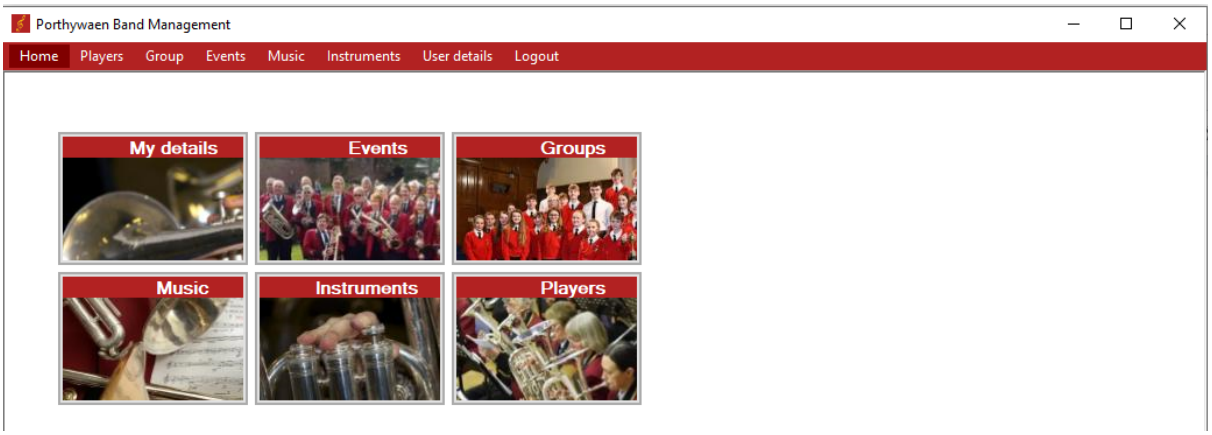

*Figure 1: The "*Home*" screen shown after "*Home*" clicked in the menu bar.*

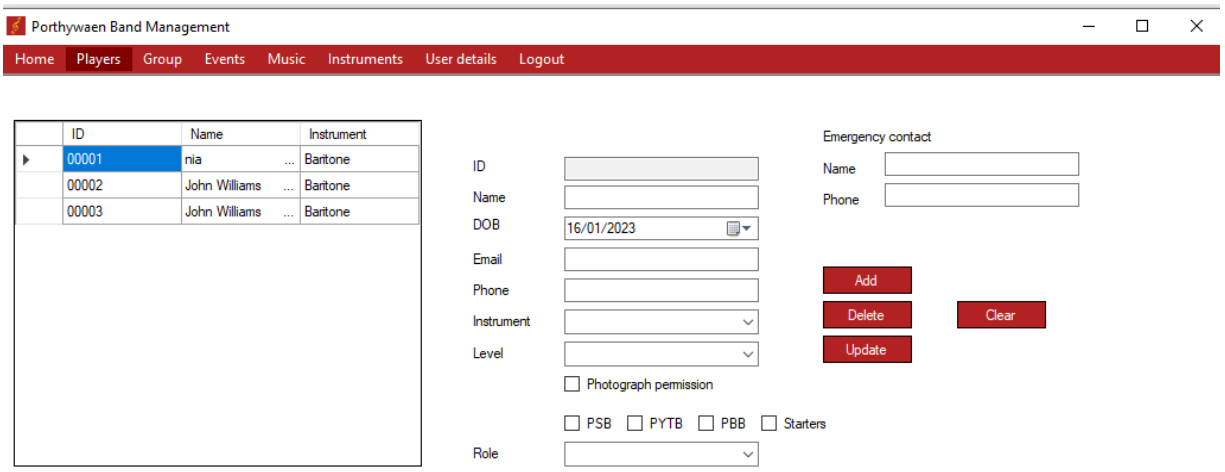

*Figure 2: The "*Players*" screen shown after "*Players*" clicked in the menu bar.*

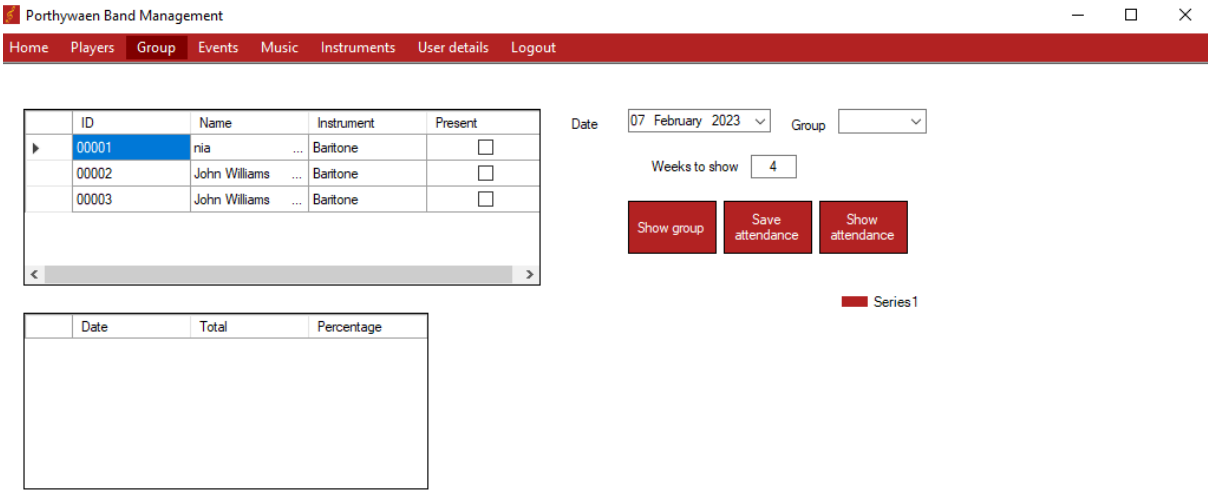

Figure 3: The Group*" screen shown after "*Group*" clicked in the menu bar.*

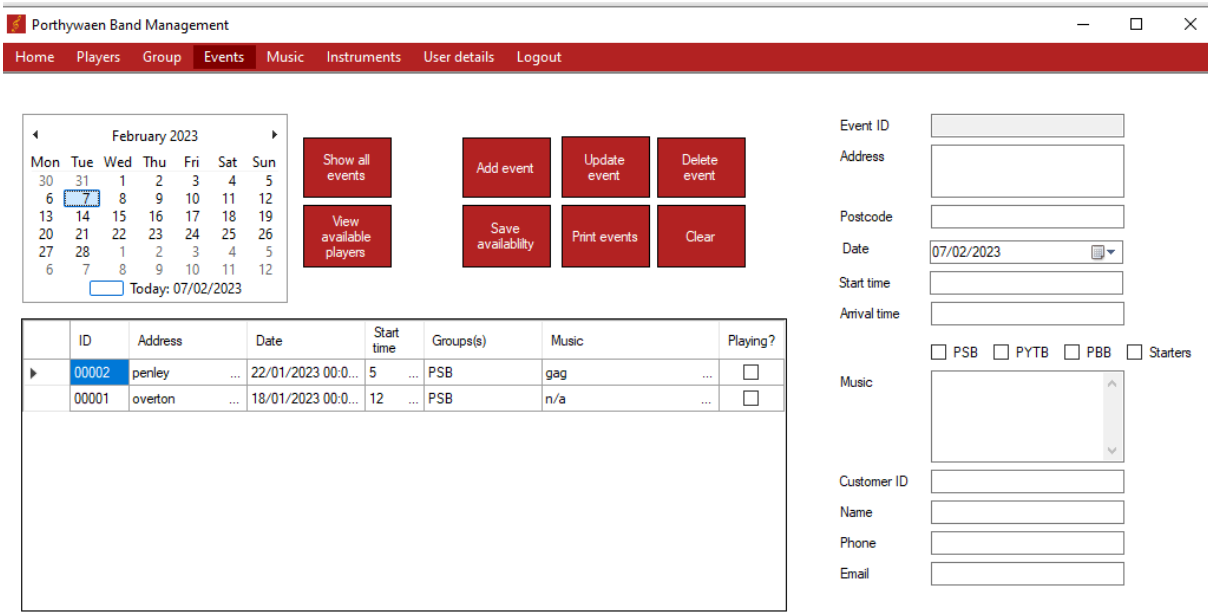

Figure 4: The "Events" screen shown after "Events" clicked in the menu bar.

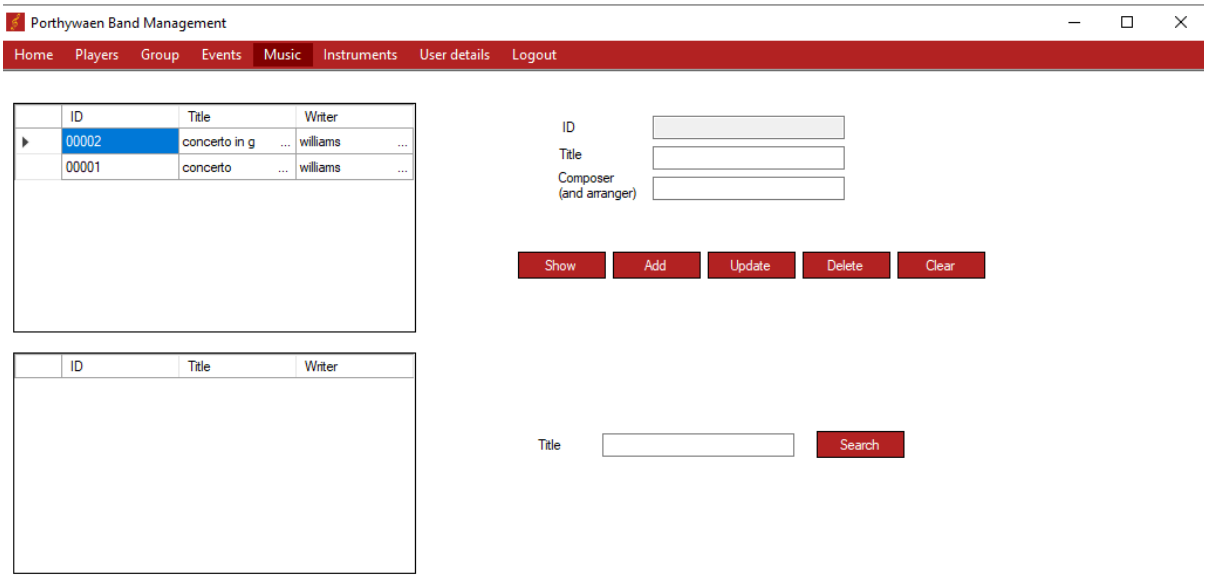

Figure 5: The "Music" screen shown after "Music" clicked in the menu bar.

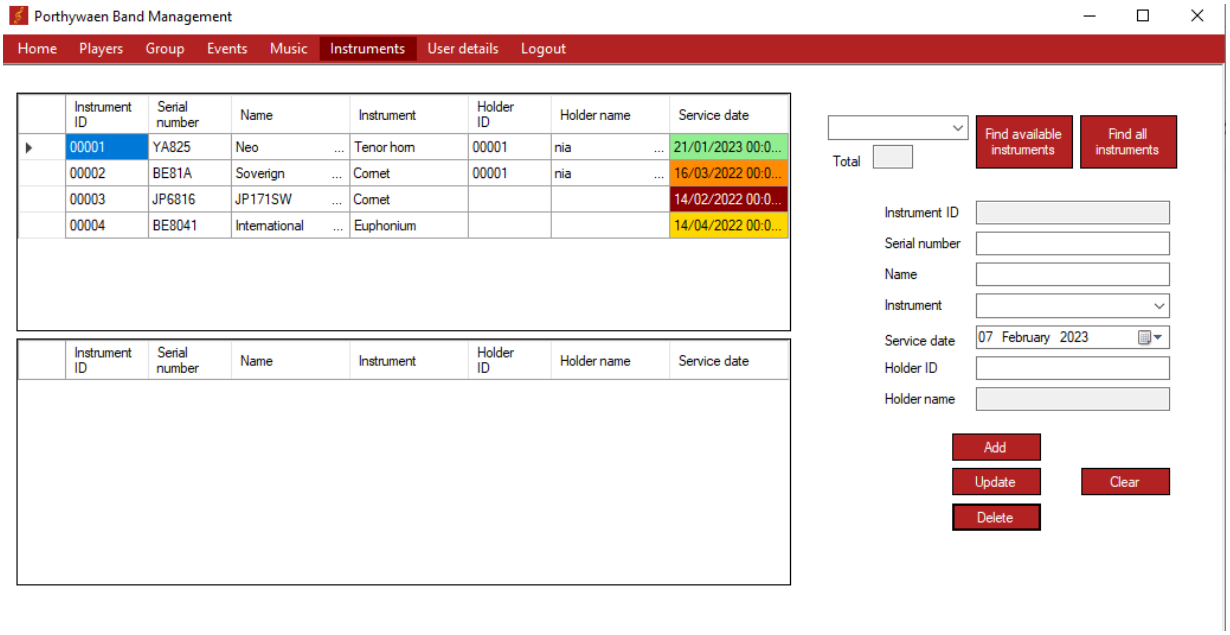

Figure 6: The "Instruments" screen shown after "Instruments" clicked in the menu bar.

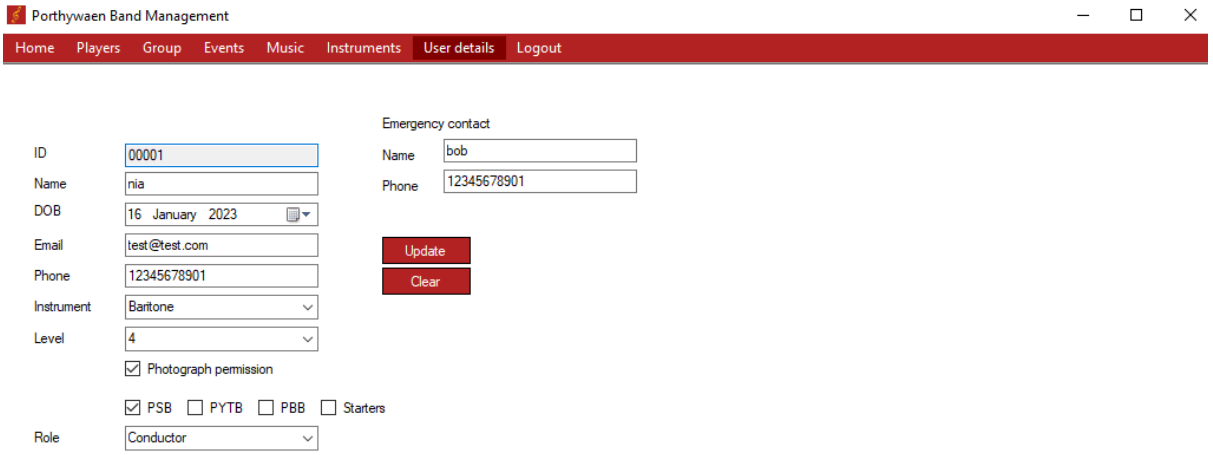

Figure 7: The "User details" screen shown after "User details" clicked in the menu bar.

Test 3

This test is to check that the input fields on all forms can be reset when "Clear'' is clicked. When “Clear" is clicked, all textboxes are cleared, the checkboxes are reset to False and the DateTimePicker is set to today’s date.

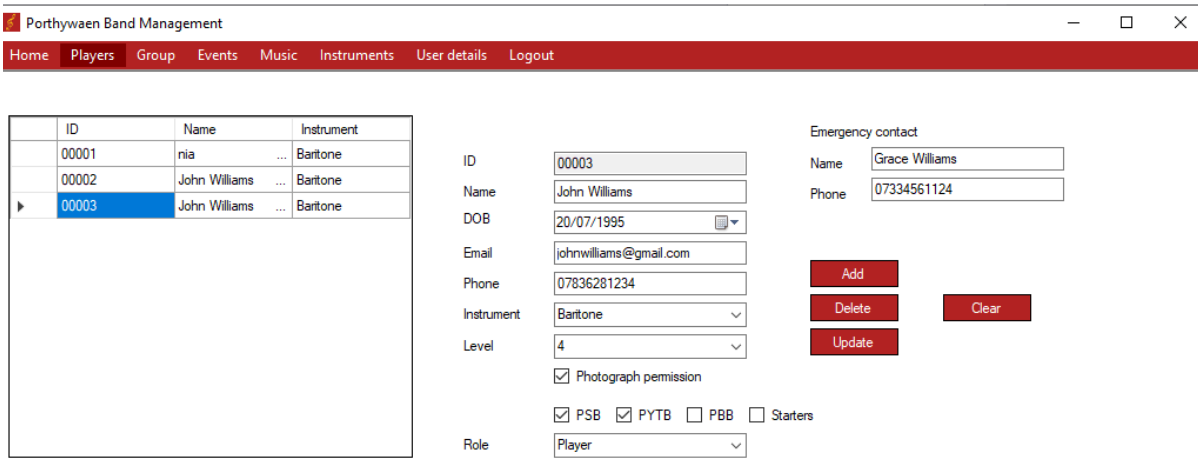

Figure 8: The “Players” screen before "Clear" clicked.

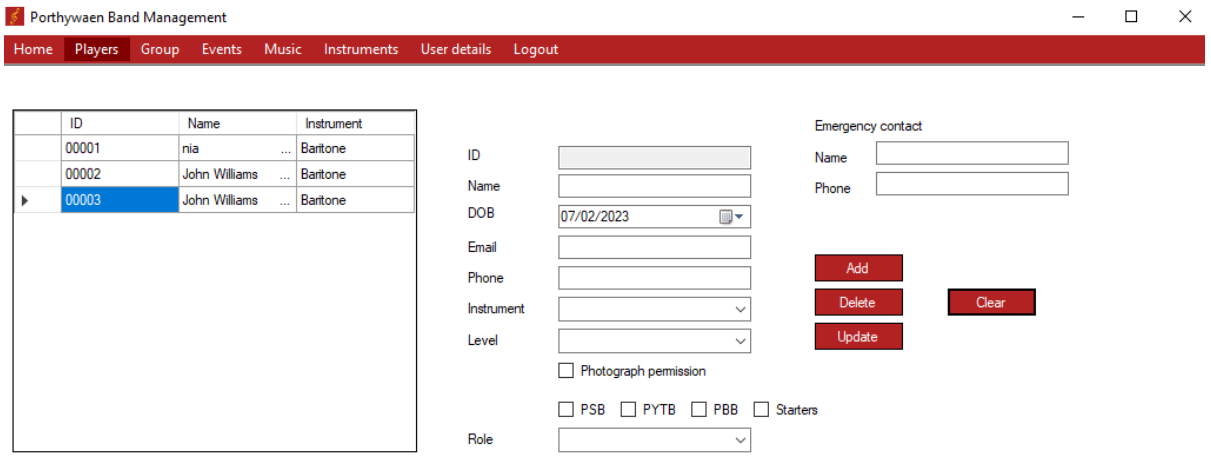

Figure 9: The “Players” screen after "Clear" clicked.

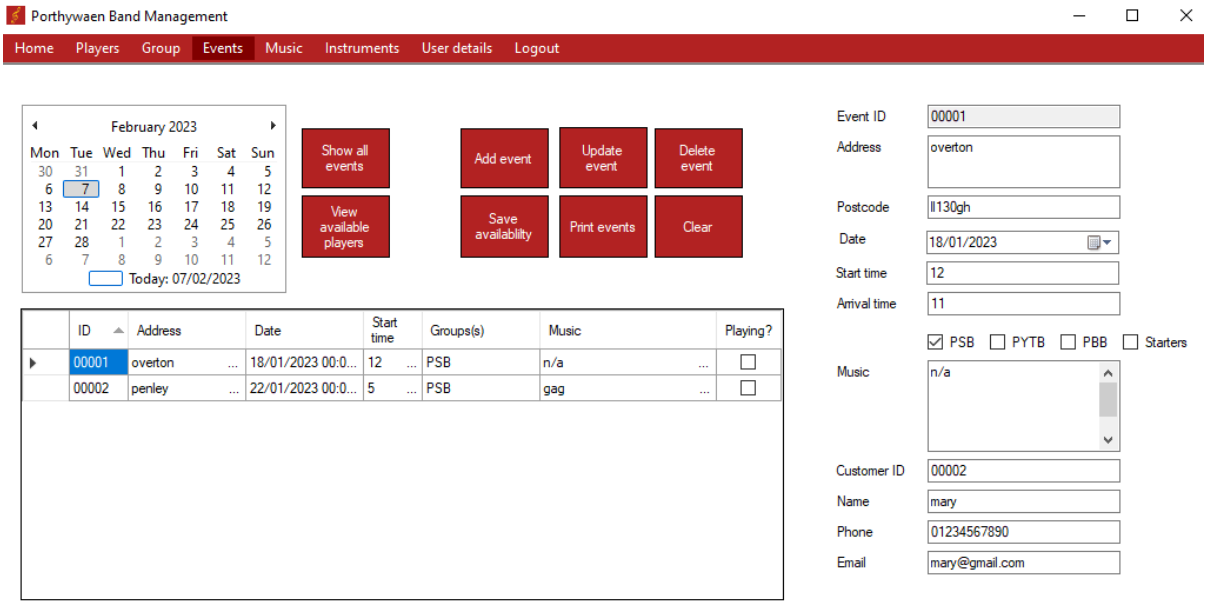

Figure 10: The "Events" screen before "Clear" clicked.

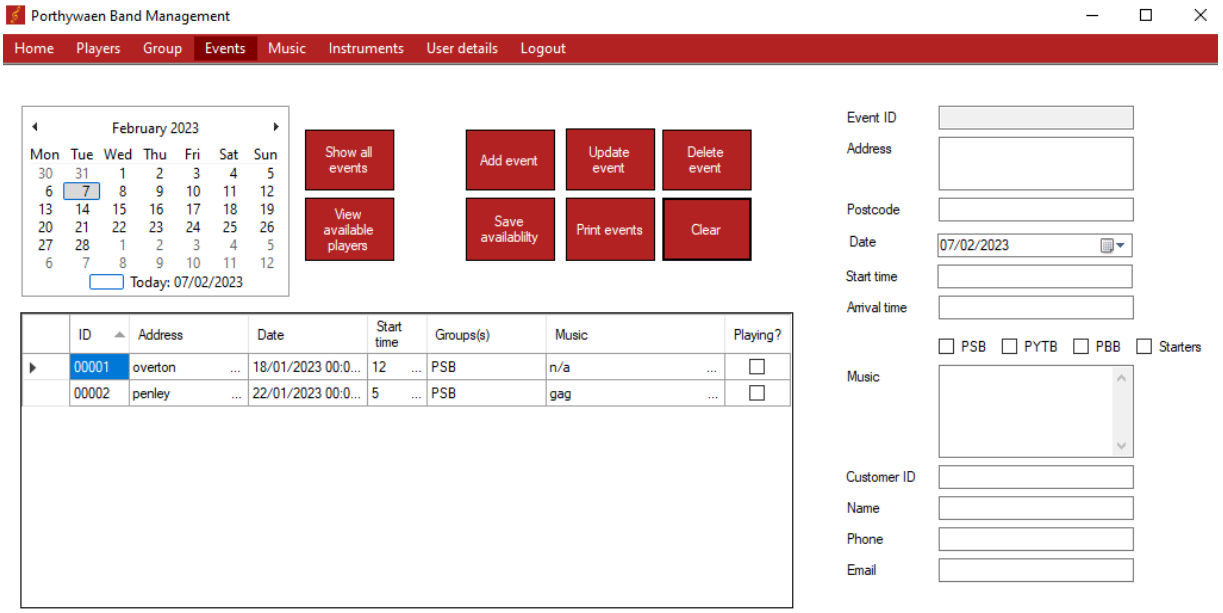

Figure 11: The "Events" screen after "Clear" clicked.

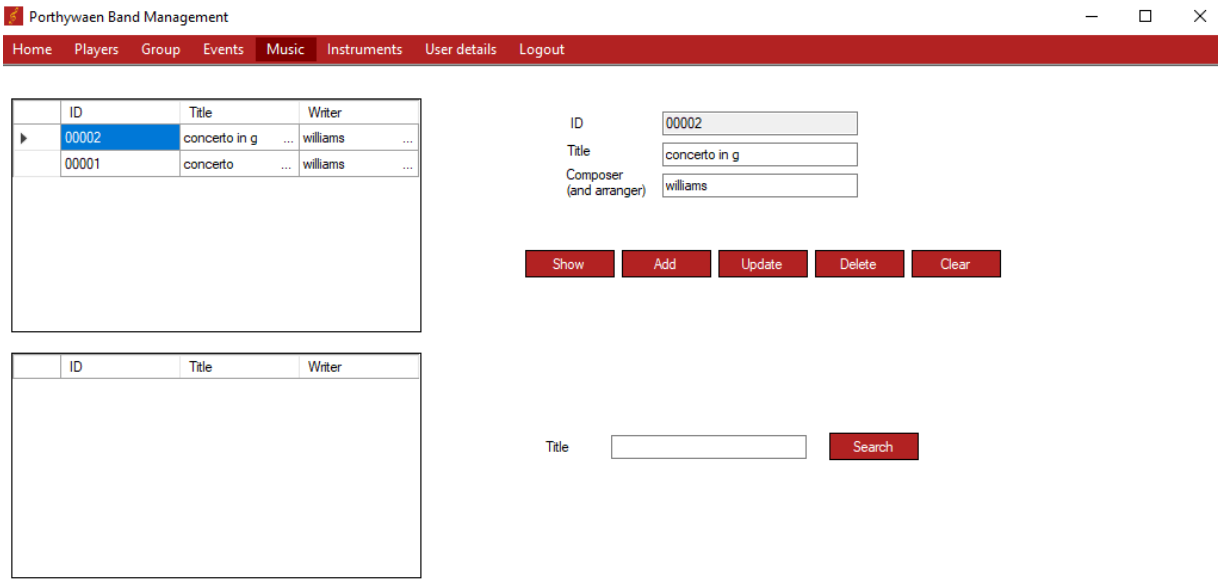

Figure 12: The "Music" screen before "Clear" clicked.

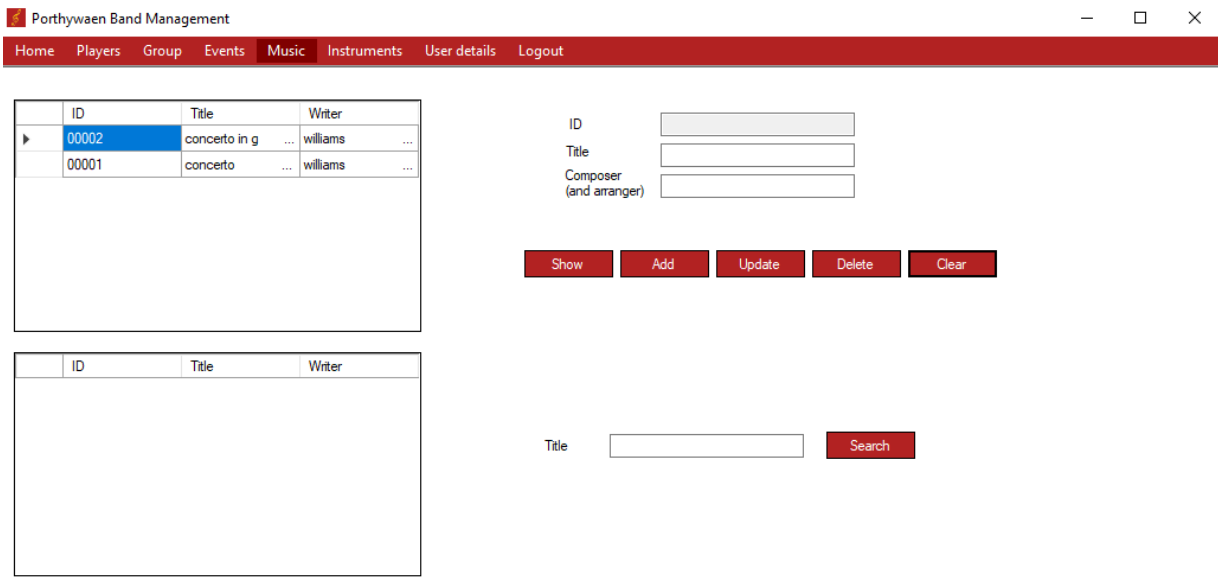

Figure 13: The "Music" screen after "Clear" clicked.

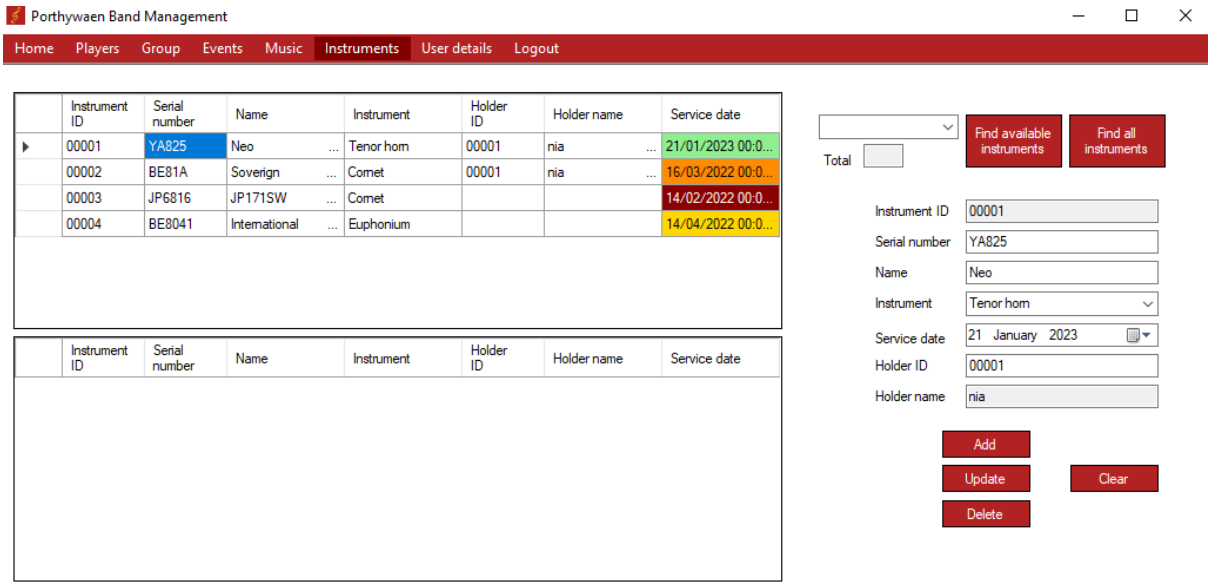

Figure 14: The "Instruments" screen before "Clear" clicked.

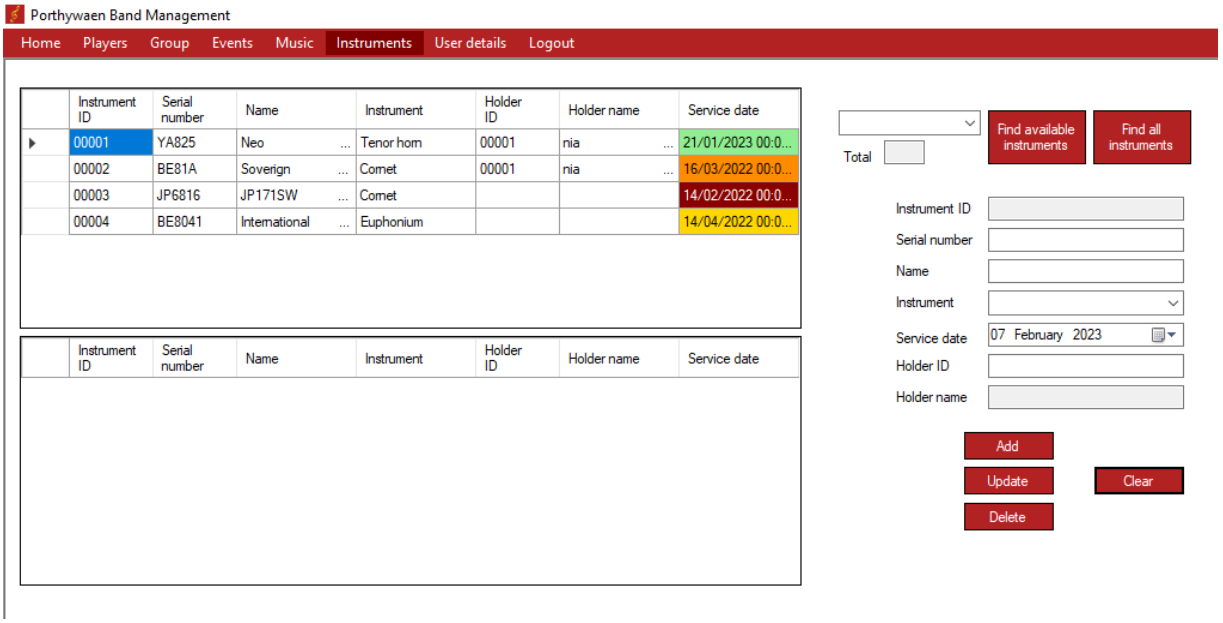

Figure 15: The "Instruments" screen after "Clear" clicked.

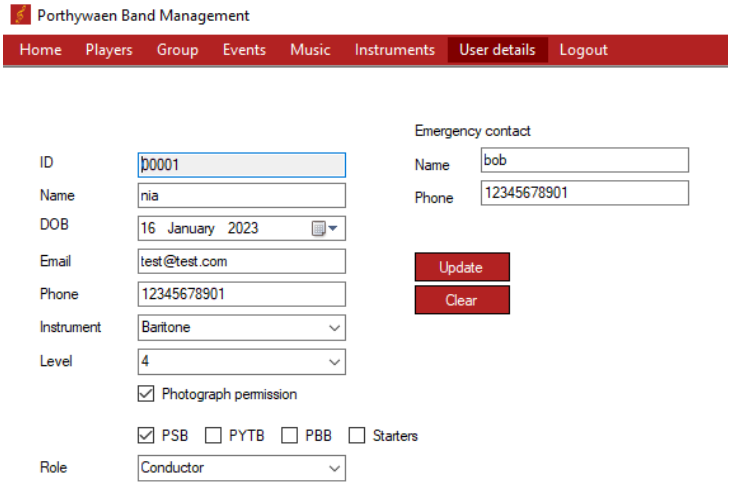

Figure 16: The "User details" screen before "Clear" clicked.

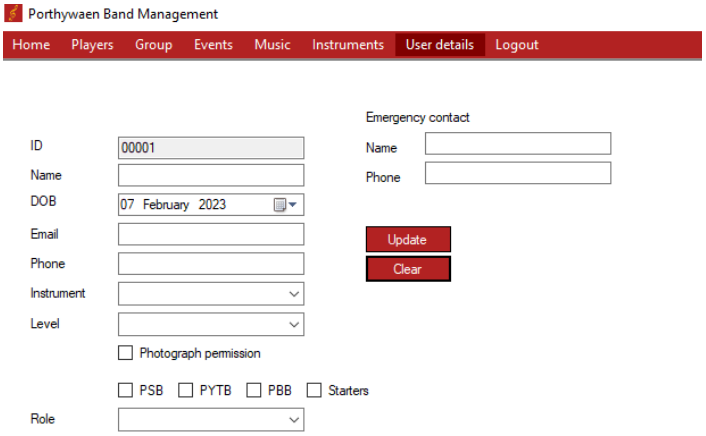

Figure 17: The "User details" screen after "Clear" clicked.

The user ID is not cleared as this form allows the user to view and update their own details. IDs should not be changed and as a result, the ID textbox is read only. This means that an error would occur when "Update" is clicked and the ID is blank and the user will not be able to resolve the issue.

Test 4

This test will check that when a form is shown or show is clicked, the relevant data is read from the database and displayed in the DataGridView on the form. To do this, the relevant file for the screen is opened, and each record is stored in a structure one at a time. The structure is added to a row in the DataGridView. This repeats for each record in the file until the end of the file is reached.

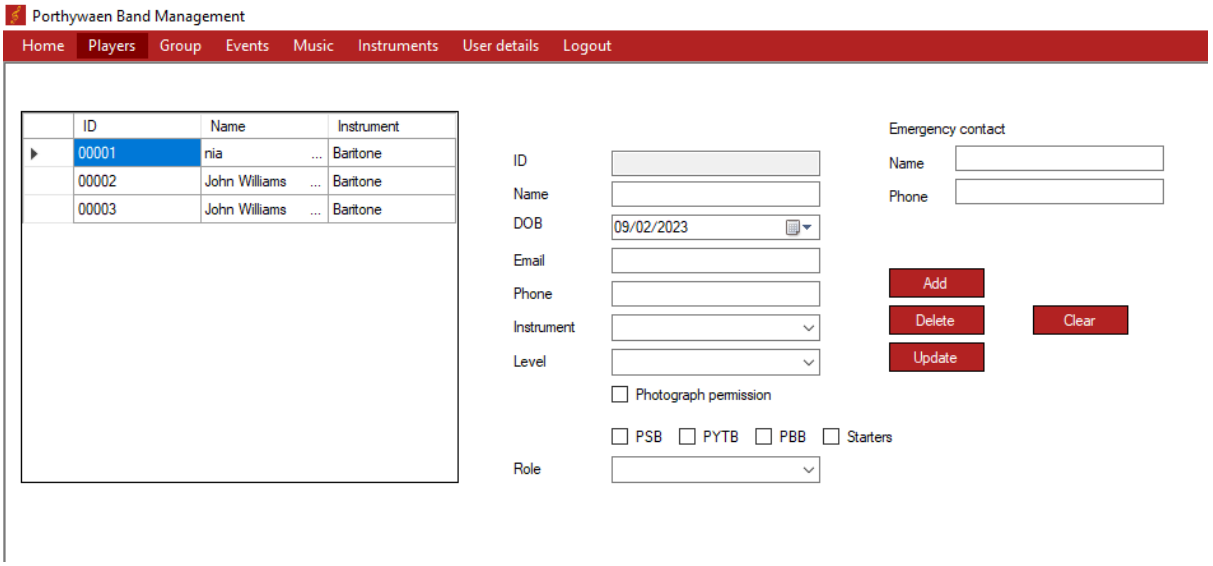

Figure 18: When the "Players" screen is shown.

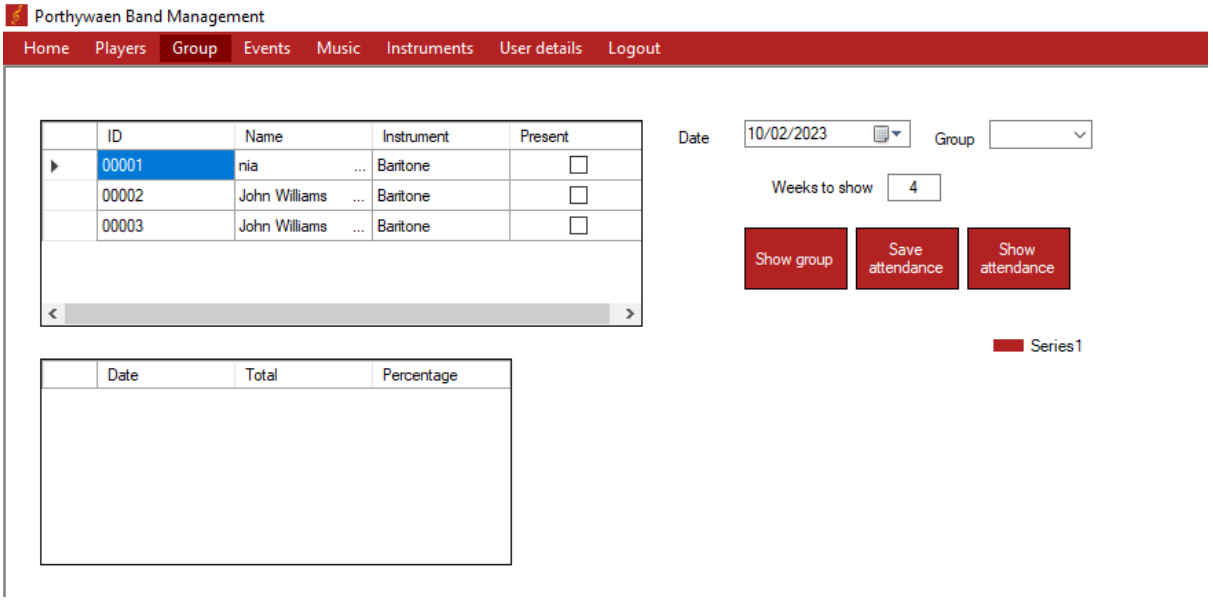

Figure 19: When the "Group" screen is shown.

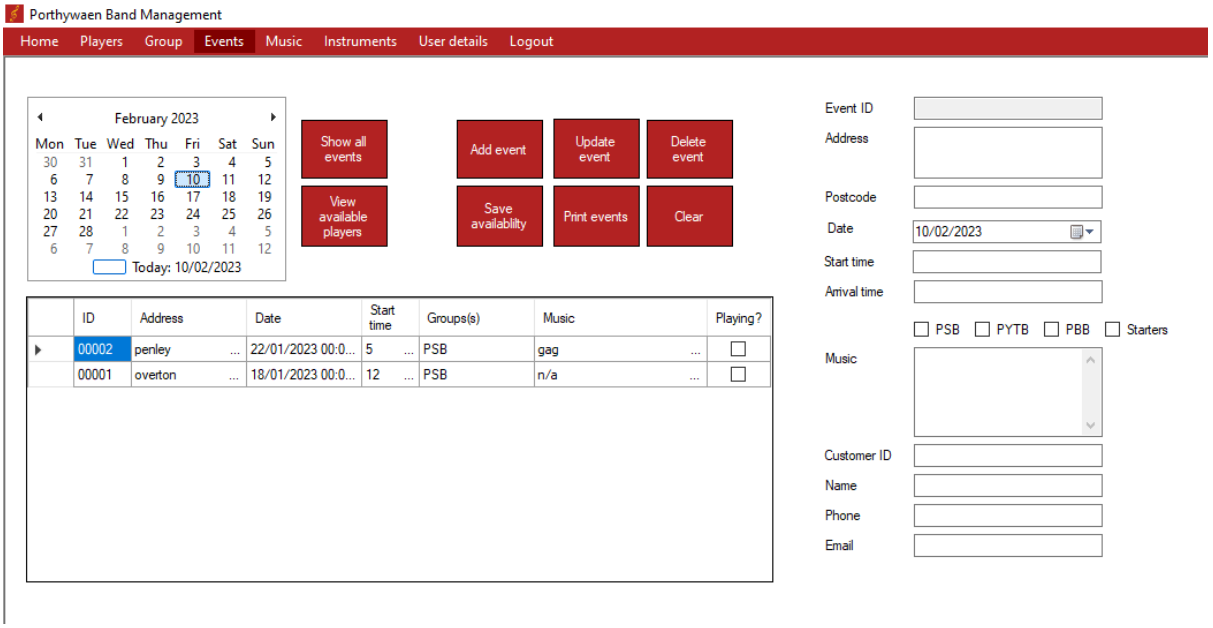

Figure 20: When the "Events" screen is shown or"Show all events" is clicked.

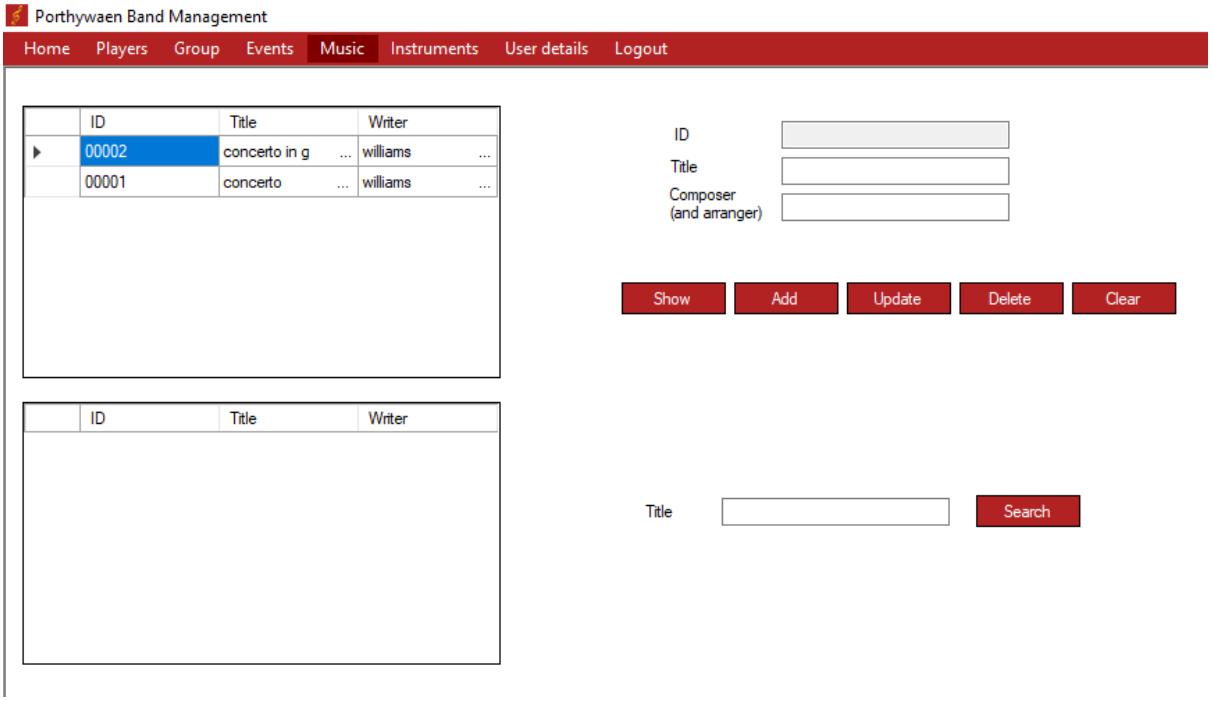

Figure 21: When the "Music" screen is shown.

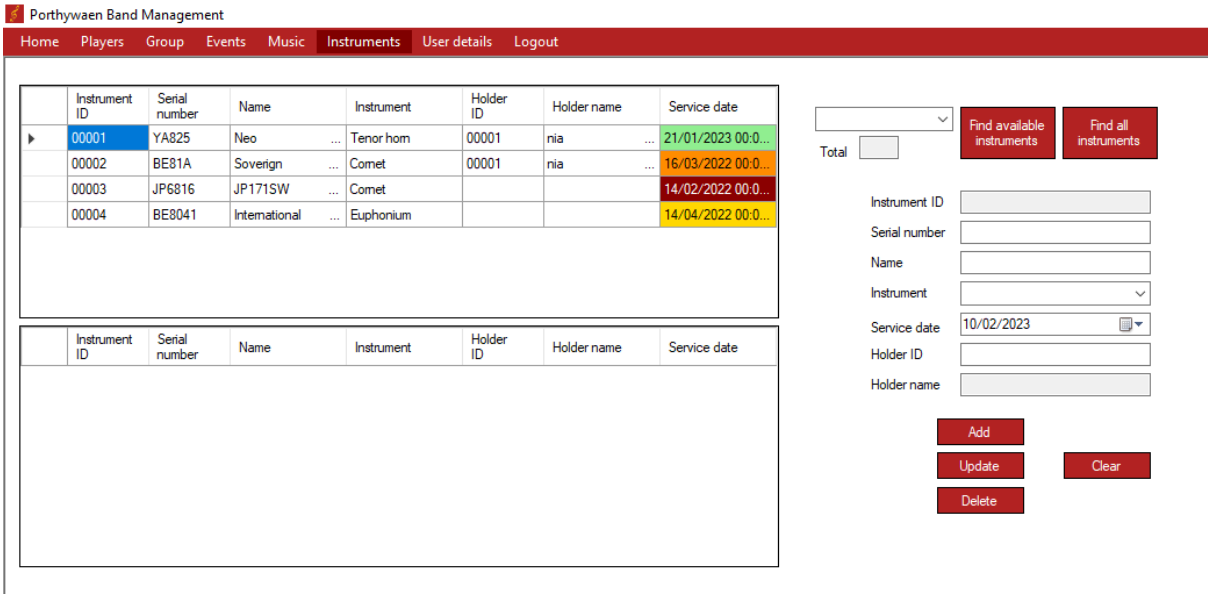

Figure 22: When the "Instruments" screen is shown.

Test 5

This test checks that when a DataGridView record is clicked, all of the record's details are shown in the relevant input fields in all forms. When a cell in the DataGridView is clicked, a subroutine is called which firstly checks that the user intends to view the records fields, and not sort or check a checkbox in the DataGridView. The relevant file is searched to find the record that was selected and the fields are stored in a structure. Each element in the structure is added to the relevant field in the form, and the string that stores groups is processed into boolean form in a separate function to check any checkboxes on the form.

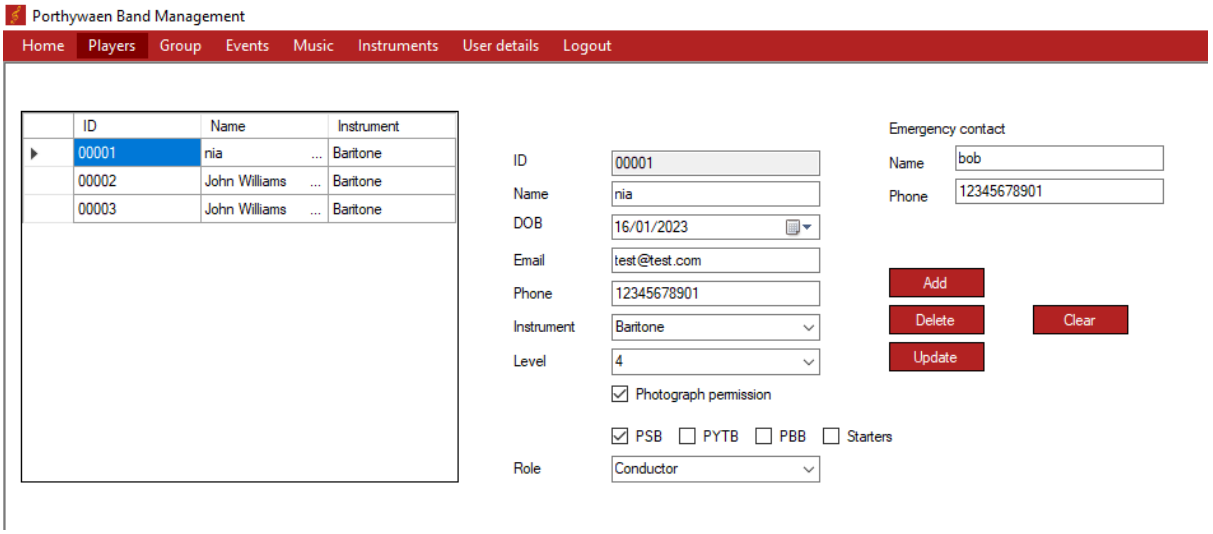

Figure 23: When a player is clicked in the "Players" screen.

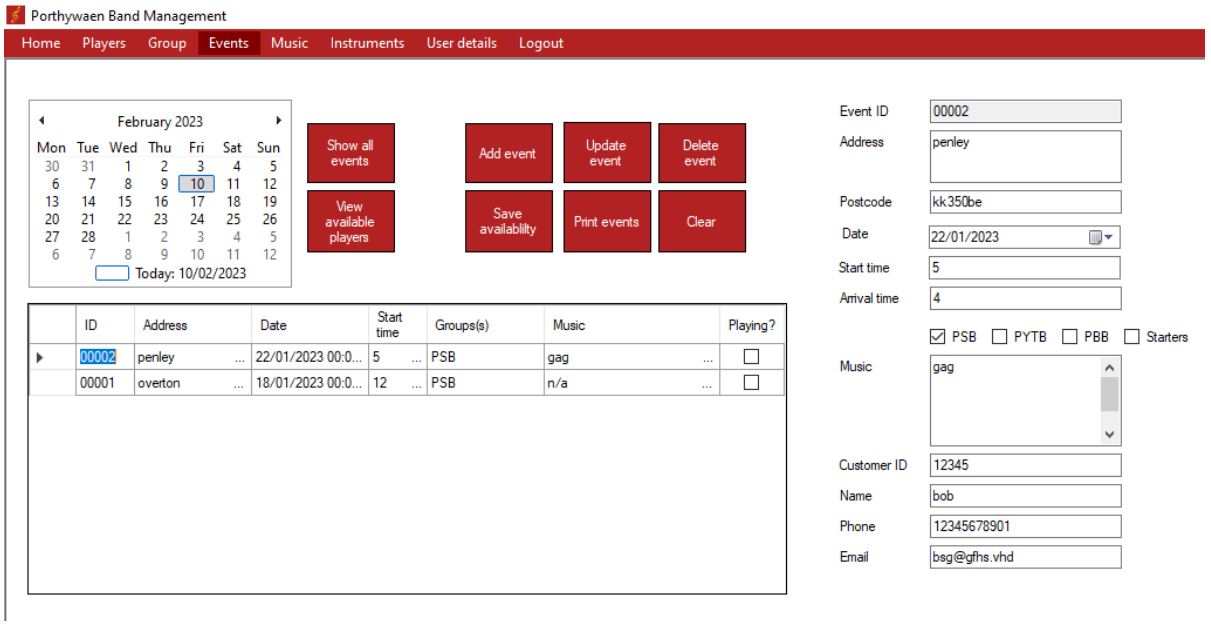

Figure 24: When an event is clicked in the "Events" screen.

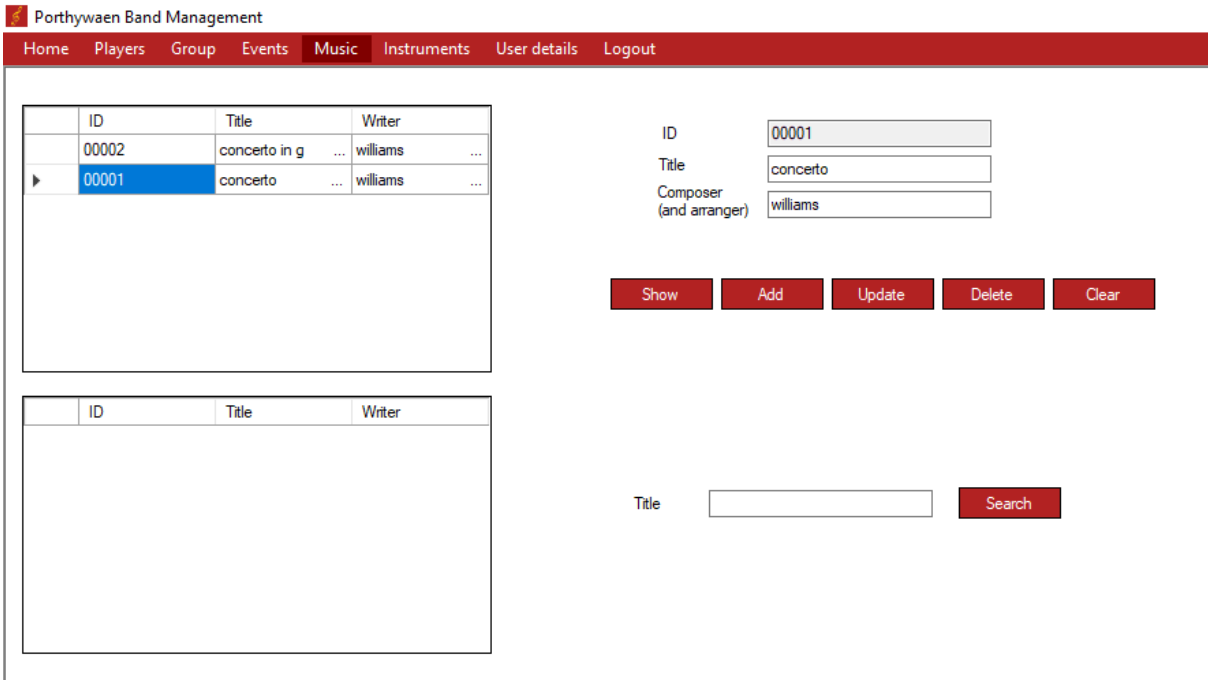

Figure 25: When a piece is clicked in the "Music" screen.

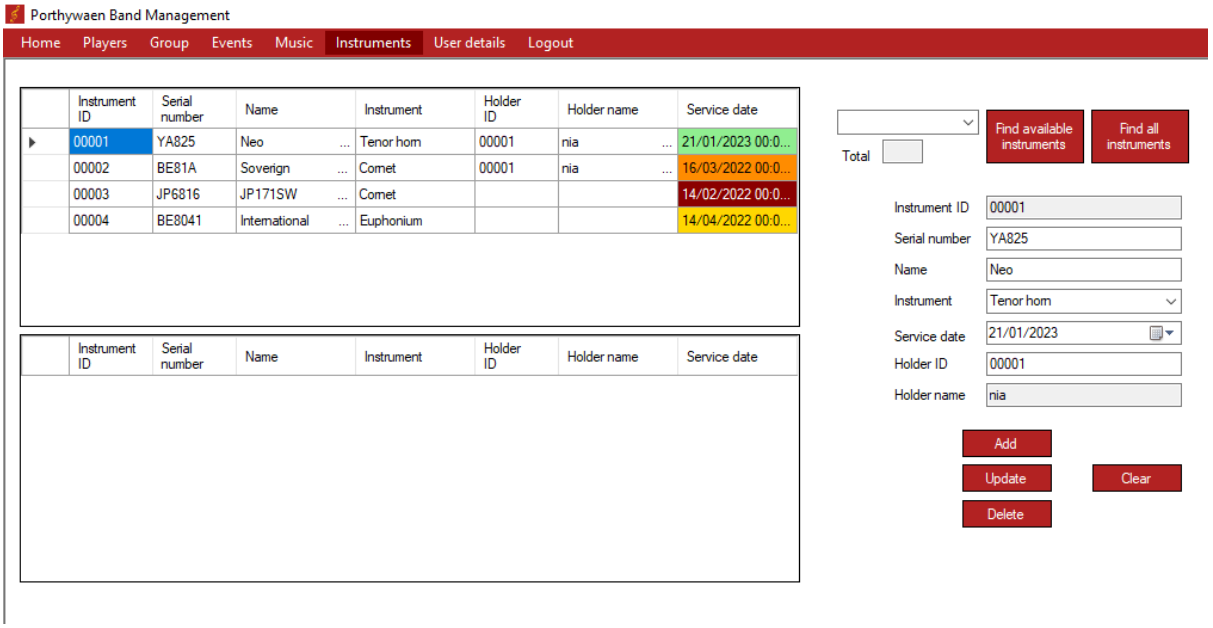

Figure 26: When an instrument is clicked in the "Instruments" screen.

Login and reset password.

Test 6

This test checks that a user is able to login to the system. When the correct login details of a member are entered and "Login" is clicked, the home screen is shown with the appropriate level of access for the user. Each record in the players file is read into a structure and the name and password element are compared to the inputs in the text boxes in the login screen. If they match, the home screen is shown.

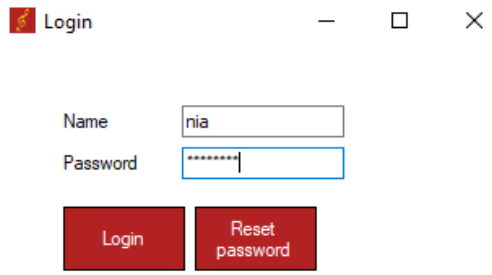

Figure 27: The "Login" screen, before the user clicks "Login".

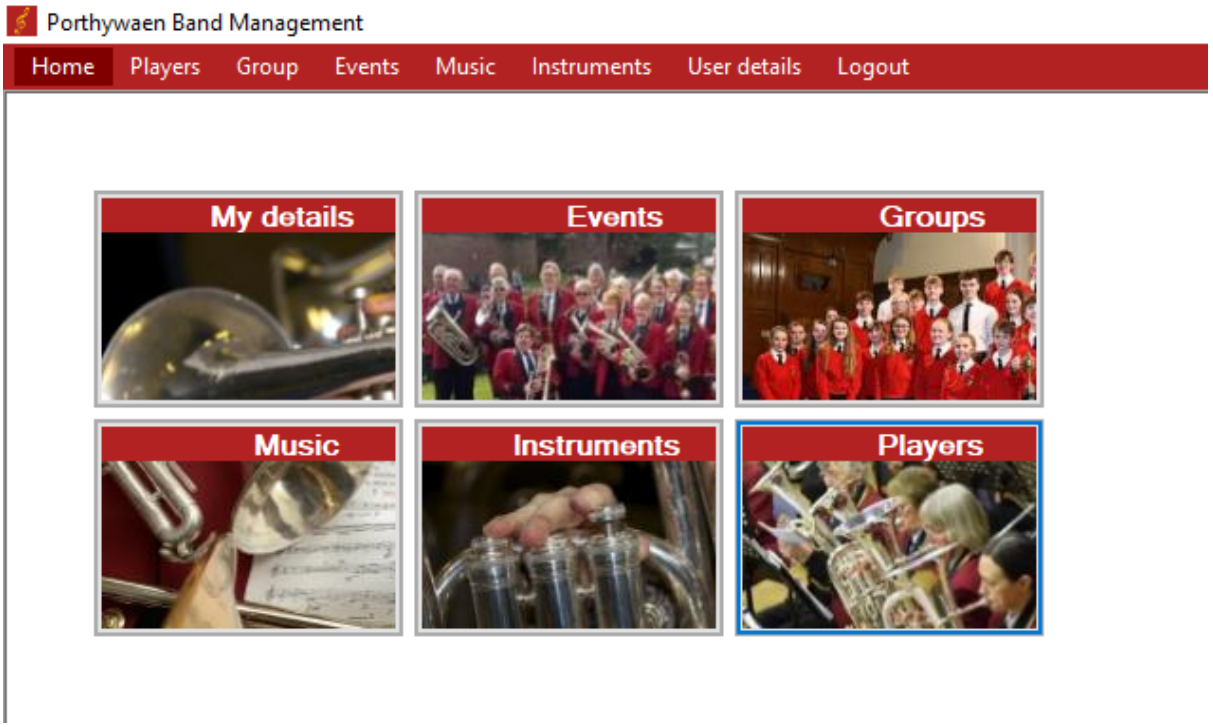

Figure 28: The "Home" screen with the conductor's level of access.

Test 7

This test checks if the program will notify the user if they do not enter their name when they login or reset their password. When only the password is entered and "Login" or “Reset password” is clicked, a message is shown and the user is allowed to input their name. Before searching the file for the player, the textboxes are checked to ensure that they are not empty to avoid errors.

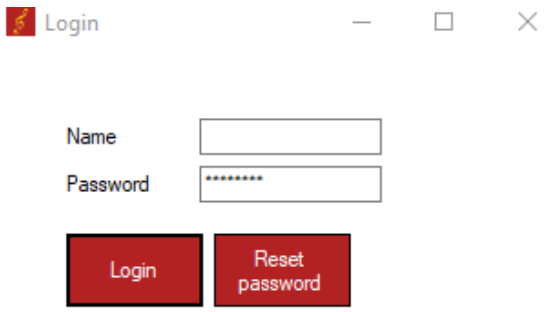

Figure 29: The “Login” screen before the user clicks “Login” or “Reset password”, without the name inputted.

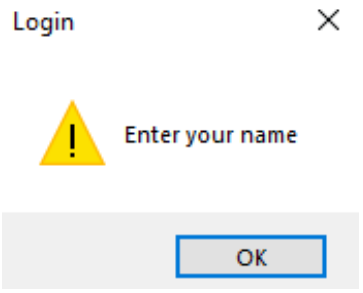

Figure 30: The message shown when the name is not inputted and “Login” is clicked.

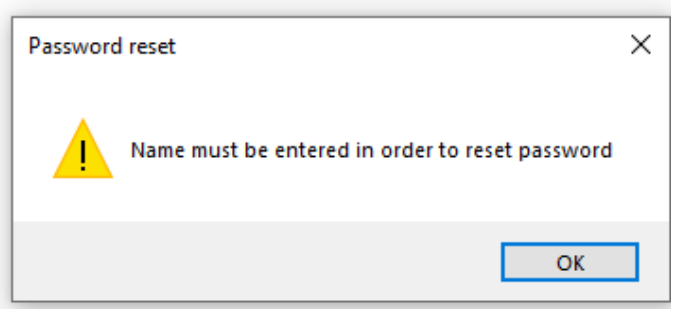

Figure 31: The message shown when the name is not inputted and “Reset password” is clicked.

Test 8

This test checks if a user is able to change their password. To do this, the user's name is entered and "Reset password" is clicked. When "Reset password" is clicked, a message is shown, informing the user that their password has been reset to their player ID. Each record in the players file is read into a structure and the name element is compared with the input in the textbox. If it matches, then the password element is set as the data in the ID element and the record is updated in the file.

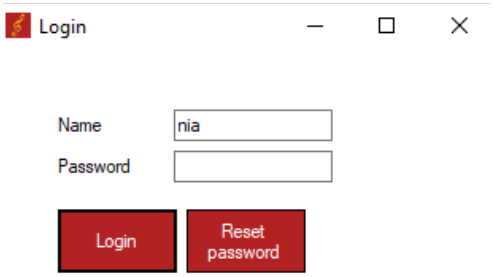

Figure 32: The "Login" screen before "Reset password" is clicked.

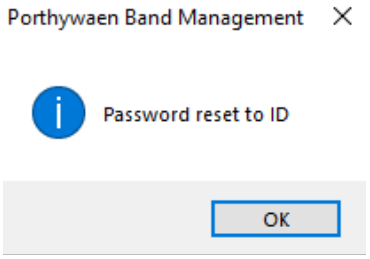

Figure 33: The message shown when "Reset password" is clicked and the password has been reset successfully.

Test 9

When a user has reset their password or they are logging in for the first time, and they enter their ID as their password and click “Login”, the change password screen is shown. This calls a subroutine which opens and searches the players file for the name that has been inputted. When the name is found and the password stored in the structure matches the player's ID, the change password screen is shown. The user is then able to input their new password twice to confirm it, and then click “Save and login”. This updates the user's password, and the user is allowed access to the system and the home screen is shown. A message is displayed to confirm the password has been changed.

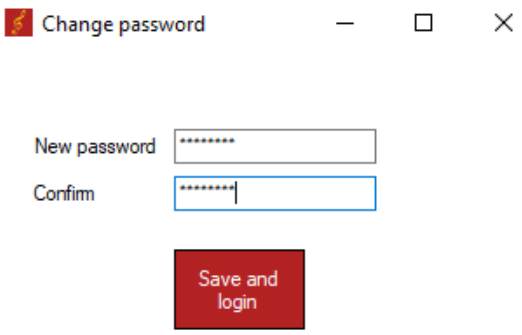

Figure 34: The “Change password” screen before the user clicks “Save and login”.

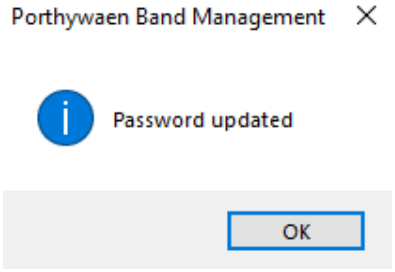

Figure 35: The message shown when “Save and login” is clicked and the password has been changed successfully.

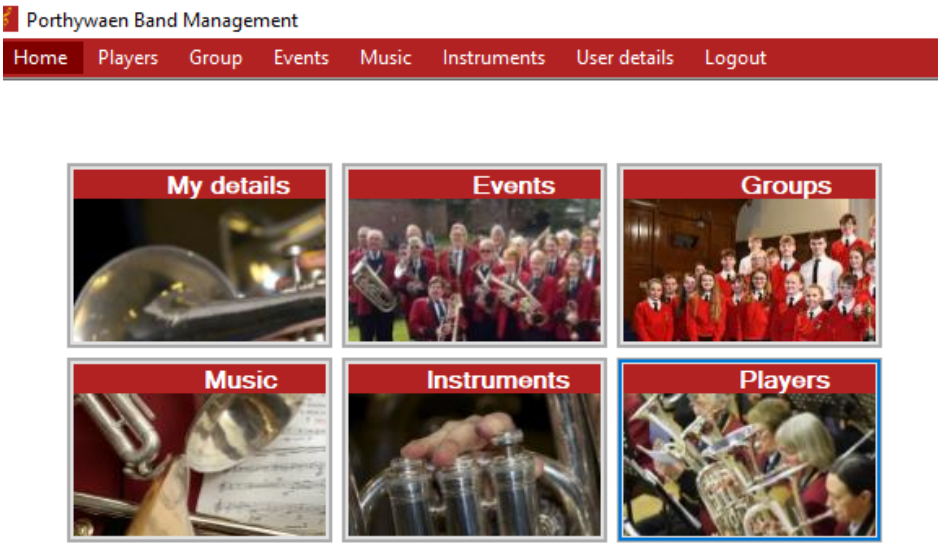

Figure 36: The "Home" screen with the conductor's level of access.

Test 10

This test checks that the user has entered their new password as they intended, using double entry. In the change password screen, if the data inputted into the textboxes do not match, a message is shown to inform the user, and they are allowed to amend their inputs.

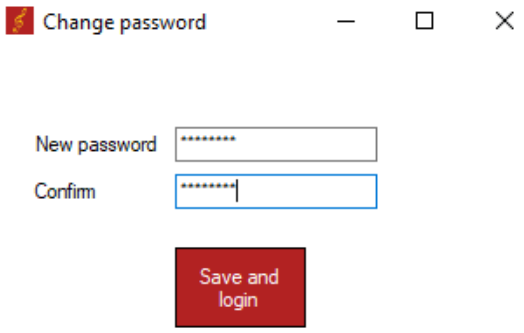

Figure 34: The "Change password" screen before the user clicks "Save and login".

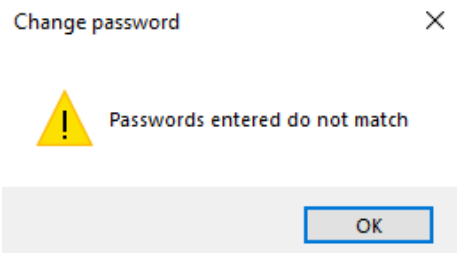

Figure 35: The message shown when "Save and login" is clicked and the passwords do not match.

Test 11

This test checks that the user must enter a password that is a minimum of 8 characters long. If a password 7 characters long is entered, a message is shown, informing the user that the password needs to be at least 8 characters long.

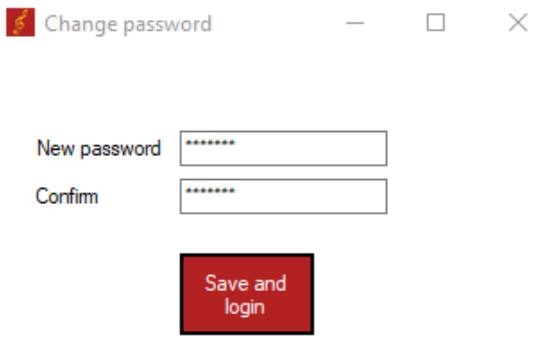

Figure 36: The "Change password" screen before the user clicks "Save and login".

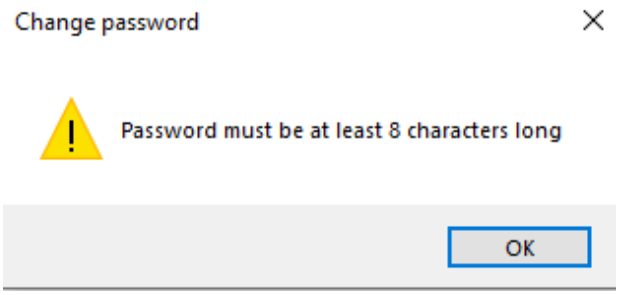

Figure 37: The message shown when "Save and login" is clicked and the password is too short.

Events

Test 12

Events are shown in a DataGridView. This test checks that events on a certain date are displayed. When the user clicks on a date in the calendar, any bookings are shown in the DataGridView. To do this, the file is searched and each record is stored in a structure. If the date element matches the date selected by the user, the record is added to the DataGridView.

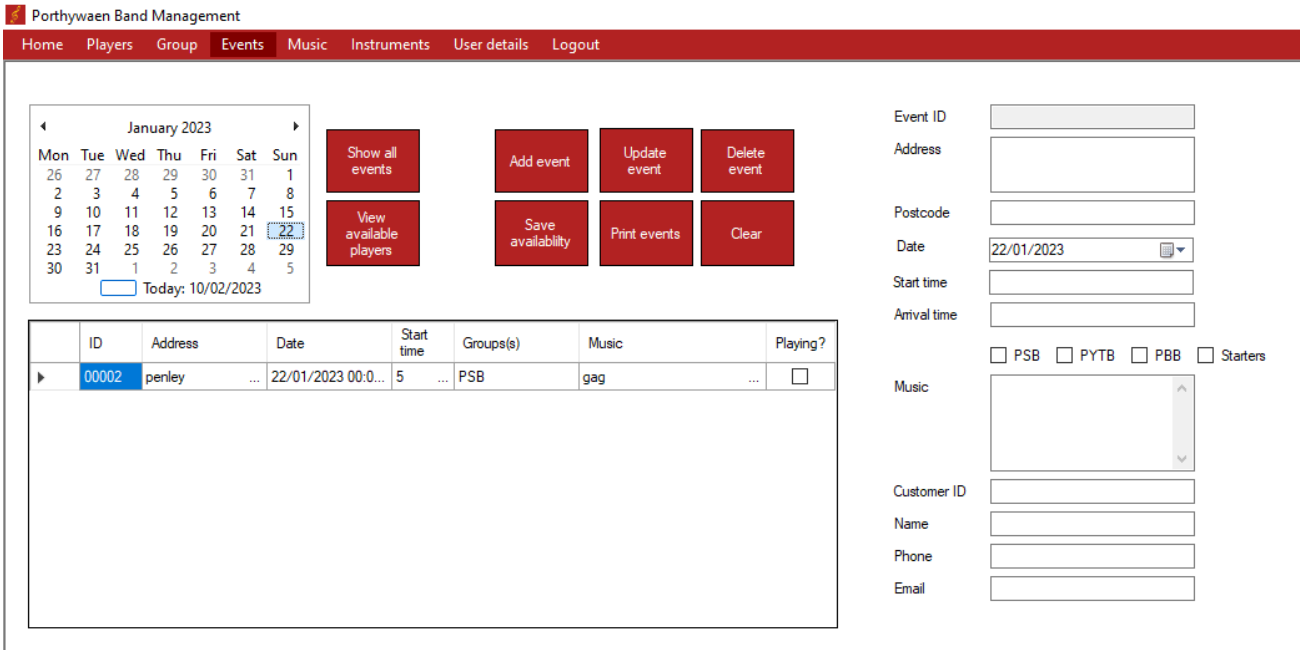

Figure 38: The "Events" screen after a date is selected.

I found when carrying out this test that if there are no events on a selected date, there is no message to tell the user this, so the user is unable to know if the system has failed or not. To fix this, I added the following lines of code which displays a message if no events with the selected date are stored in the database. The variable eventsFound is set to True when an event is added to the DataGridView.

```
If eventsFound = False Then
    MsgBox("No events booked for the selected date", vbInformation, "Events")
End If
```

When no events are found on a selected date, a message is displayed informing the user that there are no bookings on the selected date.

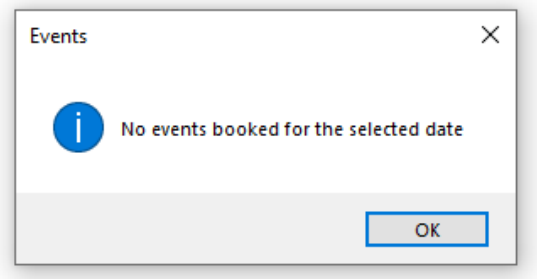

Figure 39: The message shown when no events are on the selected date.

Test 13

This test checks if an event can be added to the system. When the user enters data into the fields on the screen and clicks “Add event”, a subroutine is called which processes the inputted data. The data is validated and the event ID is generated by searching the file for a possible ID and incrementing the ID by 1 each time an ID is found. The final ID is padded with zeros to ensure it is the same length as all the other IDs. The data is appended to the file and the DataGridView is refreshed to show the changes, and a message is shown.

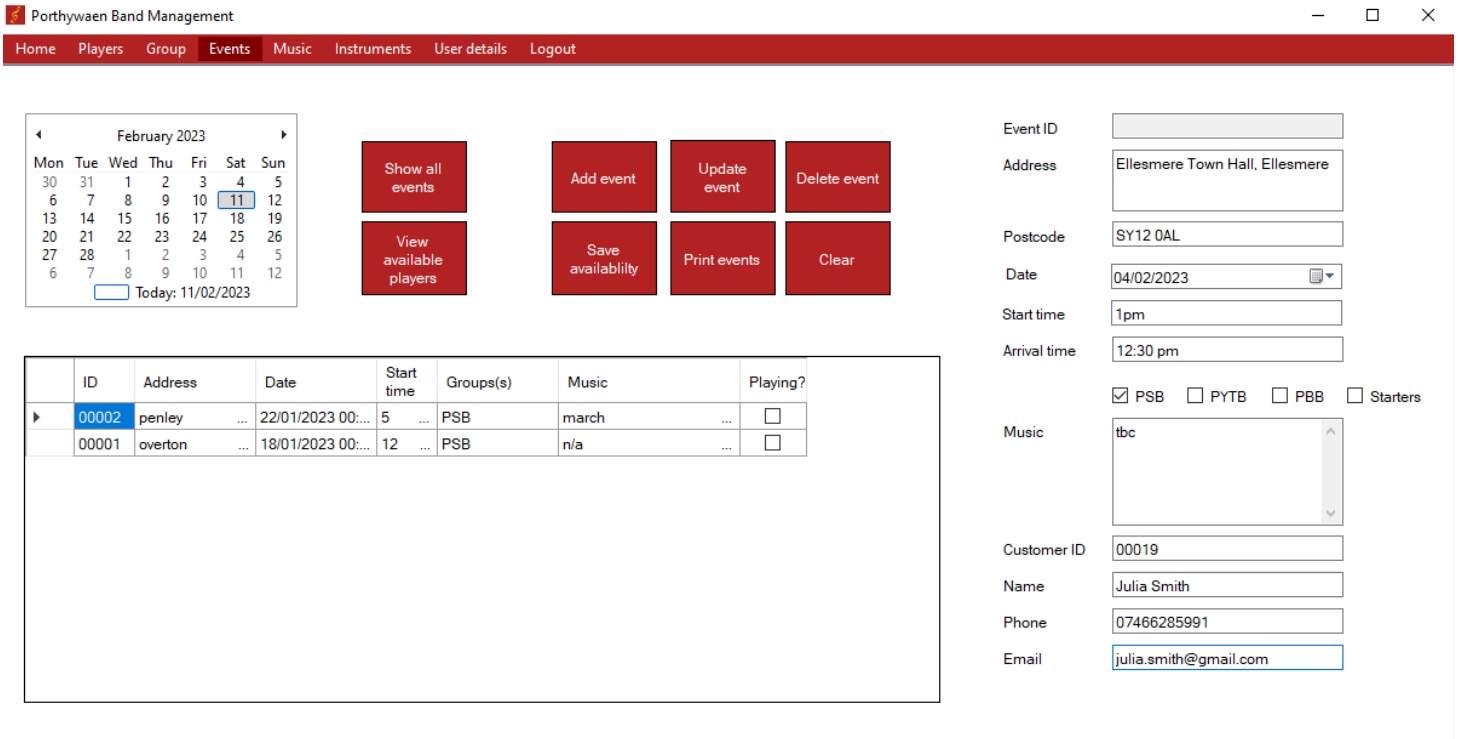

Figure 40: The “Events” screen with a new event inputted before “Add event” clicked.

When testing this, I came across an error. The following error message was shown despite a valid email being entered into the system.

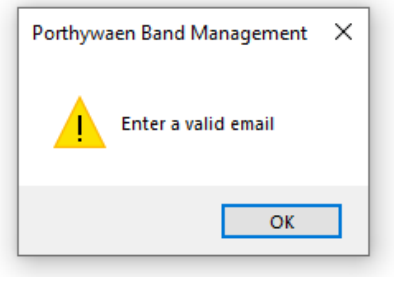

Figure 41: The message shown when “Add event” clicked.

After looking at the code, I found that the regex expression pattern used to validate the email address did not include special characters such as a “.” that was entered in the email used in this test. In order to fix this error, I changed the regex expression pattern to include special characters, and any number of blank spaces, which occur when a record is updated due to the way data is stored in the

database, in the event the data is not trimmed correctly. I also ensured textbox data was trimmed, to reduce the changes of any further runtime errors.

This is the regex expression pattern that works correctly in the program.

```
Dim postcodeFormat As String = "^[A-Z]{1,2}[0-9]{1,2}[A-Z]?\s?[0-9][A-Z]{2}([\s]+)?$"
Dim postcodeMatch As Match = Regex.Match(postcode, postcodeFormat)
```

When “Add event” is clicked after this correction, the event was successfully added to the file. A message was shown and the DataGridView was refreshed to show the new event.

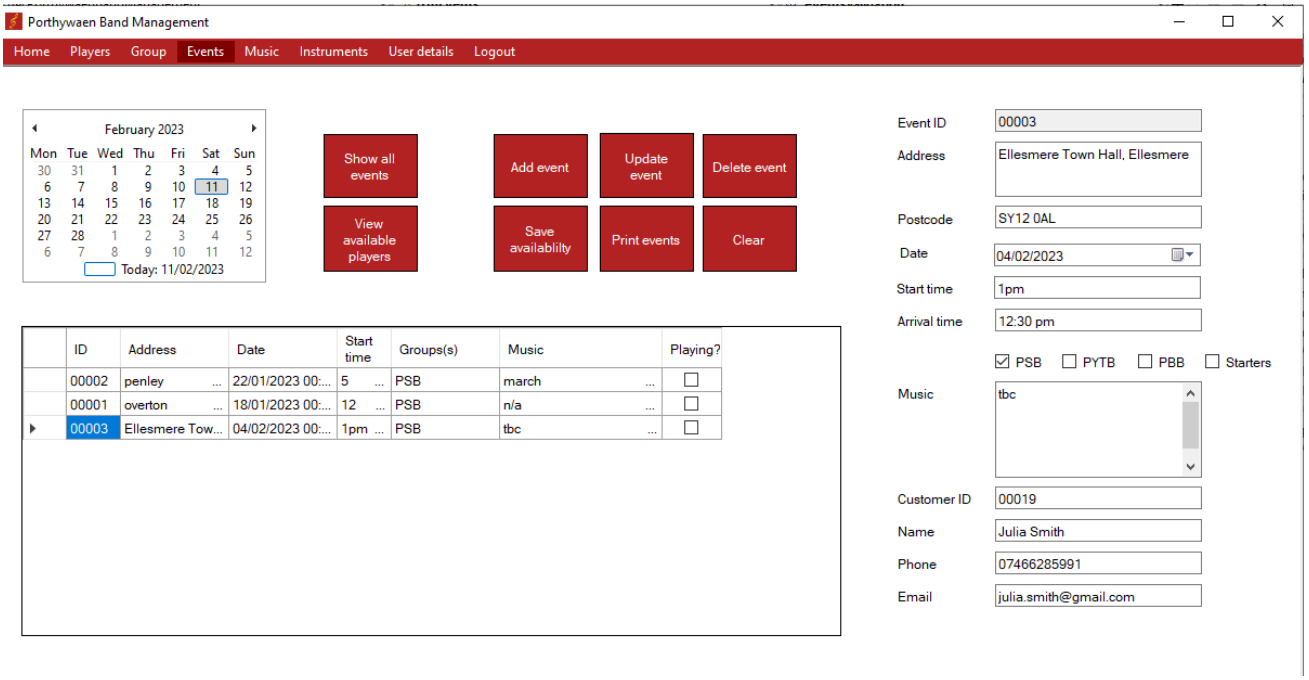

Figure 42: The “Events” screen after “Add event” is clicked.

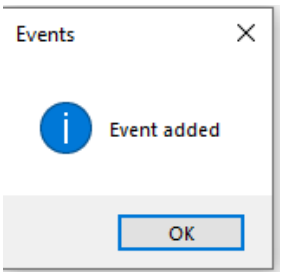

Figure 41: The message shown when “Add event” clicked and the event has been successfully added to the file.

Test 14

This test checks if an event can be updated. When an event is selected, the user can change any of the details and click "Update" . The data inputted into the system is validated and then stored in a structure. The record number of the record to be updated is determined from the DataGridView cell clicked and that record is updated in the file. A message is shown and the DataGridView is refreshed to show the changes.

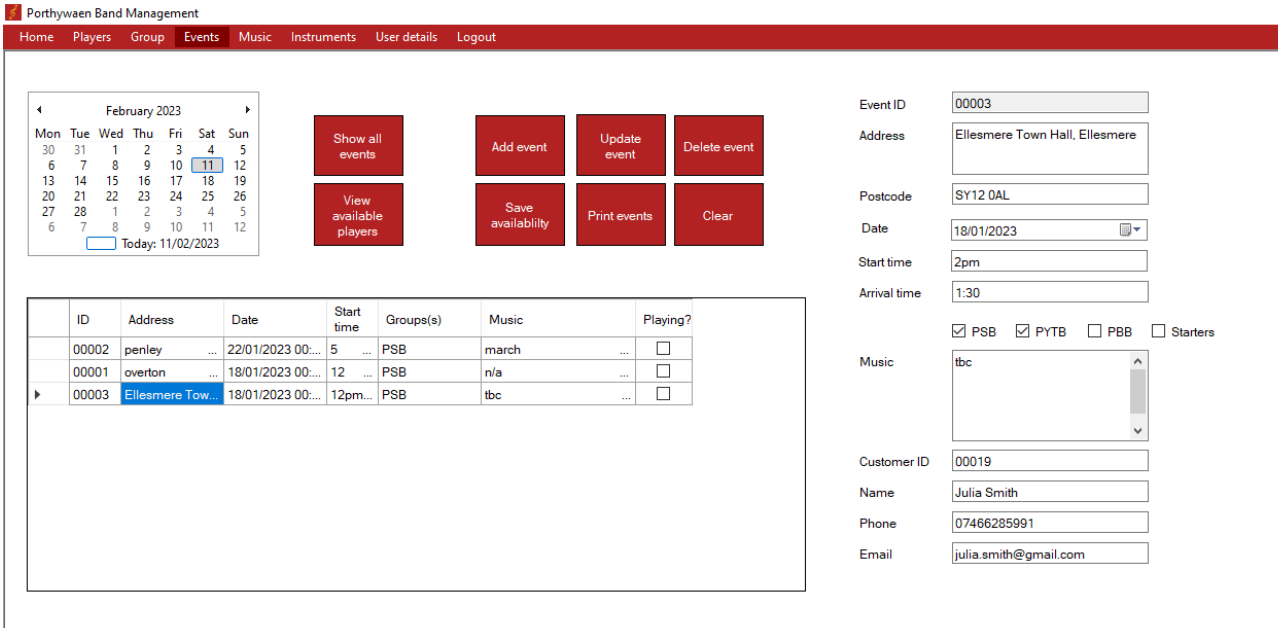

Figure 43: The "Events" screen before "Update event" is clicked.

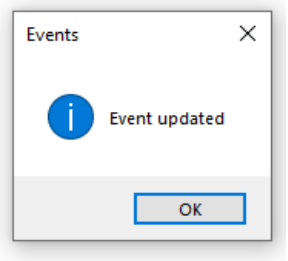

Figure 44: The message shown when "Update event" is clicked.

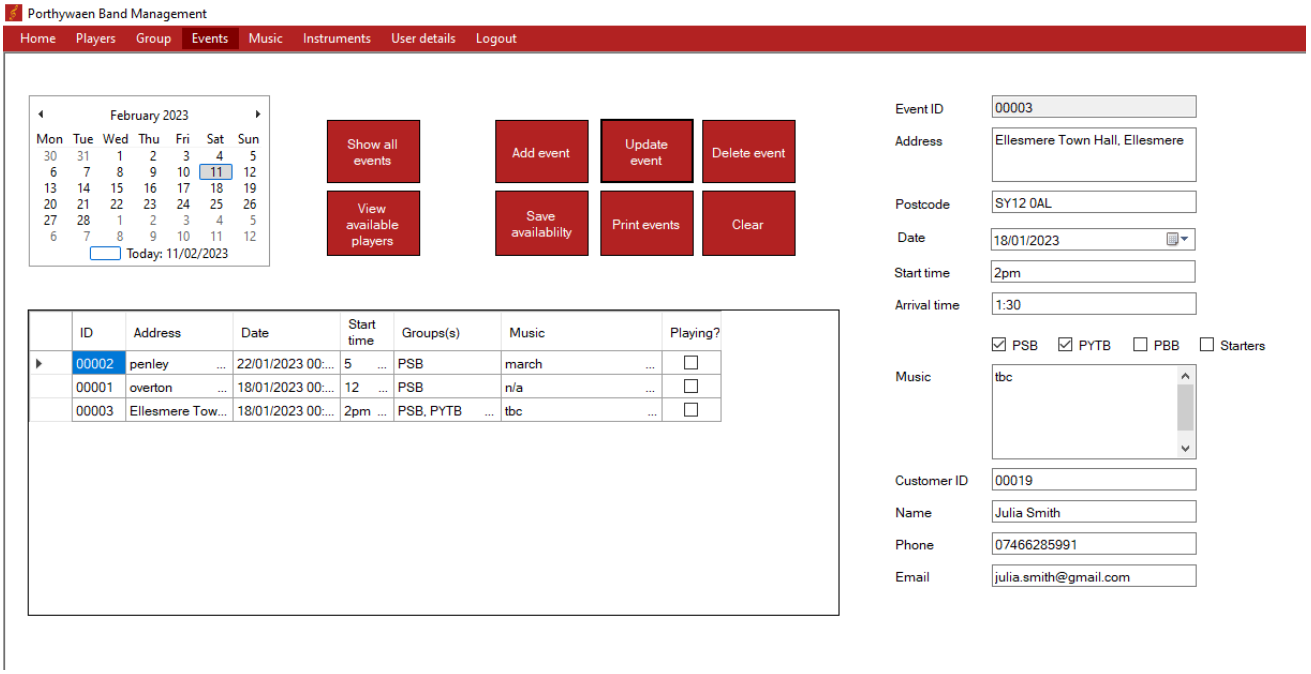

Figure 45: The "Events" screen after "Update event" is clicked.

Test 15

This test checks that the user must enter the address where the event is being held. When data is entered into all the input fields except the address textbox and “Update event” or “Add event” is clicked, a message is shown and no data is stored in the file.

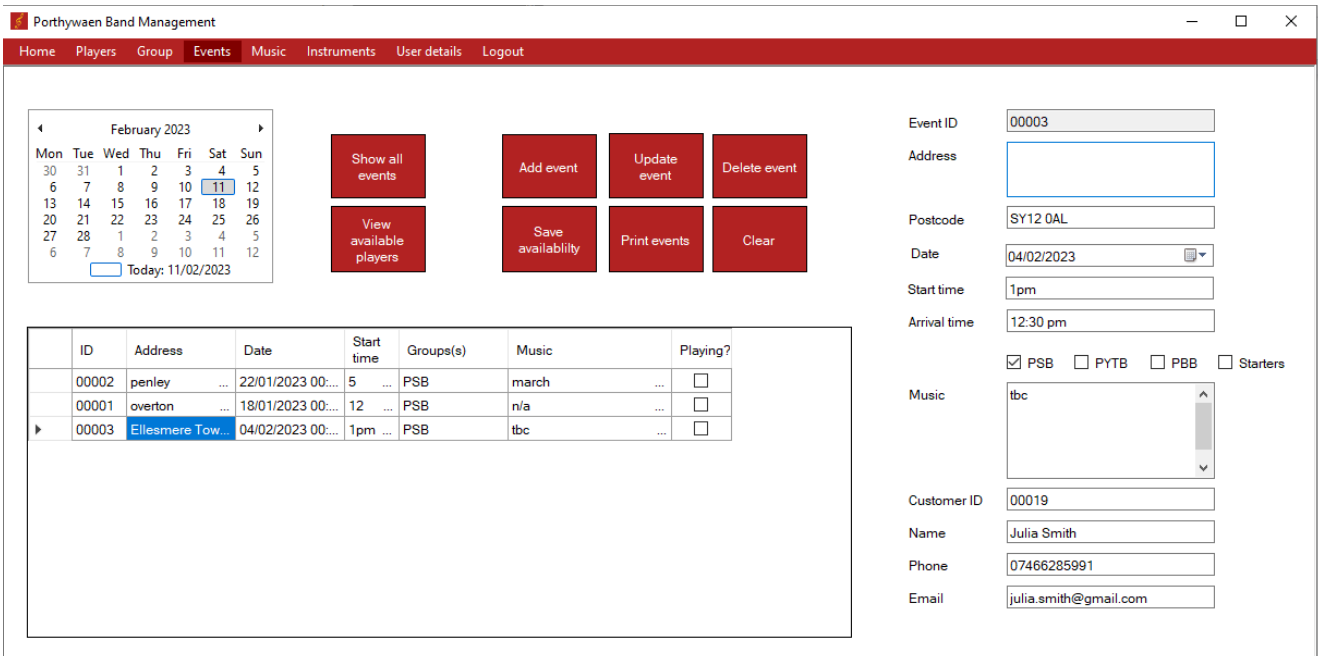

Figure 46: The “Events” screen before “Update event” or “Add event” is clicked and no address has been inputted.

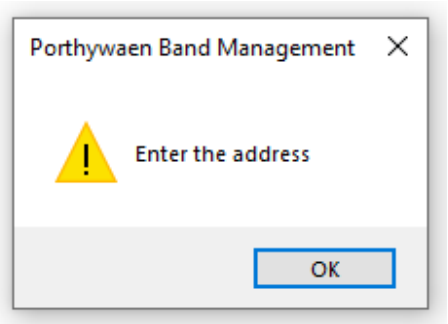

Figure 47: The message shown after “Update event” or “Add event” is clicked and no address has been inputted.

Test 16

This test checks that the user must input the postcode in the typical postcode format. When data that is not in postcode format is entered, a message is shown and no data is stored in the database.

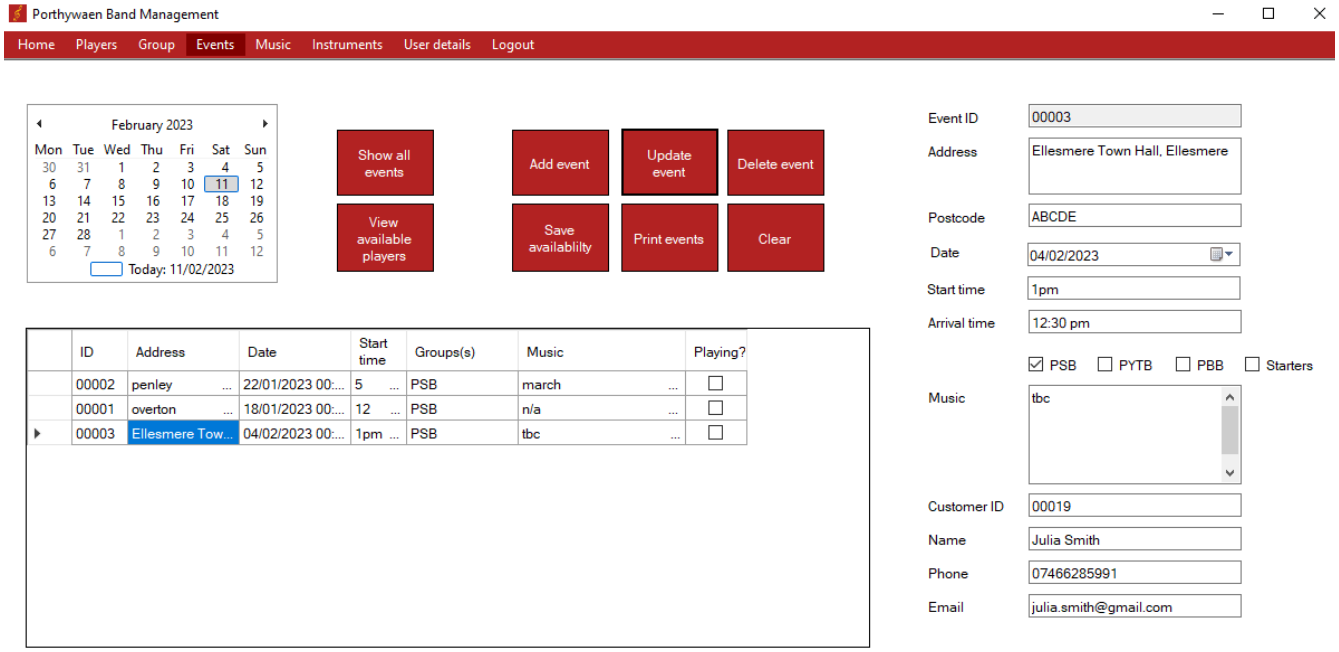

Figure 46: The "Events" screen before "Update event" or "Add event" is clicked and the postcode inputted is in an invalid format.

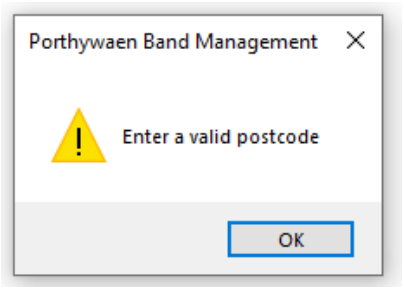

Figure 47: The message shown after "Update event" or "Add event" is clicked and the postcode inputted is in an invalid format.

Test 17

This test checks that the user must input a phone number that is 11 characters long. When the phone number is not 11 characters long a message is shown and the data will not be stored to the database.

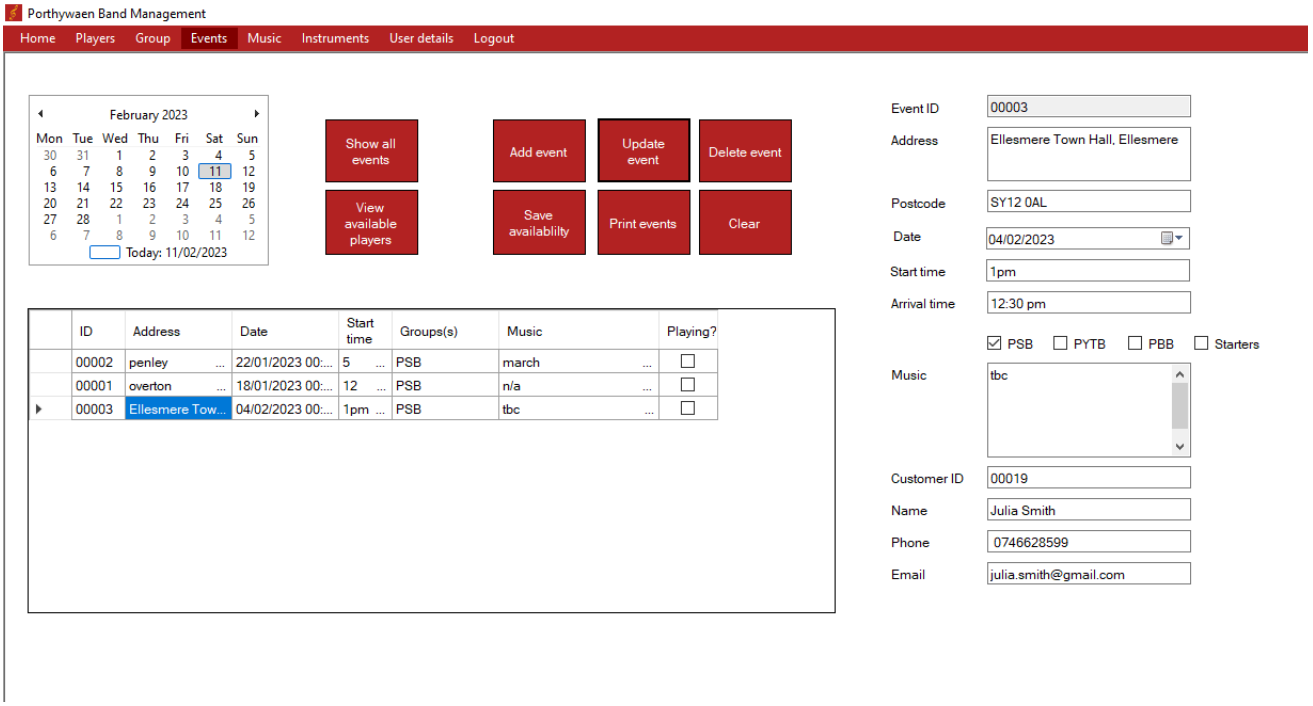

Figure 48: The "Events" screen before "Update event" or "Add event" is clicked and the phone number inputted is too short.

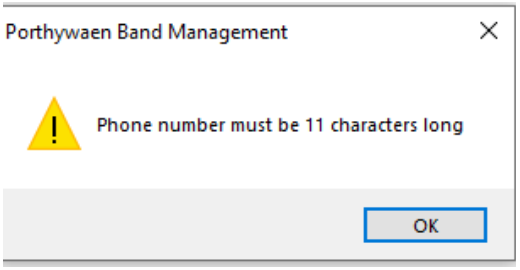

Figure 49: The message shown when "Update event" or "Add event" is clicked and the phone number inputted is too short.

Test 18

This test checks the date is within an acceptable range. When a date that is before the year 1900 is entered, a message is shown, and no data is stored in the database.

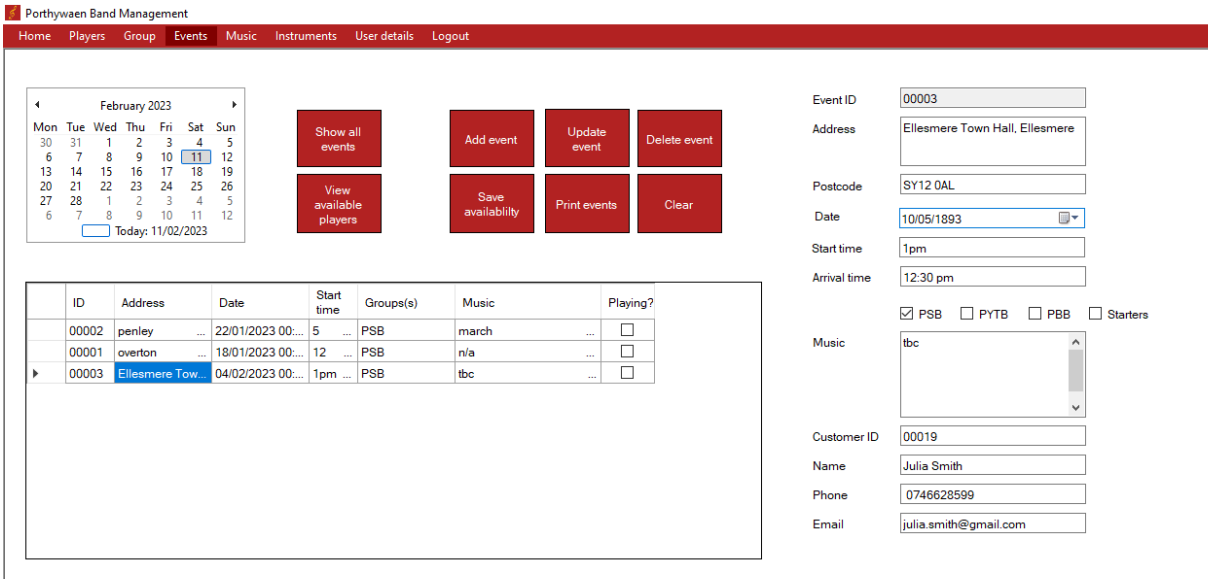

Figure 50: The "Events" screen before "Update event" or "Add event" is clicked and the date is invalid.

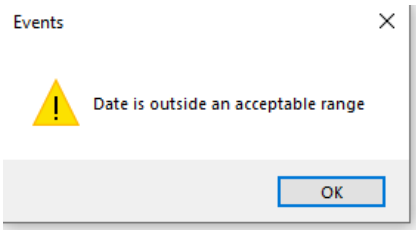

Figure 51: The message shown when "Update event" or "Add event" is clicked and the date is invalid.

Test 19

This test checks that the date selected for the booking is available. When all data is inputted correctly and “Add” is clicked, if a booking is found that is on the same data as inputted for the new event a message is shown. The user can choose whether to add the event to the system or not.

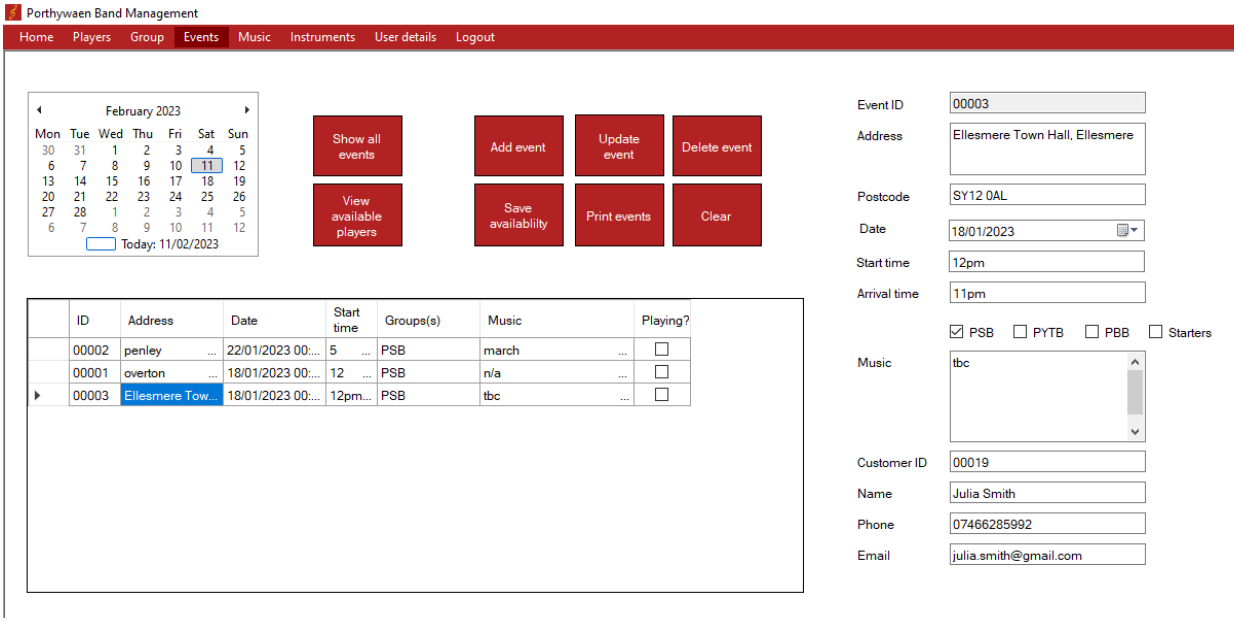

Figure 52: The “Events” screen before “Add event” is clicked.

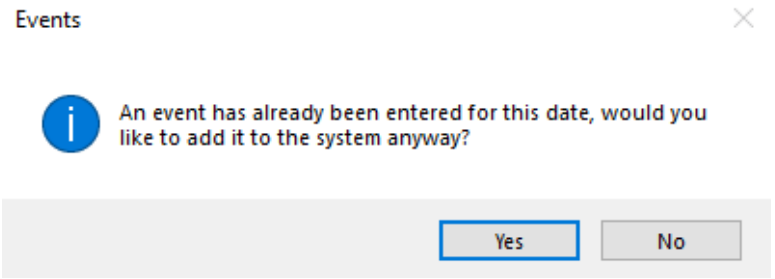

Figure 53: The message shown when “Add event” is clicked and there is another event booked on the inputted date.

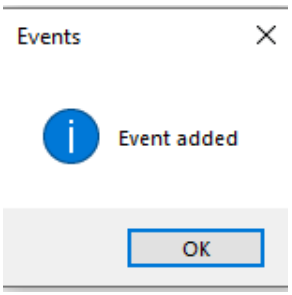

Figure 54: The message shown after “Yes” is clicked and the event has been added to the file.

Test 20

This test checks if an event can be deleted. When an event is selected and "Delete" is clicked, the event will be removed from the database. Each record in the file is compared to the record that has been selected to be deleted. If it is not this record, then it is added to a temp file. When the record that is to be deleted is found, the record is skipped and not added to the temp file. When the end of the file is reached, the temp file is renamed and the old file is deleted. A message is shown and the DataGridView is refreshed to remove the event. This event remains in the input fields so the user is able to add the event again in case they accidentally deleted it.

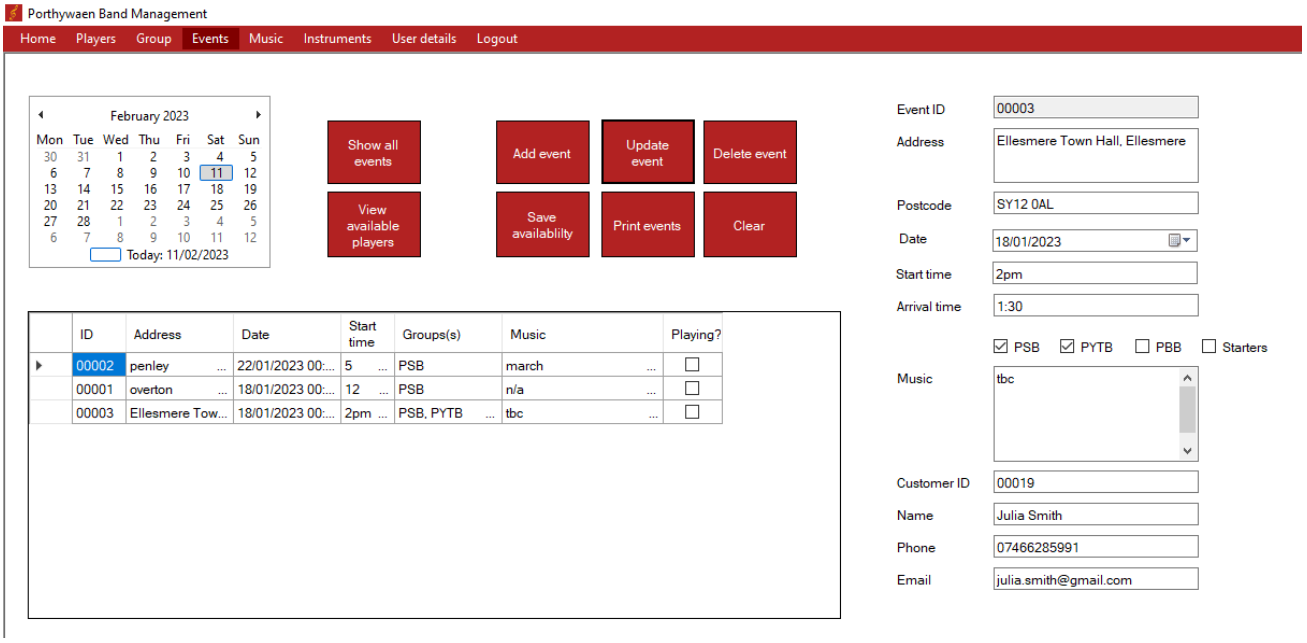

Figure 55: The "Events" screen before "Delete event" clicked.

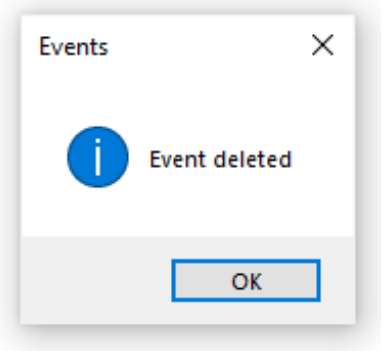

Figure 56: The message shown when "Delete event" clicked.

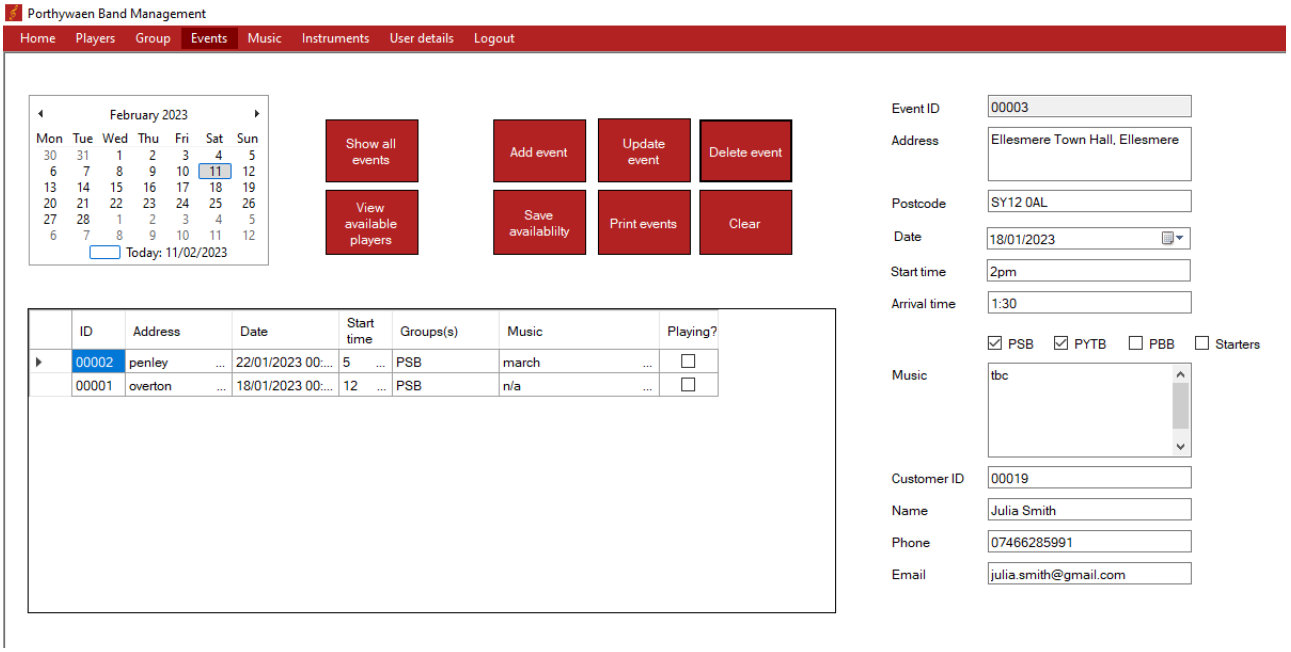

Figure 55: The "Events" screen after"Delete event" clicked.

Test 21

This test checks if the events can be printed. When “Print events” is clicked, a bitmap is made of the DataGridView, and a print preview is shown. The user can then print from the print preview.

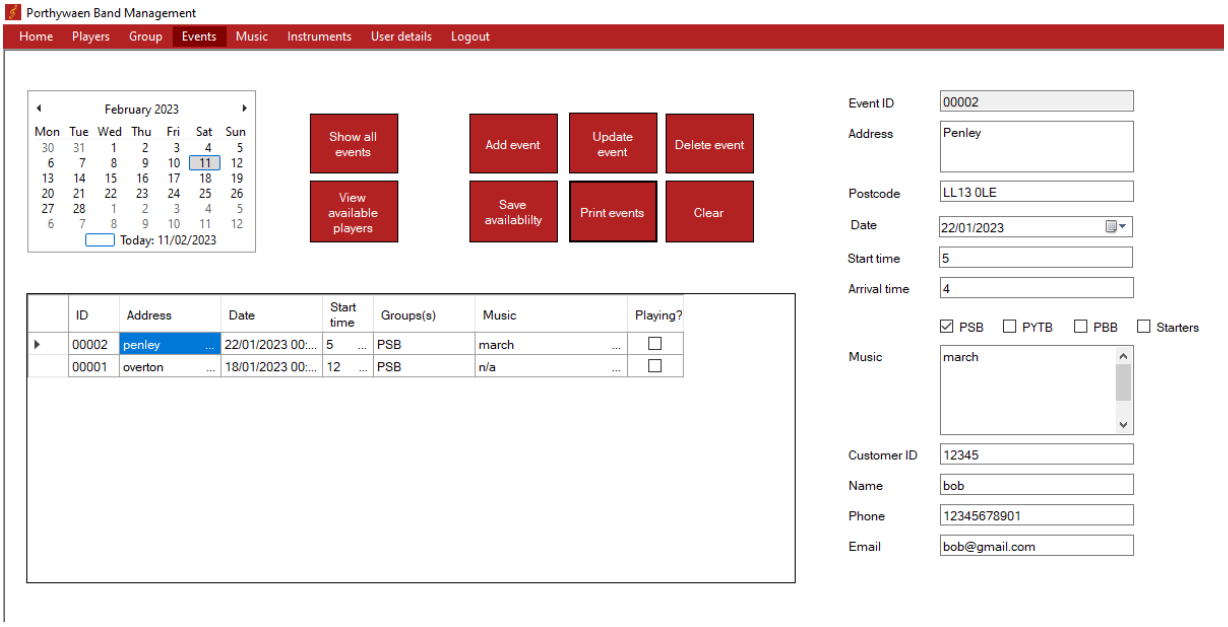

Figure 56: The “Events” screen before “Print events” is clicked.

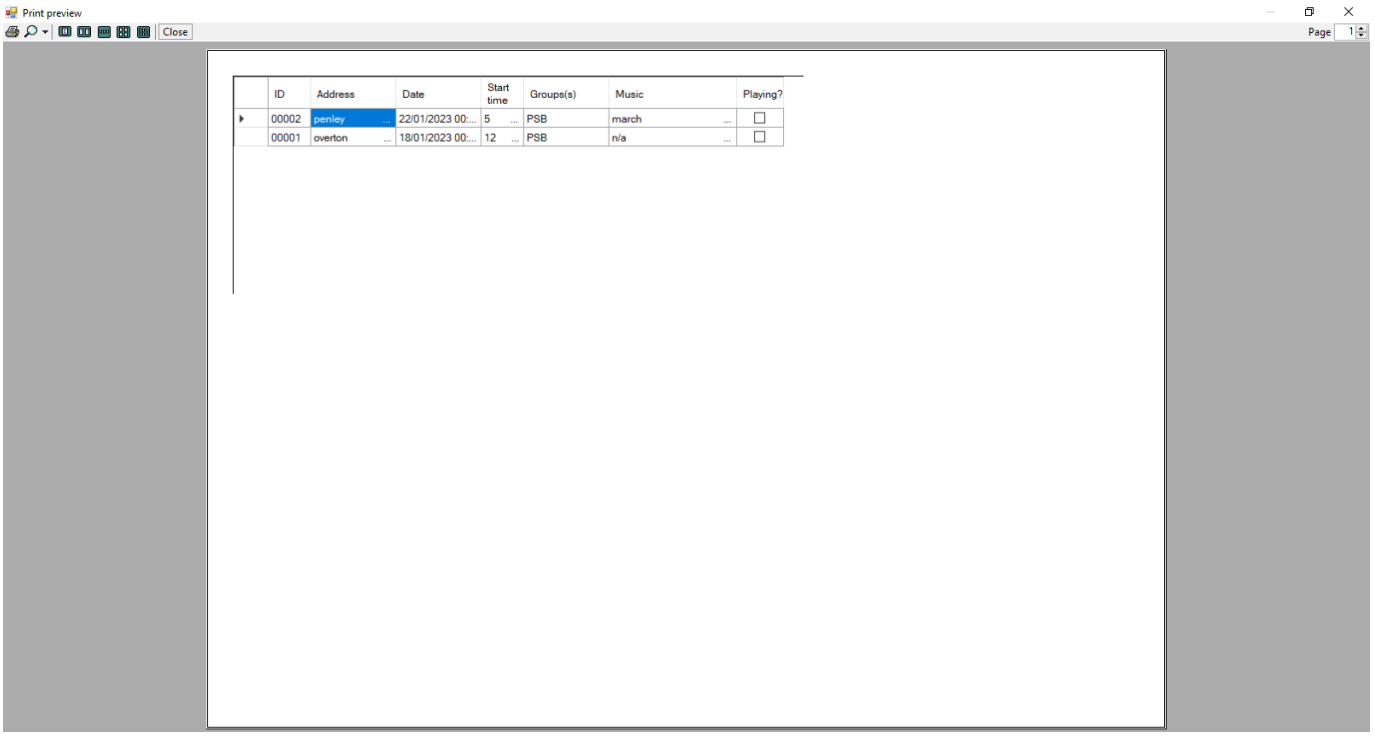

Figure 56: The print preview shown when “Print events” is clicked.

Test 22

This test checks if the user is able to save their availability for events. The user can check the checkboxes in the DataGridView next to each event to indicate that they are available to play in the event. The responses can be saved by clicking “Save availability”. For each event in the DataGridView in turn, the file is opened and searched for any responses for the event. If any responses are found, they are split by player and stored in an array. If responses for the event are found, each array element is checked for the user's ID. If it is not, then all the responses are recombined from the array and the user's response is appended. The response is then saved in the file and a message is shown.

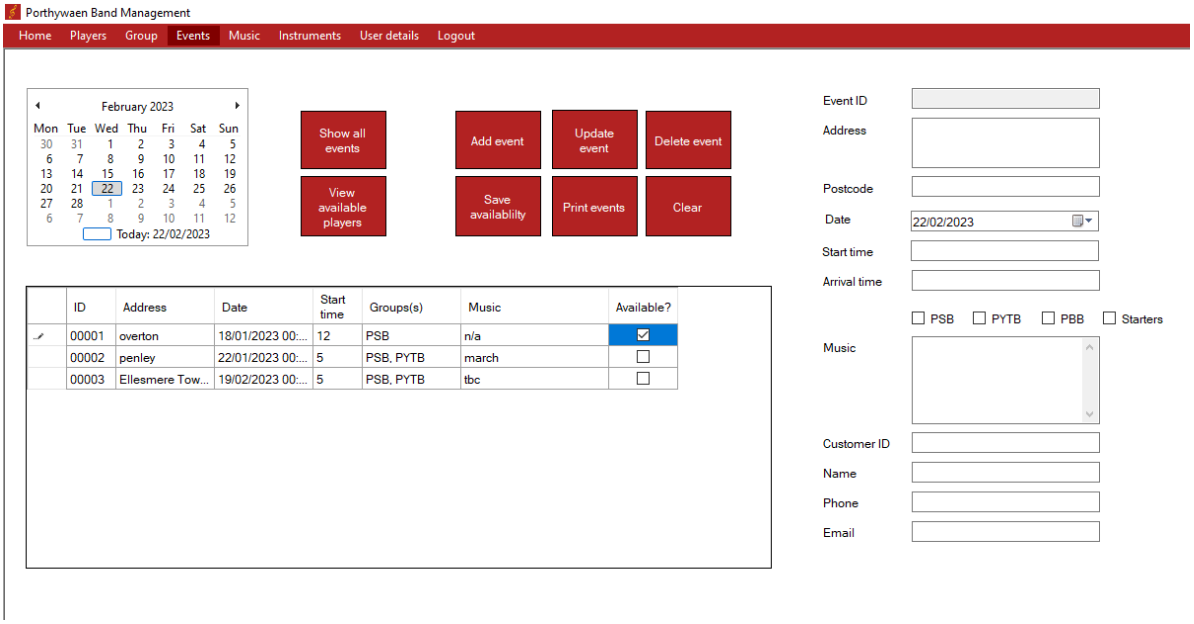

Figure 57: The “Events” screen with availability marked in the DataGridView before “Save availability is clicked.

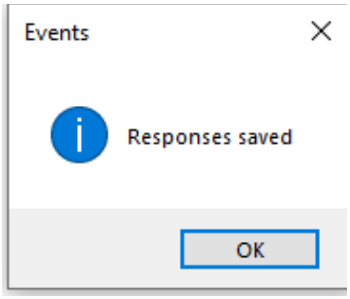

Figure 58: The message shown when the users availability is saved when “Save availability” is clicked and the availability has been saved in the file.

Test 23

This test checks if the user only has one response for each event. For each event in the DataGridView in turn, the file is opened and searched for any responses for the event. If any responses are found, they are split by player and stored in an array. If responses for the event are found, each array element is checked for the user's ID. If the user's ID is found, the response stored in the array is compared to the inputted response. If they match, a message is shown and no changes are made to the responses for the event.

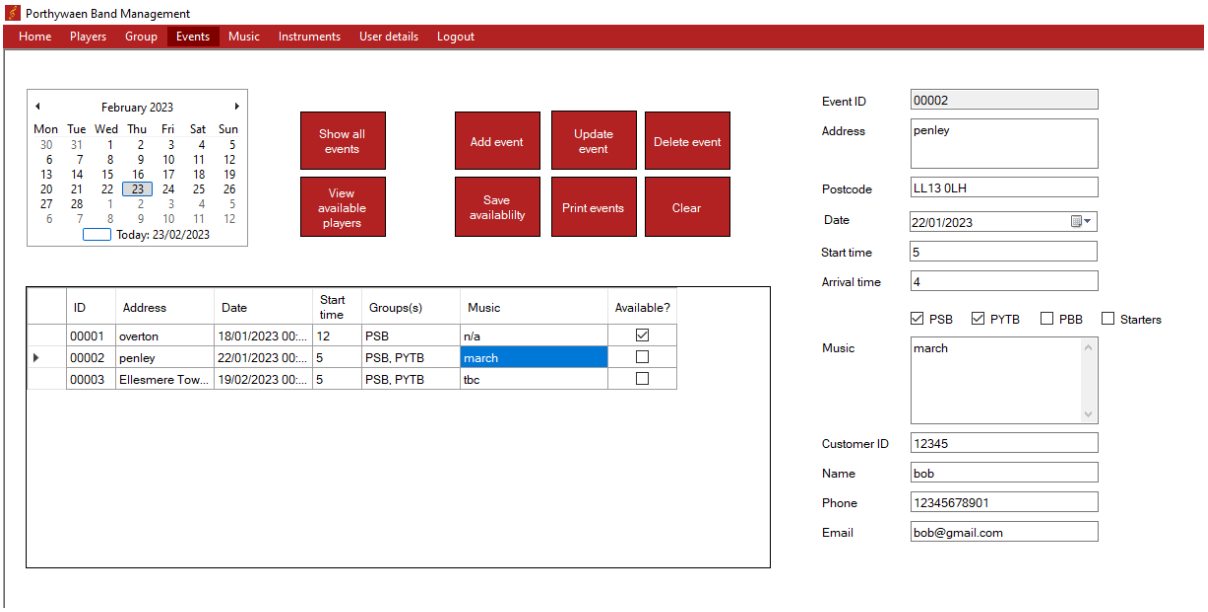

Figure 59: The "Events" screen with availability marked in the DataGridView before "Save availability is clicked.

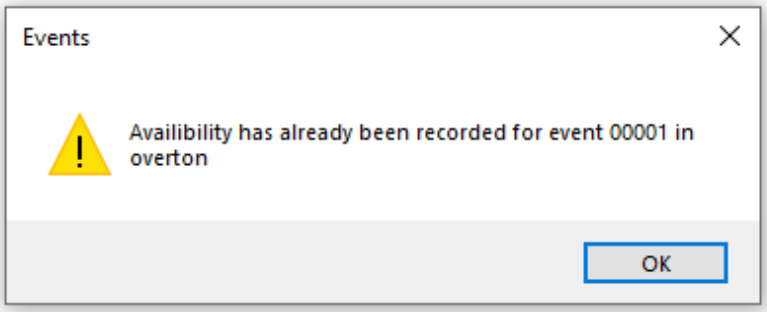

Figure 60: The message shown when the same availability has been recorded for an event when "Save availability" is clicked.

Test 24

This test checks that a player's availability for an event can be updated. For each event in the DataGridView in turn, the file is opened and searched for any responses for the event. If any responses are found, they are split by player and stored in an array. If responses for the event are found, each array element is checked for the user's ID. If the user's ID is found, the response stored in the array is compared to the inputted response. If they do not match, a message is shown, asking the user if they wish to update their availability for the event. If "No" is clicked, no changes are made to the user's availability for the event. If "Yes" is clicked, the user's availability is updated, which is confirmed by a message.

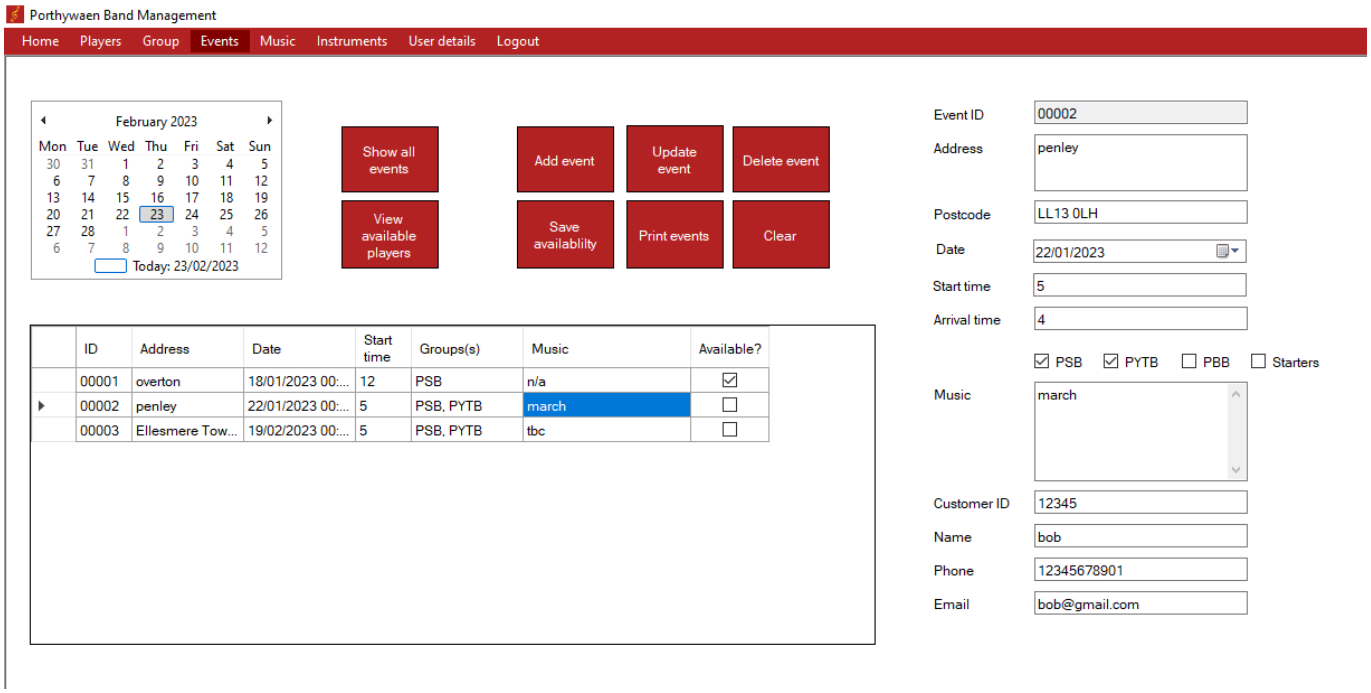

Figure 61: The "Events" screen with availability marked in the DataGridView before "Save availability is clicked.

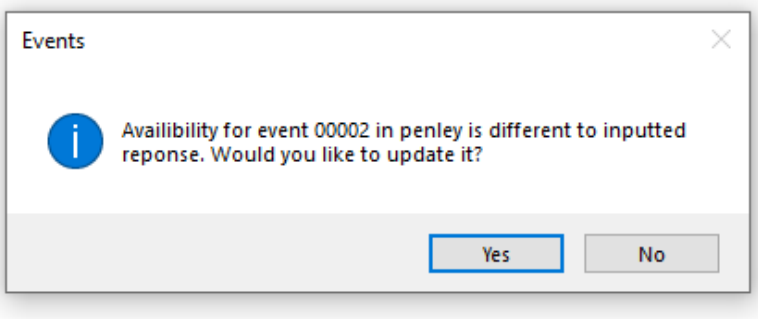

Figure 60: The message shown when different availability has been recorded for an event when "Save availability" is clicked.

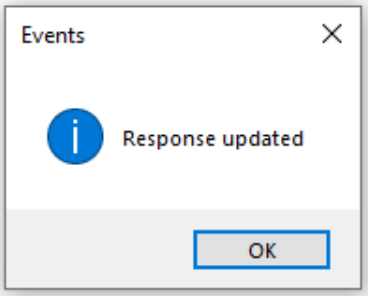

Figure 60: The message shown when "Yes" is clicked and the availability has been updated.

Test 25

This test checks users can see the available players for an event. When an event is selected and “View available players" is clicked, the response file is searched for the event ID which is found from the DataGridView row clicked. When the event is found, the record is stored in a structure. Each response recorded in the responses element is split and stored in an array for each player ID. The players file is opened and searched for each playerID in the array and the player’s ID, name and instrument are added to the table. The event booking file is also opened and searched for the event ID and some details are shown to make it easier for the user to know which event the players shown are available for.

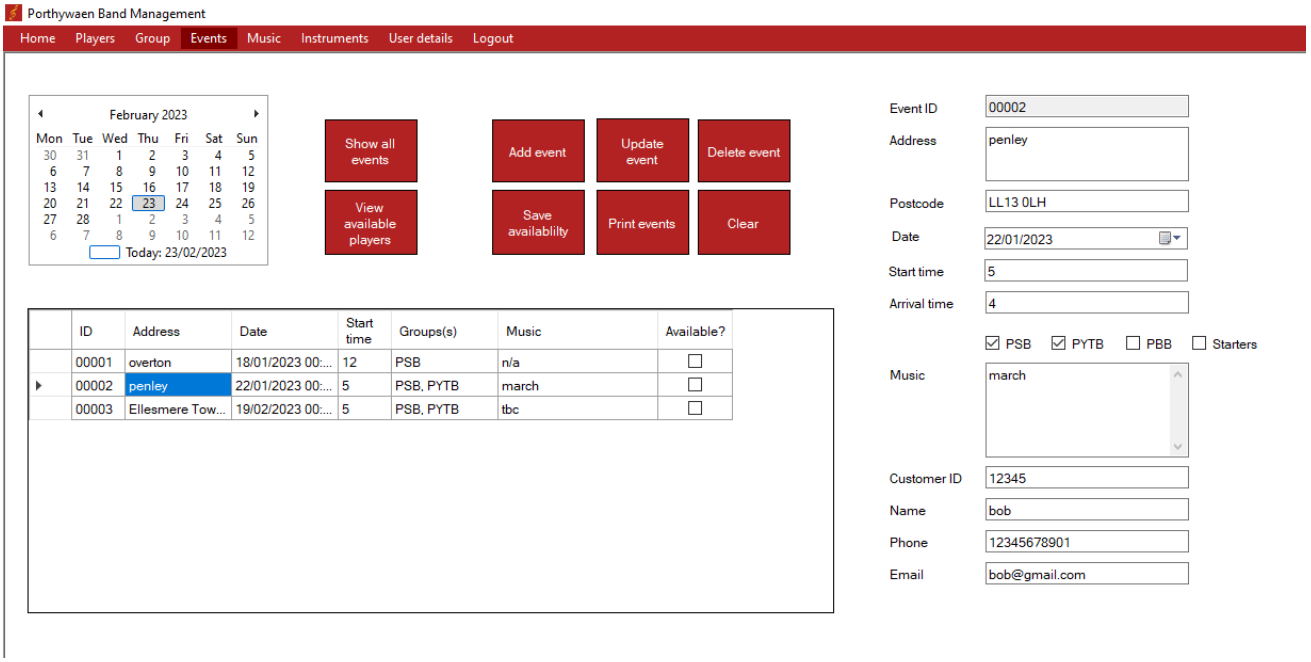

Figure 62: The “Events” screen with an event selected before “View available players” is clicked.

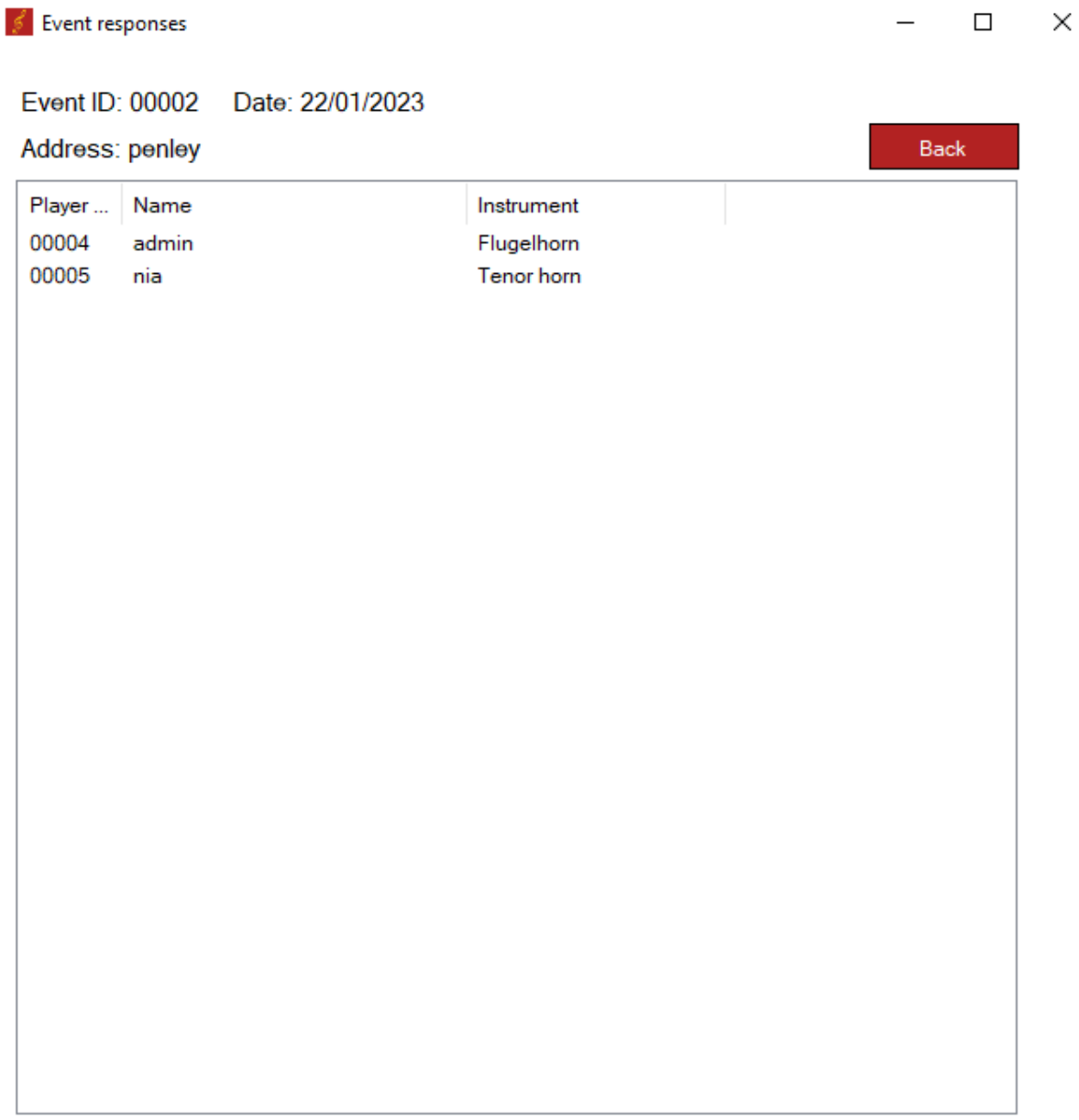

Figure 63: The table that shows the players that are available for the selected event.

Test 26

This test checks that an event must be selected to show the available players for an event. When “View available players” is clicked, the subroutine checks that a response has been stored. If the user clicks “View available players” without clicking an event in the DataGridView beforehand, a message is shown informing them that they need to select an event before viewing the players availability.

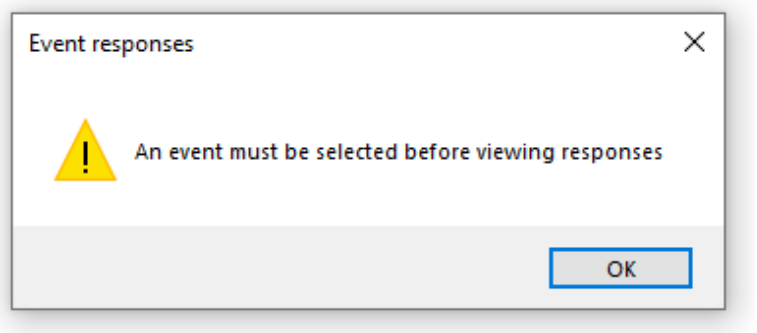

Figure 64: The message shown if a user does not select an event before clicking “View available players”.

Group

Test 27

This test checks if a group can be shown. When the user selects a group, the players file is opened and each player record is stored in a structure. If the group element of the structure contains the groups inputted, the player is added to the DataGridView. If the end of the file is reached and there are no players in the group selected, a message is shown.

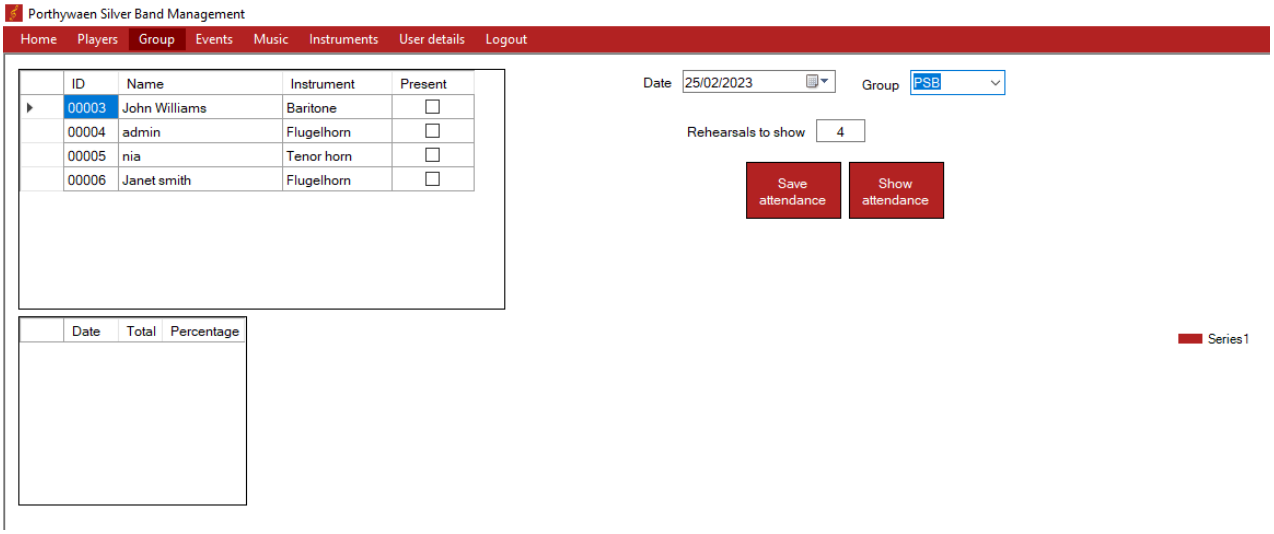

Figure 65: The "Group"screen when "PSB" is selected.

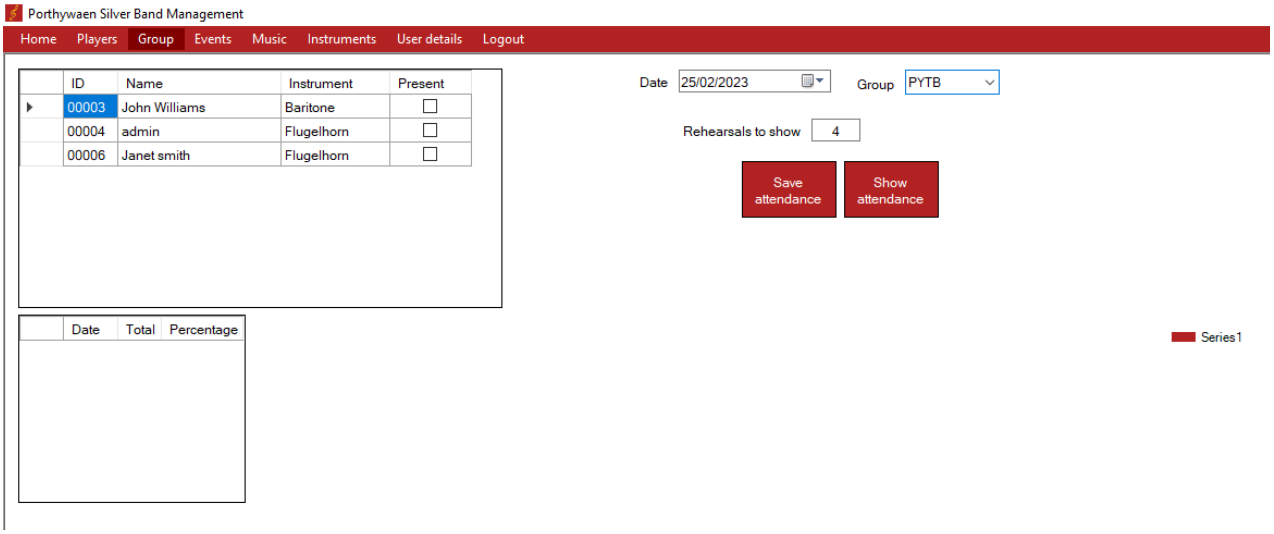

Figure 66: The "Group"screen when "PYTB" is selected.

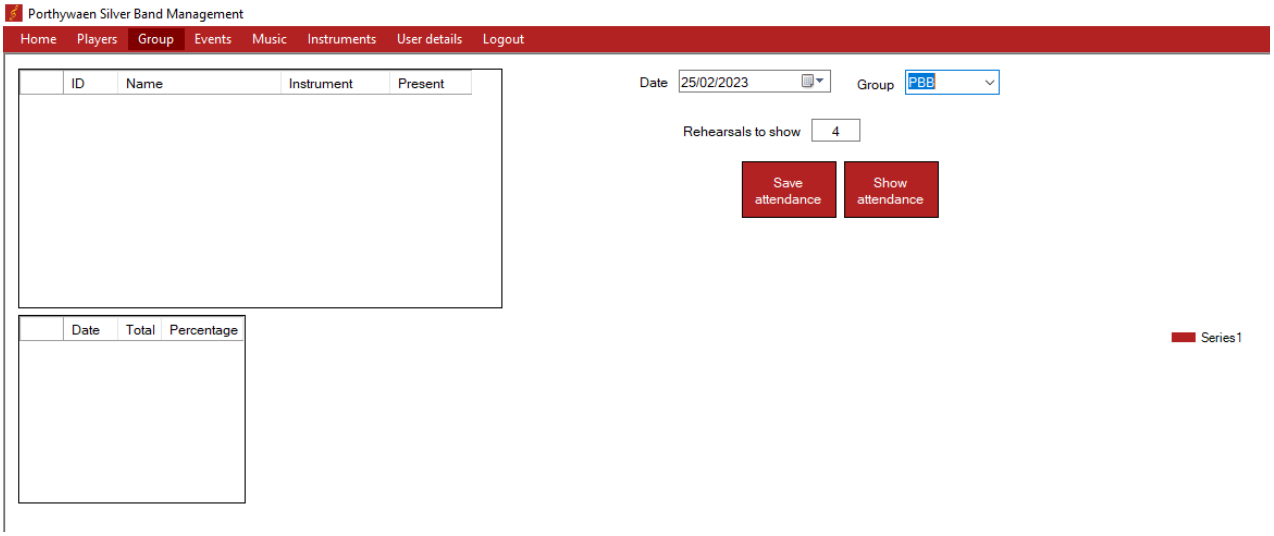

Figure 67: The "Group"screen when "PBB" is selected.

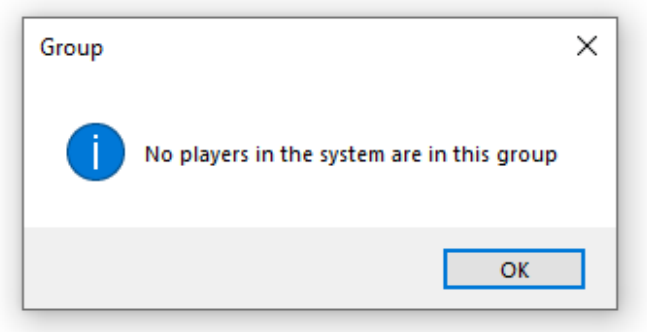

Figure 68: The message shown when a group is selected has no players.

Test 28

This test checks attendance can be recorded on a certain date. The user can mark attendance by selecting a group and selecting a date to record attendance for. They can then tick the checkboxes next to the players that are present and click “Save attendance”. For each event in the DataGridView in turn, the file is opened and searched for any marks for the group on the selected date. If any marks are found, they are split by player and stored in an array. If marks for the group and date are found, each array element is checked for the user's ID. If it is not, then all the responses are recombined from the array and the user's response is appended. The response is then saved in the file and a message is shown informing the user that the attendance has been saved.

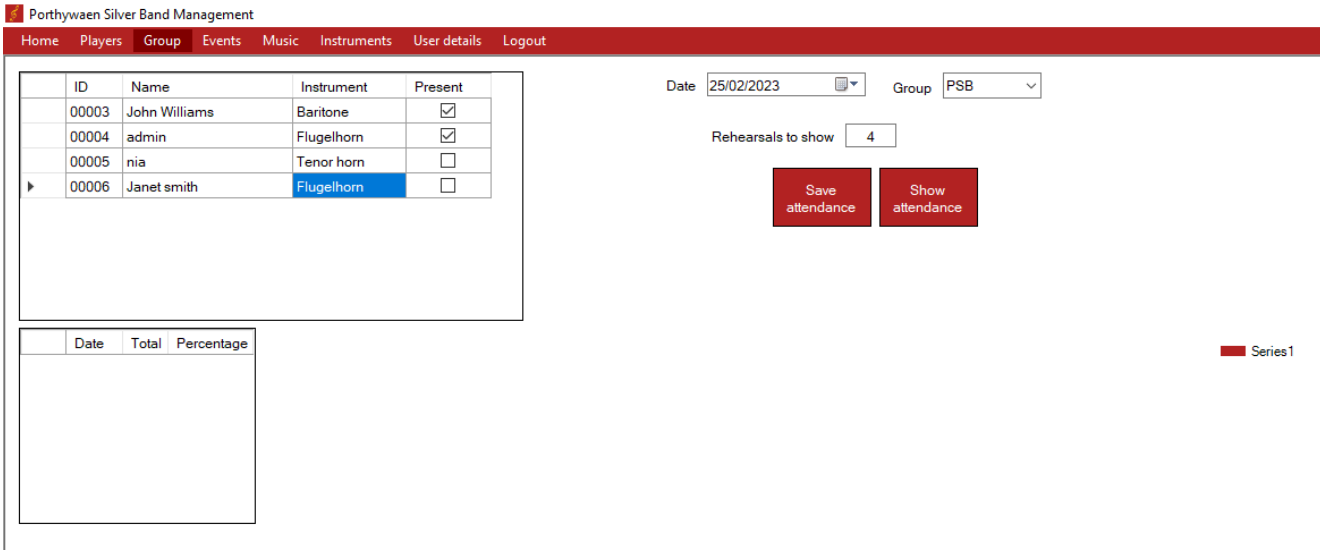

Figure 69: The “Group” screen with attendance inputted before “Save attendance” is clicked.

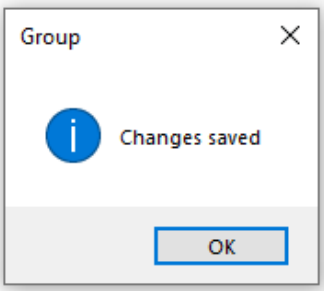

Figure 70: The message shown when “Save attendance” is clicked and the attendance has been recorded successfully.

Test 29

This test checks that attendance can be updated. For each record in the DataGridView in turn, the file is opened and searched for any marks for the group on the selected date. If any marks are found, they are split by player and stored in an array. If marks for the group and date are found, each array element is checked for the user's ID. If the user's ID is found, the response stored in the array is compared to the inputted mark. If they do not match, a message is shown, informing the user that a different mark has been recorded for this player and asking them if they want to update it. If "No" is clicked, no changes are made to the attendance. If "Yes" is clicked, the attendance is updated and a message is shown to confirm it.

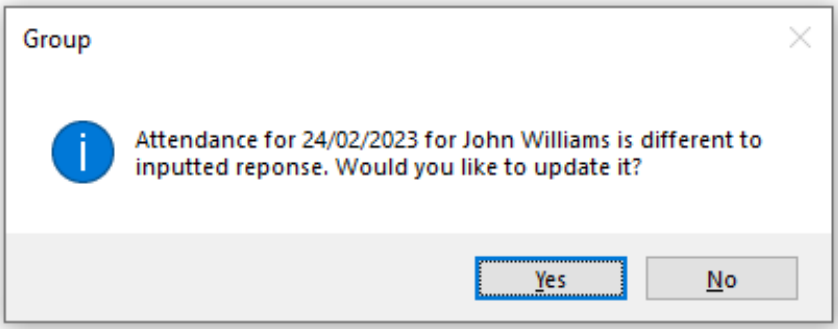

Figure 71: The message shown when a different mark has been recorded for a group on a date when "Save attendance" is clicked.

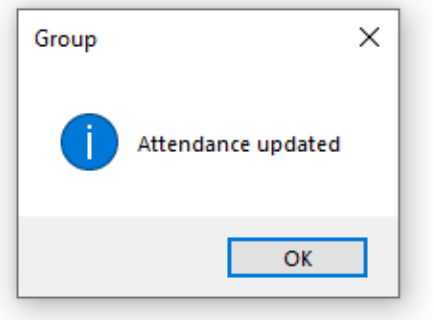

Figure 72: The message shown when "Yes" is clicked and the attendance has been updated.

Test 30

This test checks that attendance can be only recorded once for each player in a group. For each record in the DataGridView in turn, the file is opened and searched for any marks for the group on the date selected. If any marks are found, they are split by player and stored in an array. If marks are found, each array element is checked for the user's ID. If the user's ID is found, the mark stored in the array is compared to the inputted mark. If they match, a message is shown and no changes are made to the responses for the event.

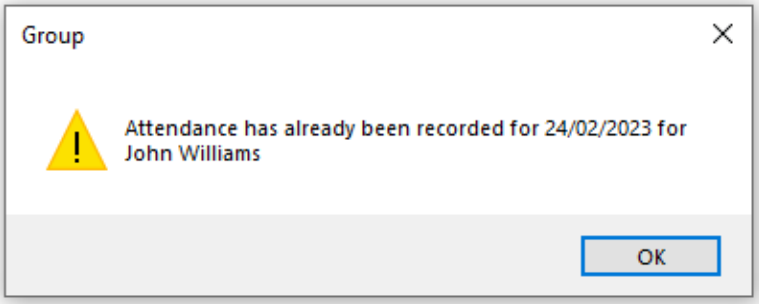

Figure 73: The message shown when the same mark has been recorded for a group on a date when "Save attendance" is clicked.

Test 31

This test checks that a group must be selected to mark and view attendance. The subroutine checks that there is data in the combobox. If no group is selected and "Save attendance" or “Show attendance" is clicked, a message will be shown informing the user that a group must be selected to save or show attendance.

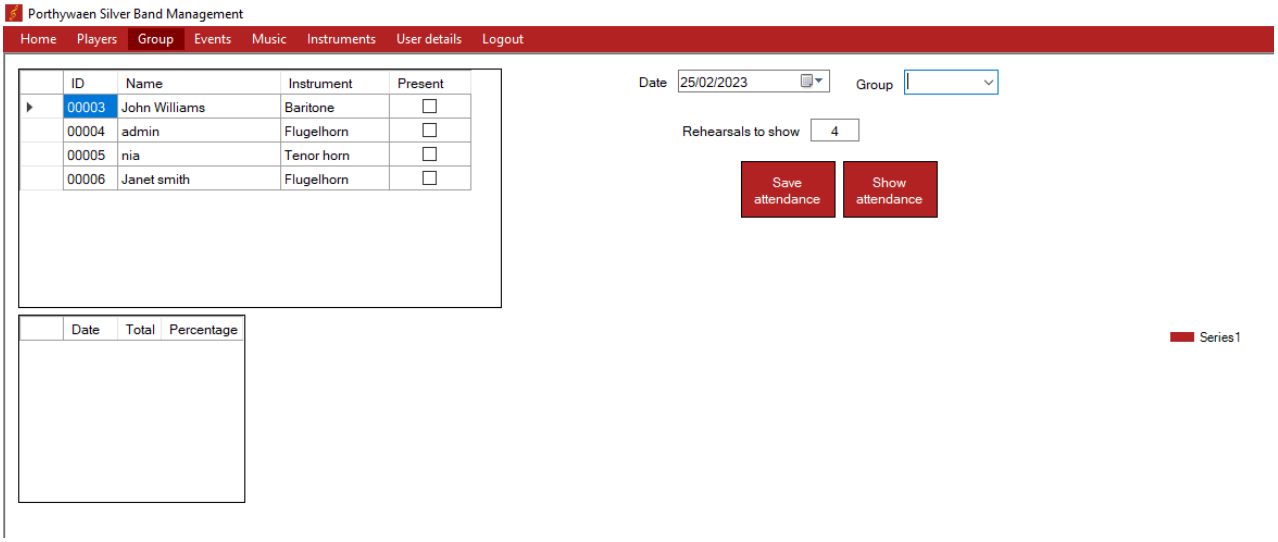

Figure 74: The “Group” screen before “Show attendance” or “Show attendance” is clicked, with no group selected.

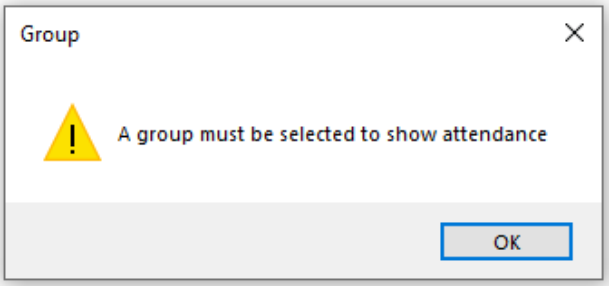

Figure 75: The message shown when “Show attendance” is clicked and no group has been selected.

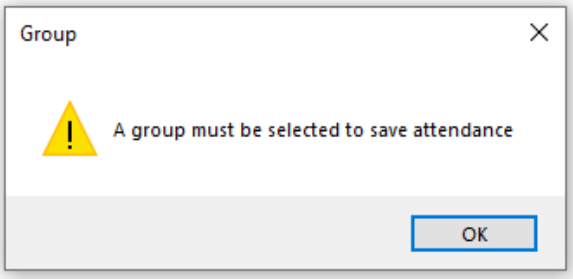

Figure 76: The message shown when “Save attendance” is clicked and no group has been selected.

Test 32

This test checks attendance can be viewed in a graph. When the user selects a group and enters the number of weeks to show, for each date, the marks are split and stored in an array. Each element in this array is read, and if the mark is True, the count is incremented by 1. The count, the total number of marks and the date are each stored in separate arrays, but at the same index. The arrays are sorted by date in descending order so the most recent dates are shown if only a few dates are selected to be viewed.

The number of dates to show in the chart is then determined by comparing the inputted number of dates to the length of the dates array. If the inputted value is less than the array length, then this is the number of dates to show. If not, then the number of dates shown equals the length of the array -1.

The percentage attendance is now calculated by dividing each count by the total read value at each index in turn up to the number of dates to show. Each percentage, the date, and the count are added to the DataGridView. Once the correct number of dates have been added to the DataGridView, the DataGridView is sorted in ascending order by date so the data in the chart will be in ascending order. Then the date and percentage from each row in the DataGridView are added to the chart.

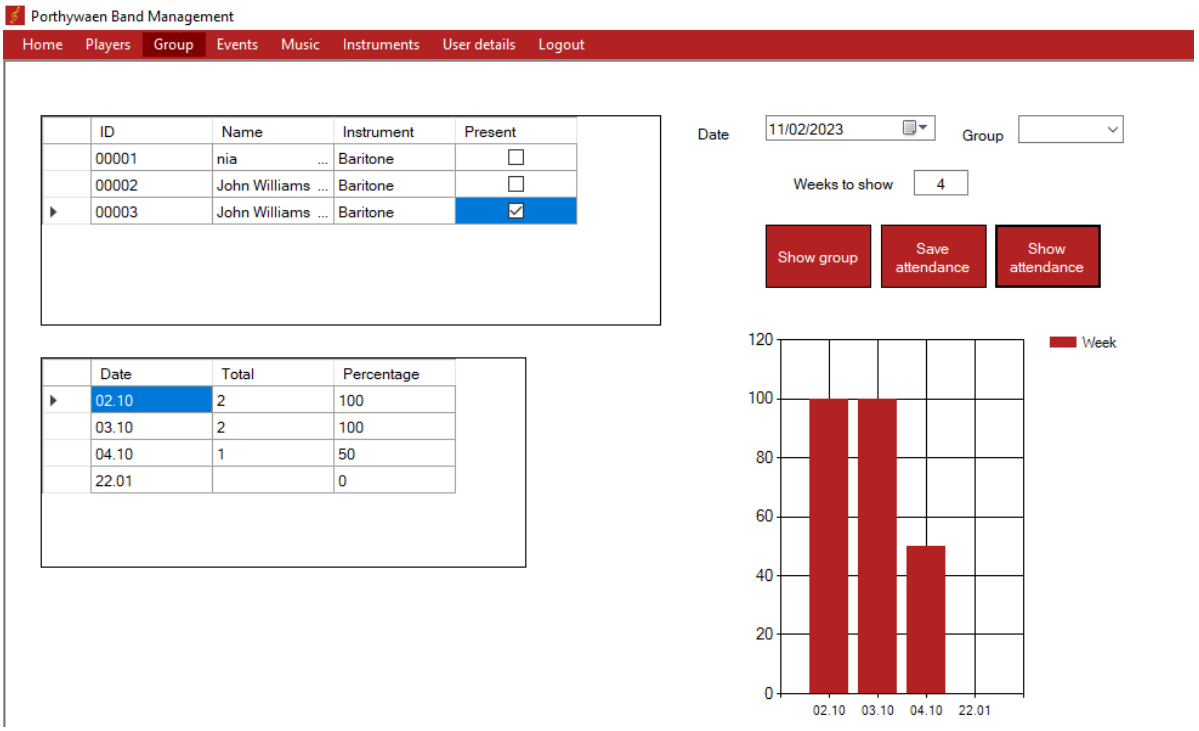

Figure 77: The "Group" screen after "Show attendance" is clicked and the attendance has been shown successfully.

When completing this test, I realised that the graph could be too small if many dates were to be displayed. To fix this, I changed the size of the chart and the attendance DataGridView. I also added axis labels to the graph to make it clearer to the user what the graph shows as the user may not know exactly what the graph shows.

```
chtAttendance.ChartAreas("ChartArea1").AxisX.Title = "Dates"
chtAttendance.ChartAreas("ChartArea1").AxisY.Title = "Percentage present"
```

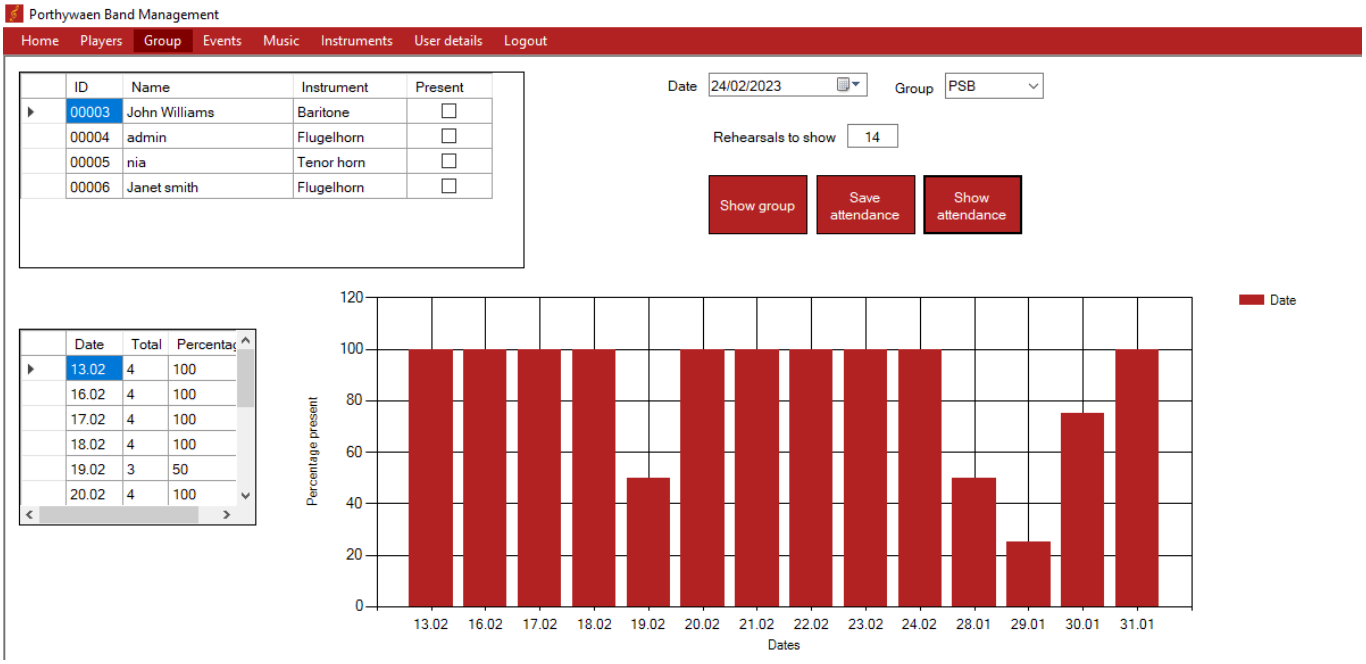

Figure 78: The "Group" screen after "Show attendance" is clicked and the attendance has been shown successfully.

Players and User details

Test 33

This test checks that a player can be added to the database. When the player's details are entered into the input fields and "Add" is clicked, a subroutine is called which processes the inputted data. The data is validated and the player ID is generated by searching the file for a possible ID and incrementing the ID by 1 each time an ID is found. The final ID is padded with zeros to ensure it is the same length as all the other IDs. The data is appended to the file and the DataGridView is refreshed to show the changes. A message is shown informing the user that the player has been added.

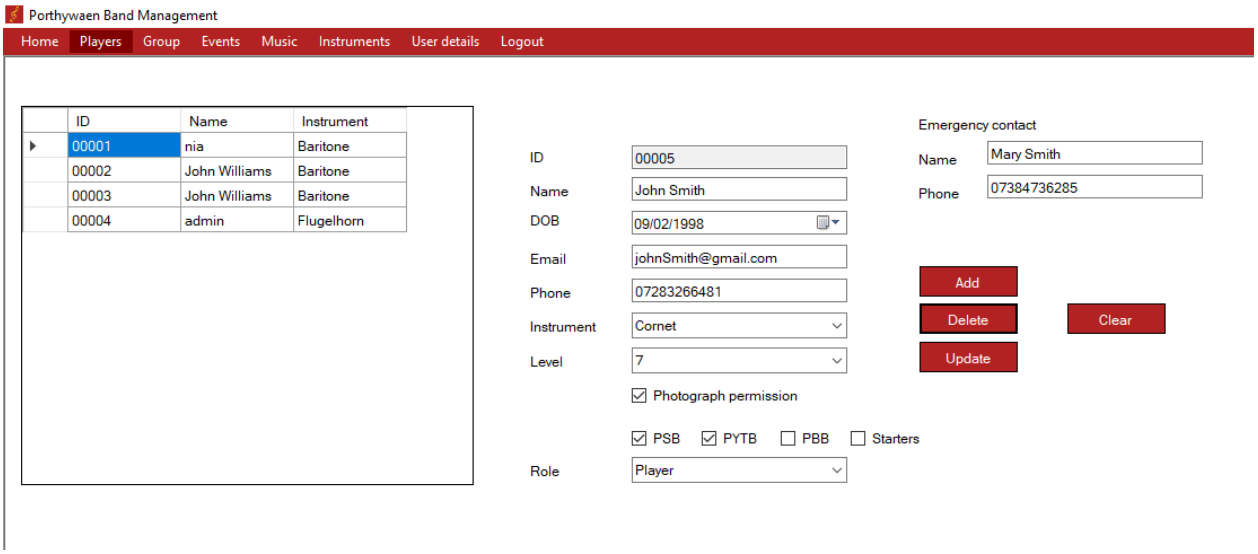

Figure 79: The "Players" screen with a new player's details inputted before "Add" is clicked.

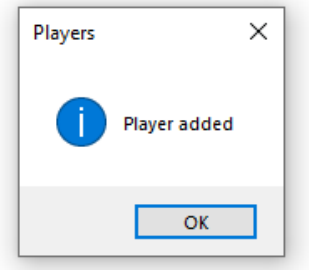

Figure 80: The message shown when "Add" is clicked and the player has been added successfully.

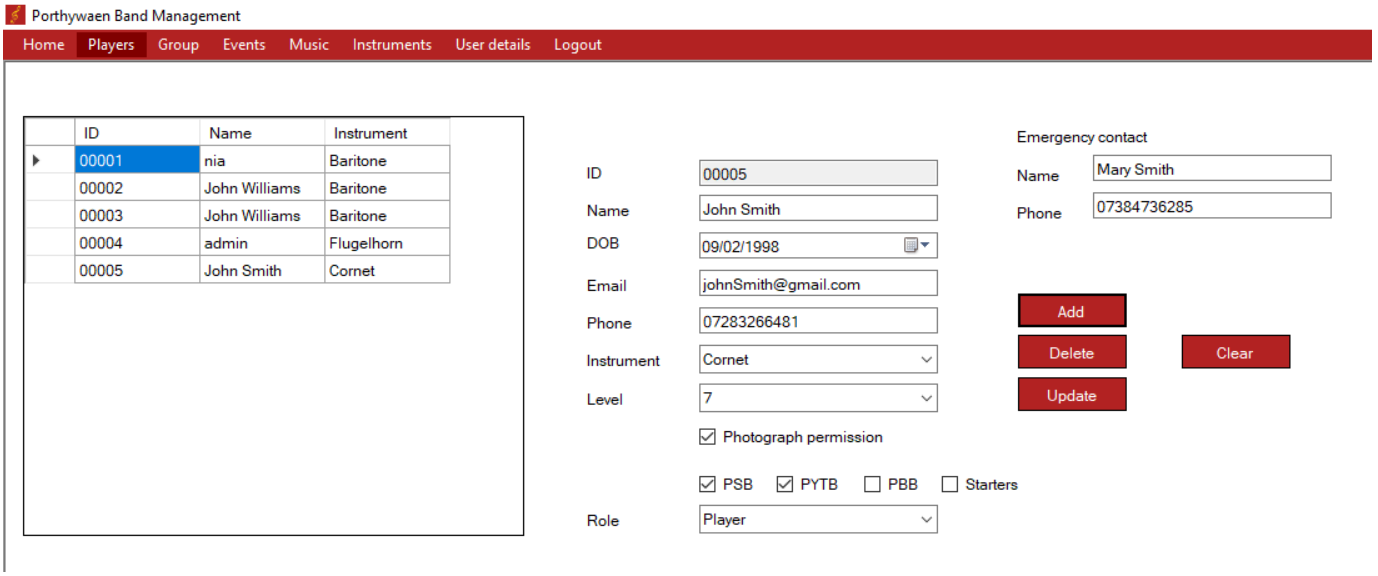

Figure 81: The "Players" screen after a player has been added.

Test 34

This test checks that a player's details can be updated. When a player is selected, the user can change any of the details and click "Update" . The data inputted into the system is validated and then stored in a structure. The record number of the record to be updated is determined from the DataGridView cell clicked and that record is updated in the file. A message is shown and the DataGridView is refreshed to show the changes.

When “Update” is clicked in “Players”:

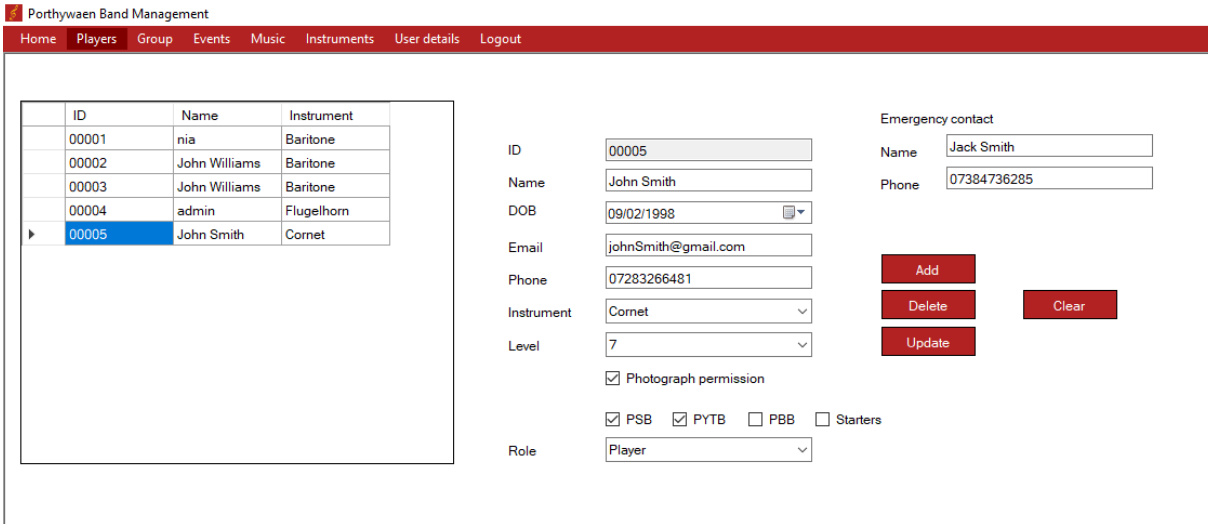

Figure 82: The “Players” screen while a player’s details are being updated.

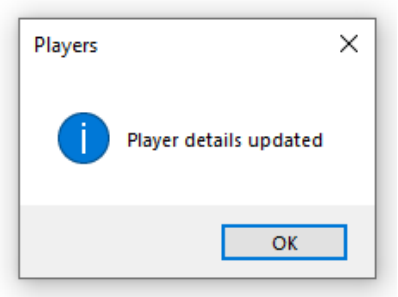

Figure 83: The message shown after “Update” is clicked and the player’s details have been successfully updated.

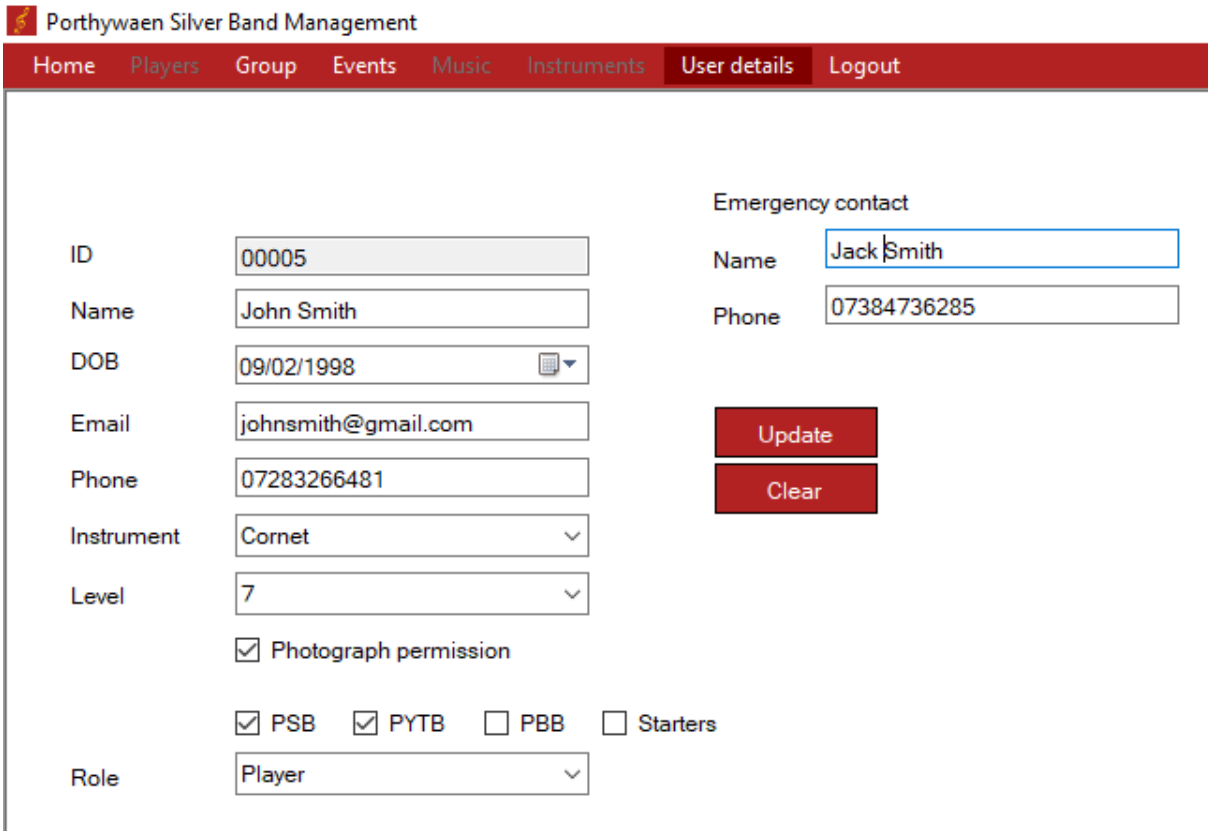

Figure 84: The "User details" screen while a player's details are being updated.

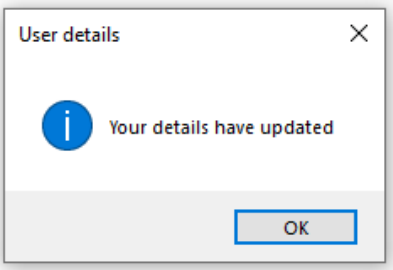

Figure 85: The message shown after "Update" is clicked and the player's details have been successfully updated.

Test 35

This test checks that the player's name is required. If the user leaves the name textbox empty and clicks "Add" or "Update" , a message is shown informing the user that the player's name must be entered.

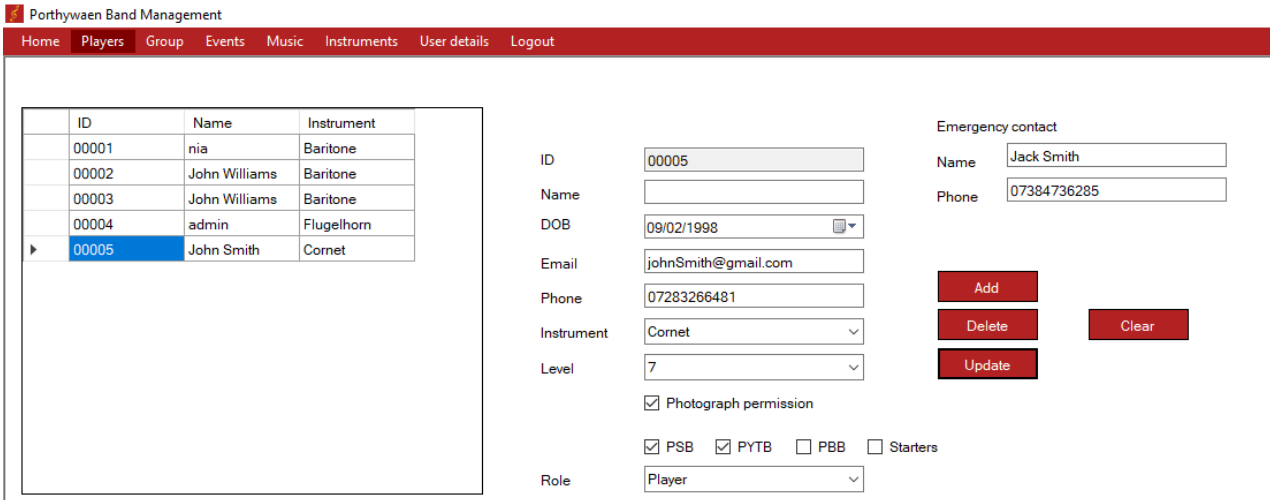

Figure 86: The "Players" screen with the name field left empty before "Add" or "Update" is clicked.

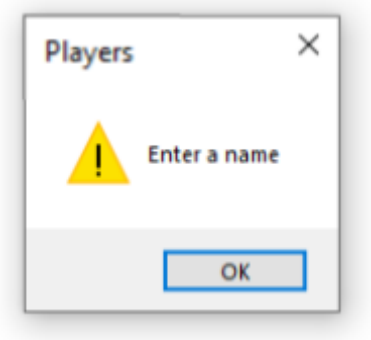

Figure 87: The message shown when the name field is left empty after "Add" or "Update" is clicked.

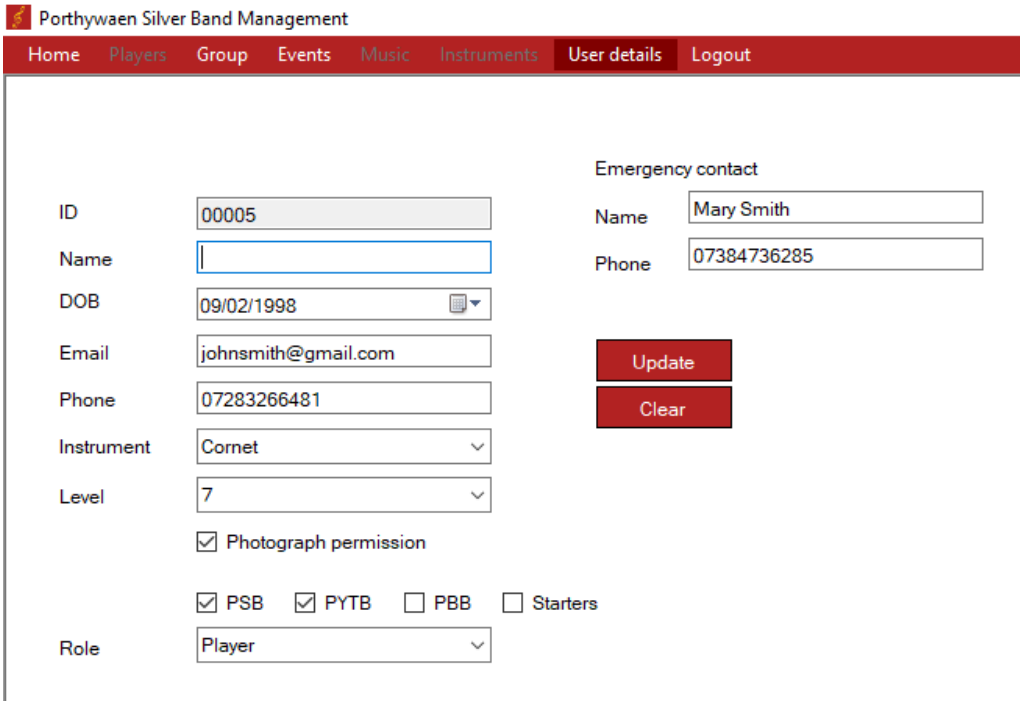

Figure 88: The "User details" screen with the name field left empty before "Update" is clicked.

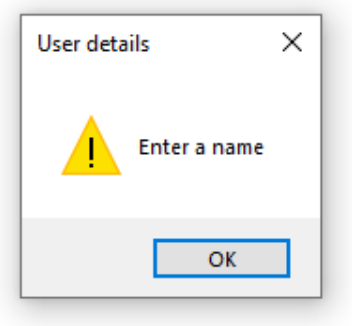

Figure 89: The message shown when the name field is left empty after "Update" is clicked.

Test 36

This test checks that the player's email is in the correct format. If the user enters data that is not in email format, a message is shown informing them that they have not entered a valid email address.

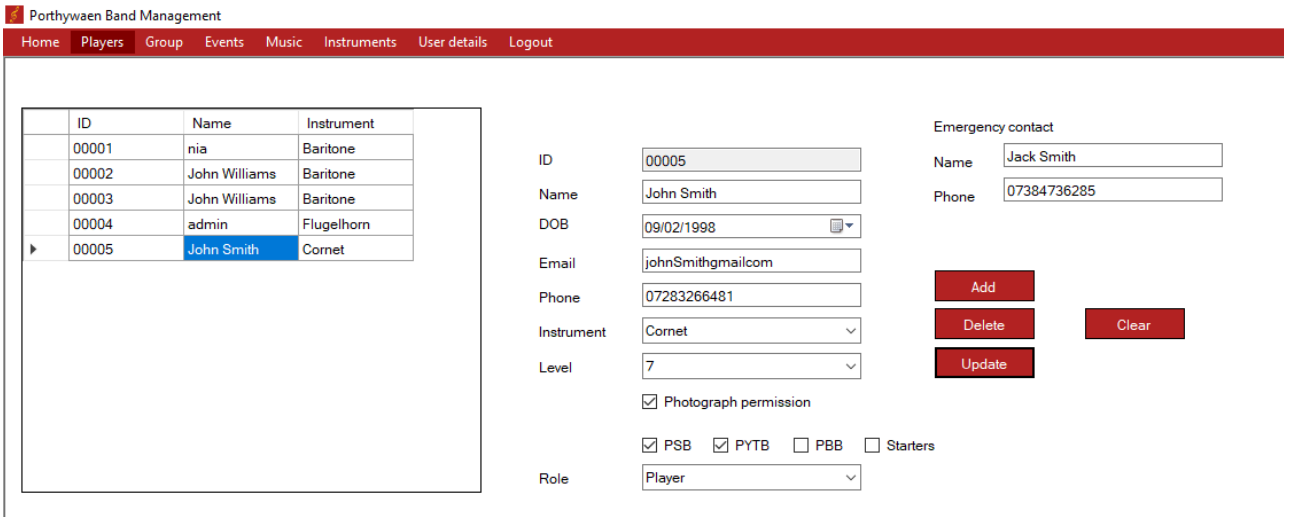

Figure 90: The "Players" screen with the email in an invalid format before "Add" or "Update" is clicked.

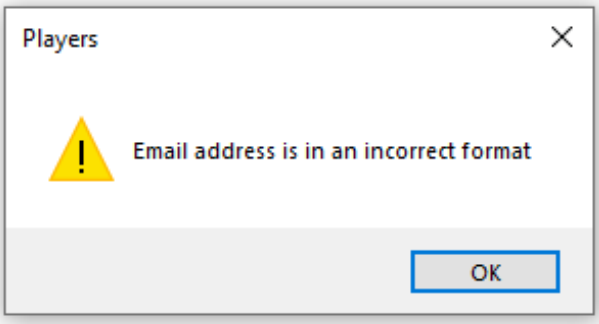

Figure 91: The message shown when the email in an invalid format after "Add" or "Update" is clicked.

In “User details”:

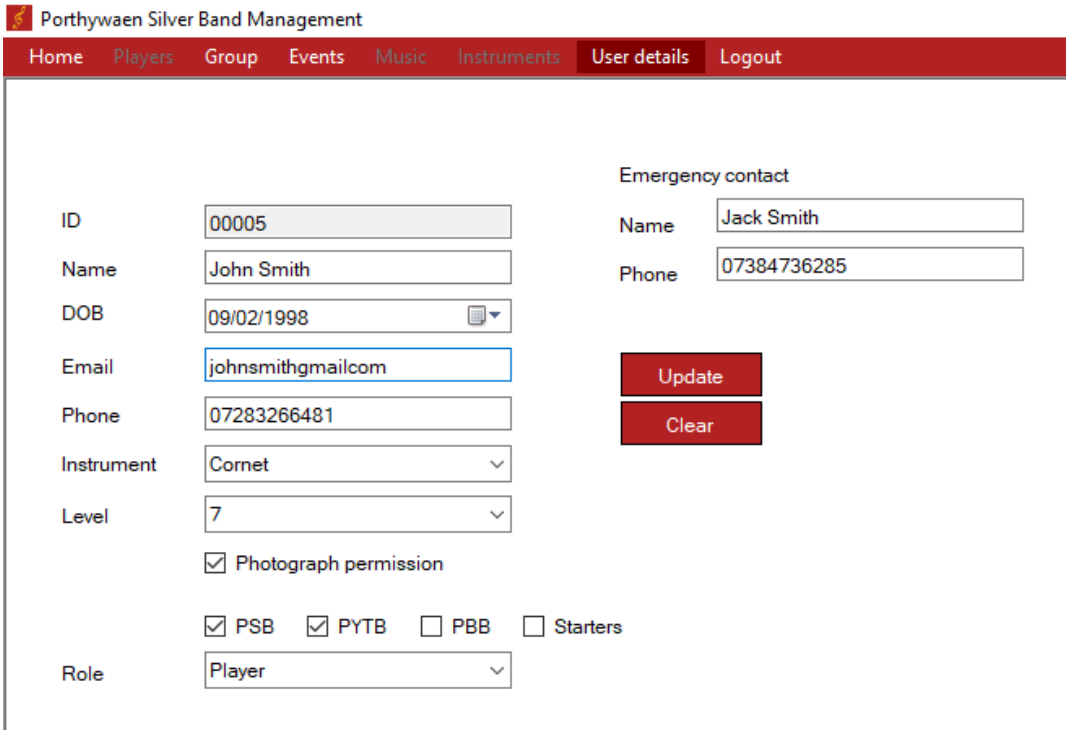

Figure 92: The “User details” screen with the email in an invalid format before “Update” is clicked.

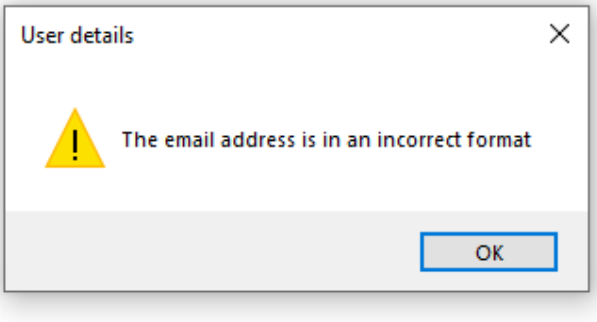

Figure 93: The message shown when the email in an invalid format after “Update” is clicked.

Test 37

This test checks if the player's DOB is within an acceptable range. If the user entered a date before the year 1900 or after the current date, a message is shown informing the user that the data is invalid.

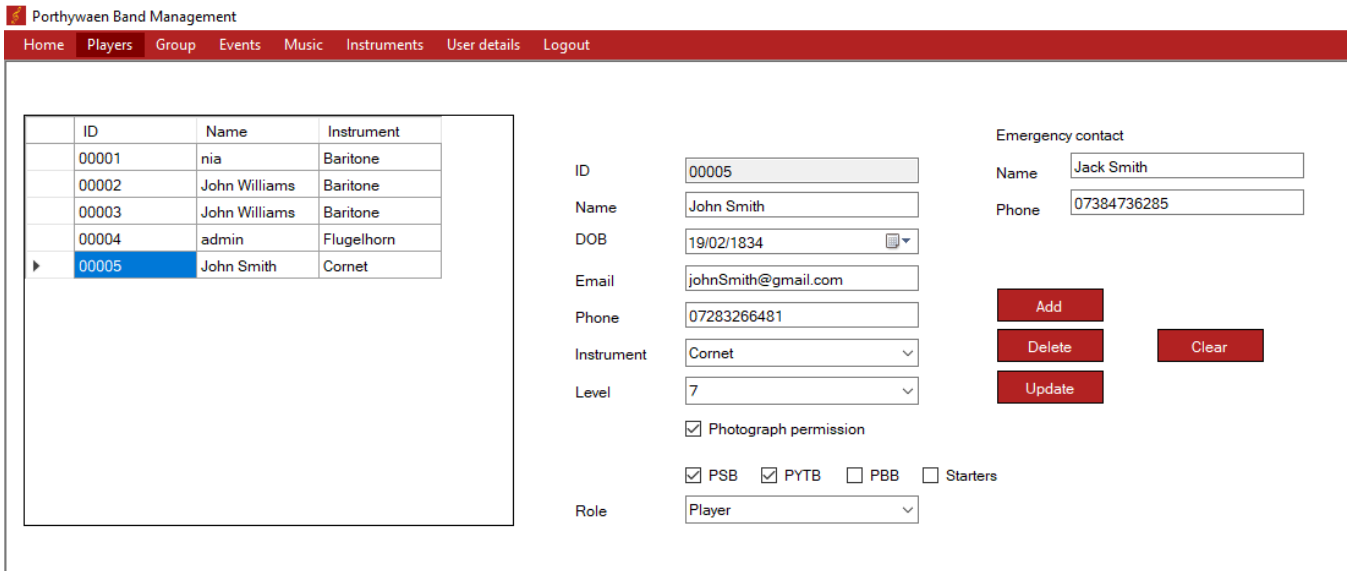

Figure 94: The "Players" screen with an invalid DOB inputted before "Add" or "Update" is clicked.

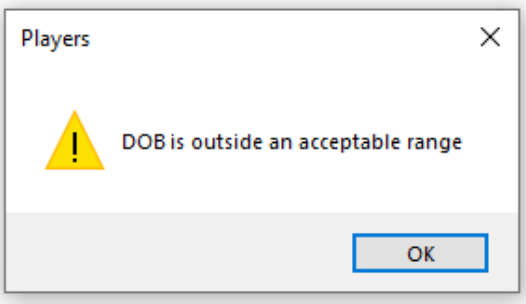

Figure 95: The message shown when an invalid DOB is inputted and "Add" or "Update" is clicked.

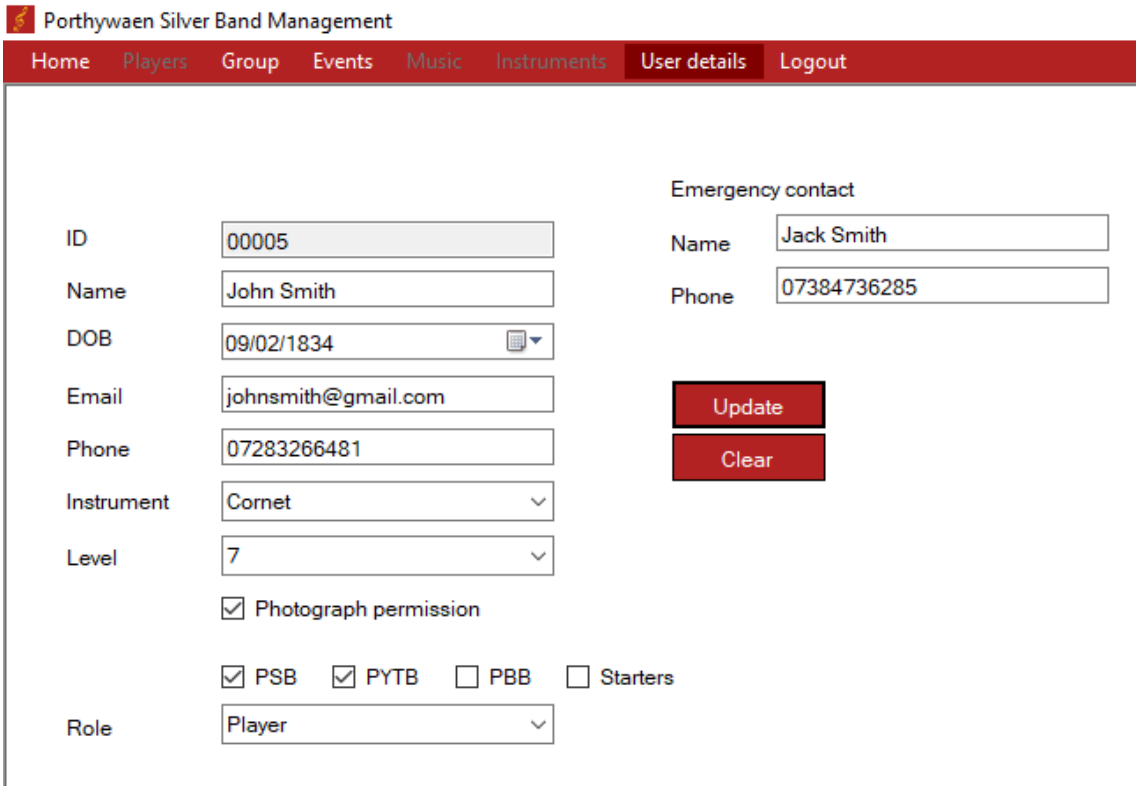

Figure 96: The "User details" screen with an invalid DOB inputted before "Update" is clicked.

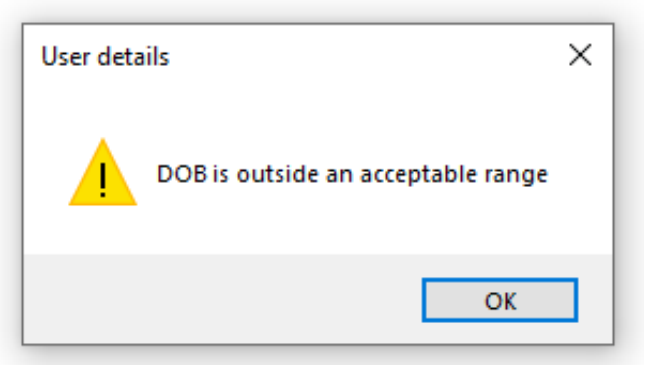

Figure 97: The message shown when an invalid DOB is inputted after "Update" is clicked.

Test 38

This test checks that the emergency contact phone number is 11 characters long. When the user enters a phone number that is 12 characters long and clicks "Add" or "Update" , the length of the input is checked to see if it is exactly 11 characters long. If it is not, a message is shown informing the user that the phone number entered is invalid.

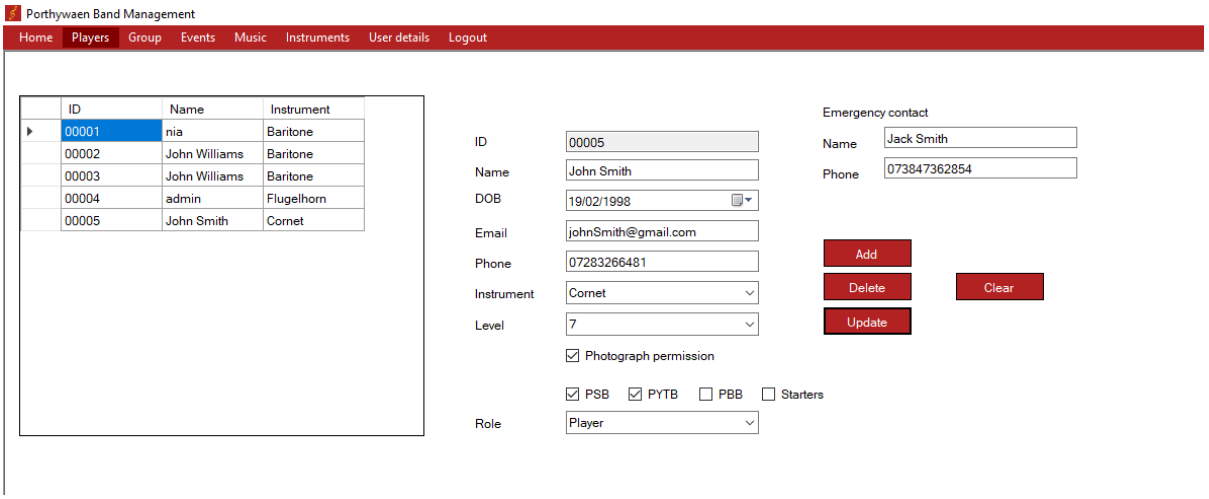

Figure 98: The "Players" screen with an invalid contact phone number inputted.

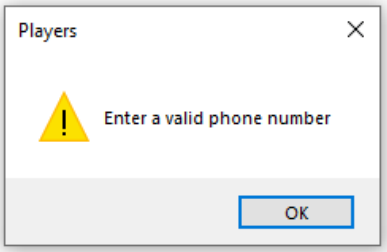

Figure 99: The message shown when an invalid contact phone number is inputted and "Add" or "Update" is clicked.

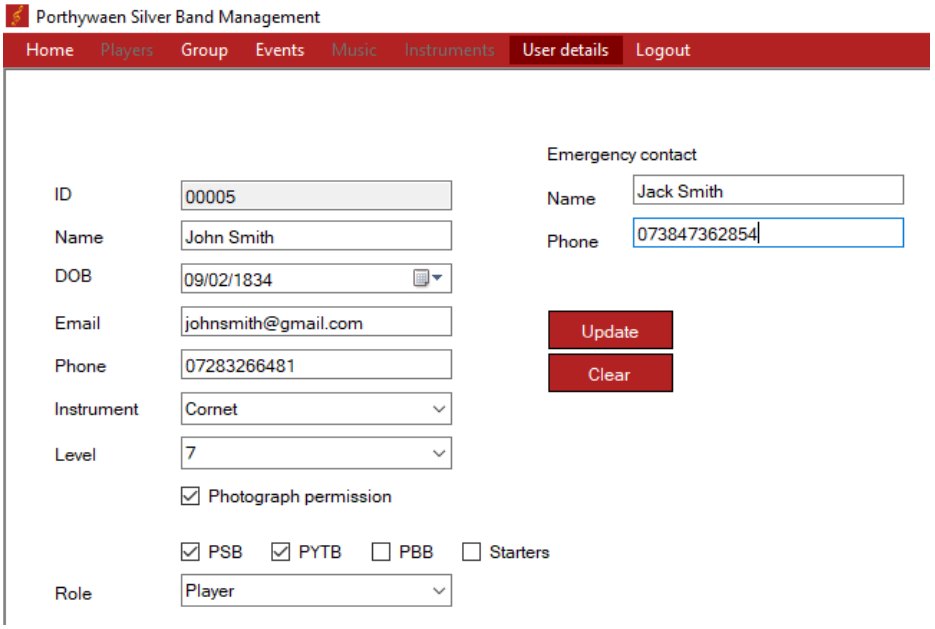

Figure 100: The "User details" screen with an invalid contact phone number inputted.

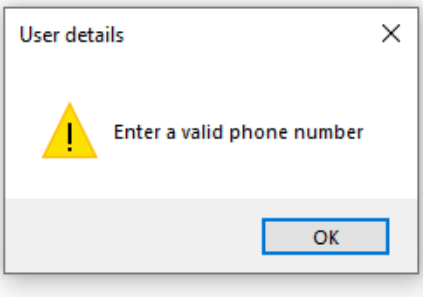

Figure 101: The message shown when an invalid contact number is inputted and "Update" is clicked.

Test 39

This test checks that at least one group must be selected. When data is entered into all fields except the groups checkboxes and "Add" or "Update" is clicked, a message is displayed informing the user that at least one group must be selected.

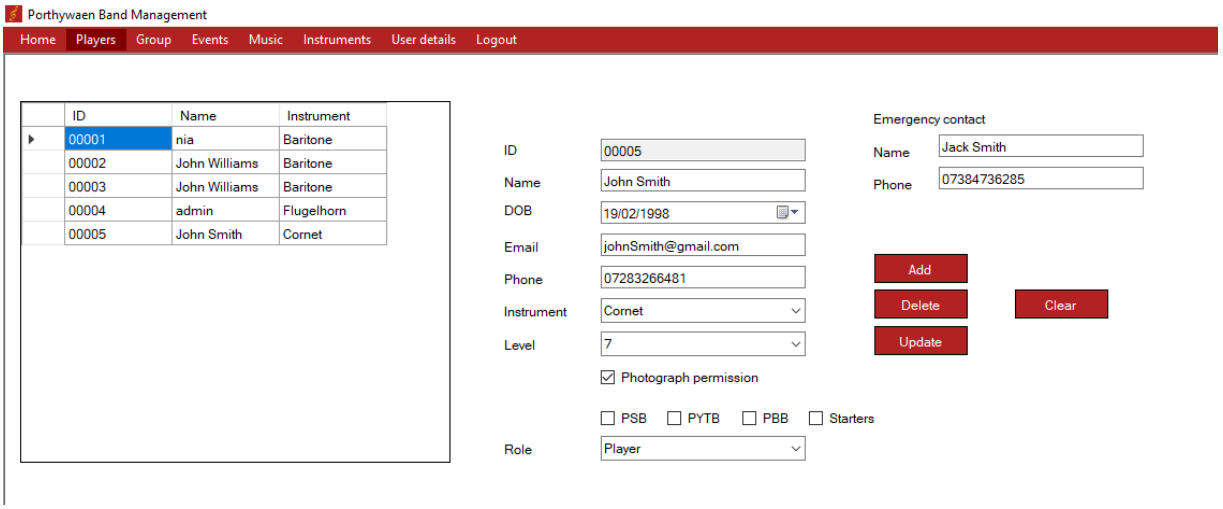

Figure 102: The "Players" screen with no groups ticked.

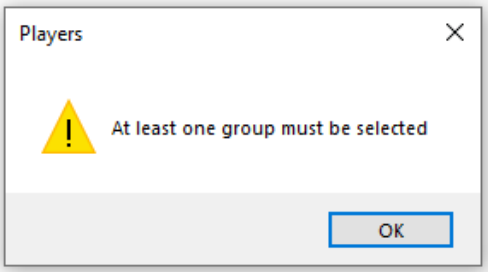

Figure 103: The message shown when no groups are ticked and "Add" or "Update" is clicked.

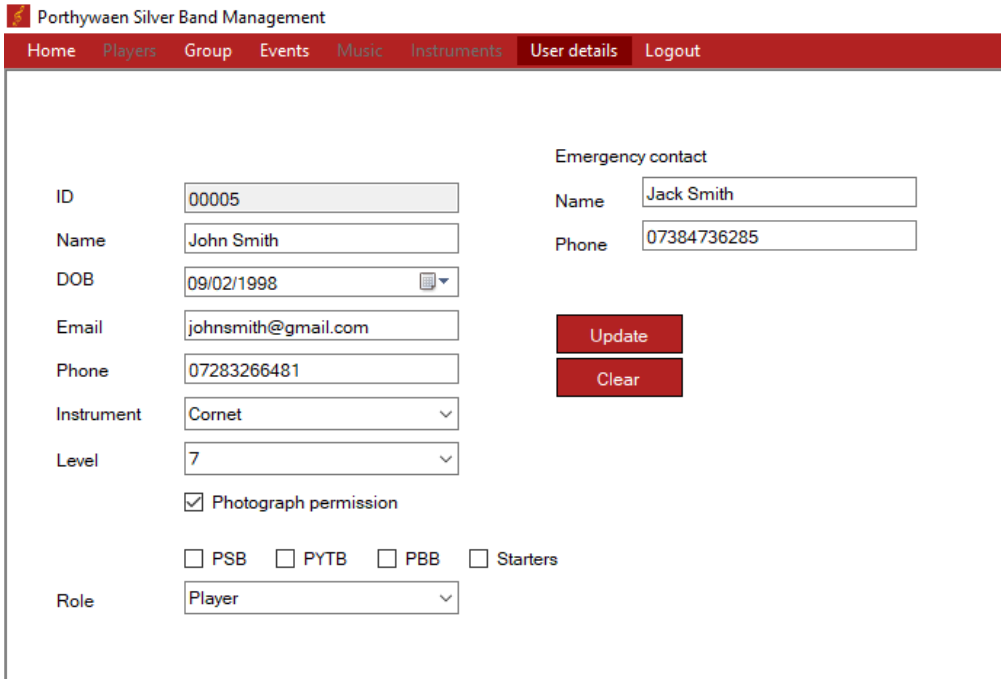

Figure 104: The "User details" screen with no groups ticked.

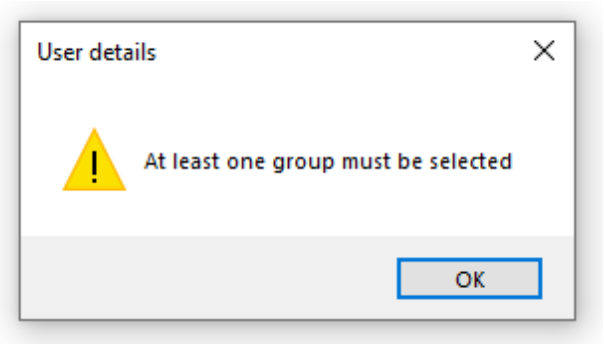

Figure 105: The message shown when no groups are ticked and "Update" is clicked.

Test 40

This test checks that a player can be deleted from the database. When a player is selected in the DataGridView and "Delete" is clicked, each record in the file is compared to the record that has been selected to be deleted. If it is not this record, then it is added to a temp file. When the record that is to be deleted is found, the record is skipped and not added to the temp file. When the end of the file is reached, the temp file is renamed and the old file is deleted. A message is shown confirming the player has been deleted and the DataGridView is refreshed to show that the player has been removed from the database.

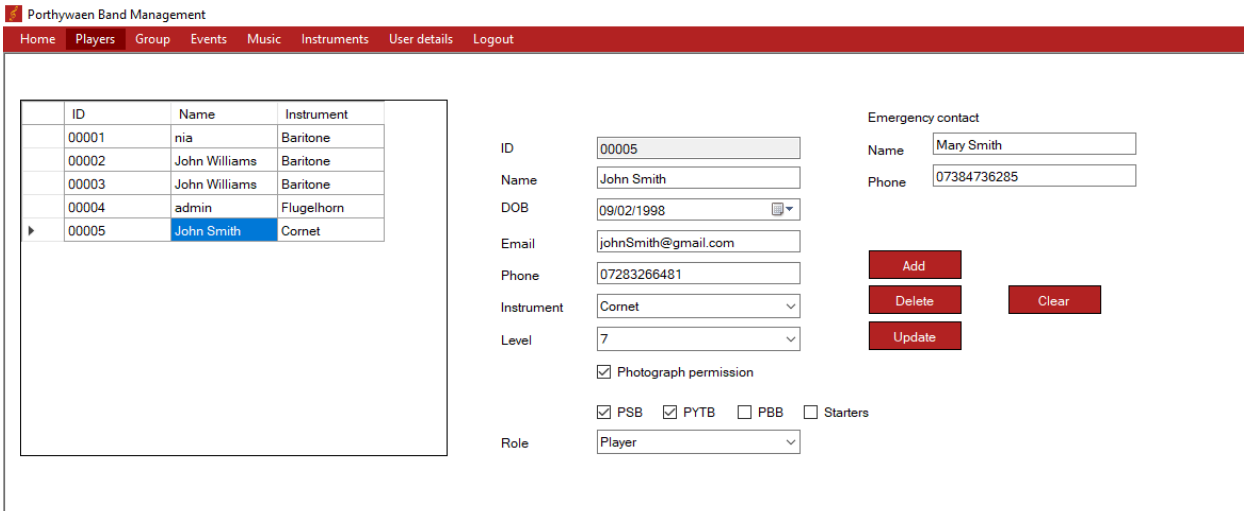

Figure 106: The "Players" screen with a player selected before "Delete" is clicked.

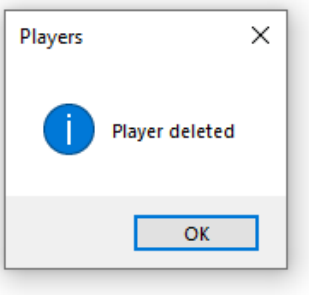

Figure 107: The message shown when a player is successfully deleted.

Music

Test 41

This test checks that a piece can be added to the database. When the user enters data into the fields on the screen and clicks add, a subroutine is called which processes the inputted data. The data is stored in a structure and validated. The music ID is generated by searching the file for a possible ID and incrementing the ID by 1 each time an ID is found. The final ID is padded with zeros to ensure it is the same length as all the other IDs. The data in the structure is appended to the file and the DataGridView is refreshed to show the changes. A message is shown to confirm the music has been added.

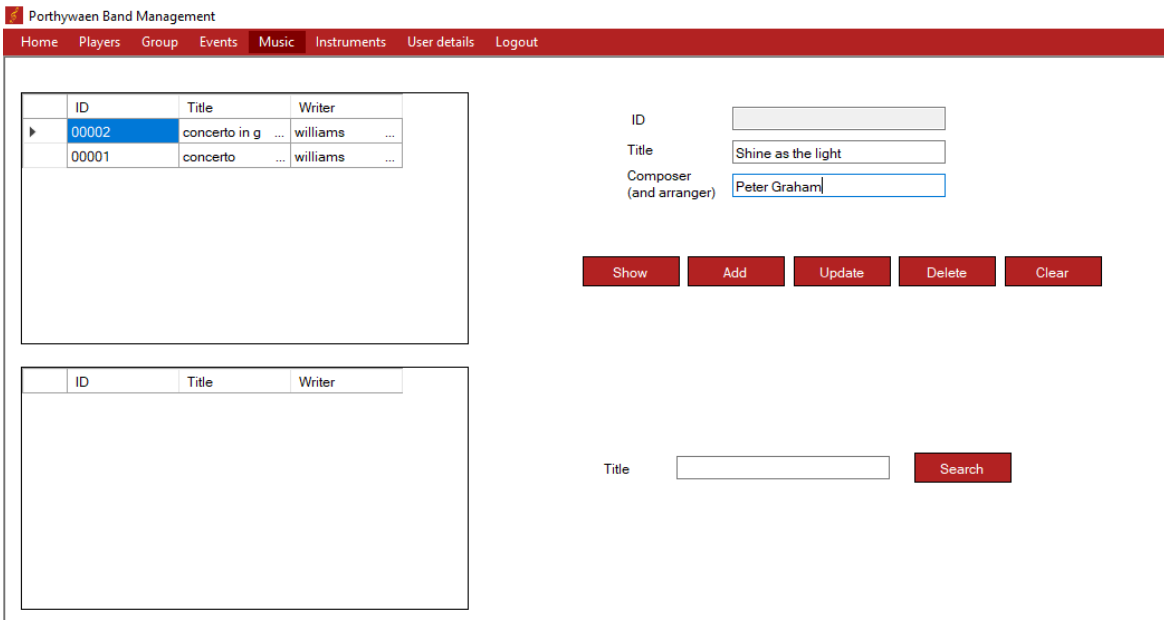

Figure 108: The "Music" screen with the details of a new piece inputted before "Add" is clicked.

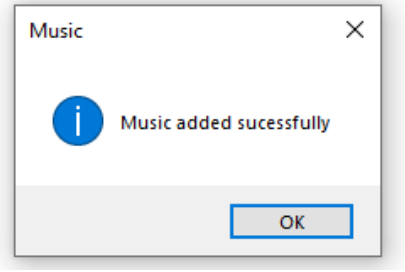

Figure 109: The message shown when "Add" is clicked and the music has been added successfully.

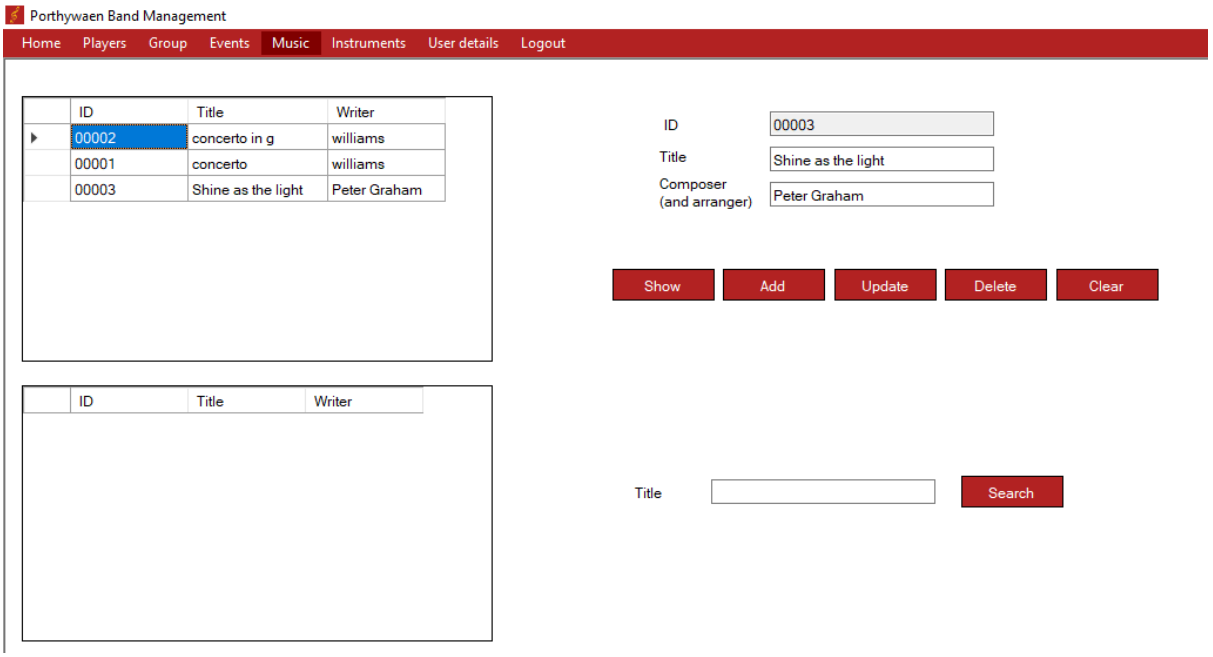

Figure 110: The "Music" screen after "Add" is clicked.

Test 42

This test checks that music can be updated. When a piece is selected, the user can change any of the details and click "Update" . The data inputted into the system is validated and then stored in a structure. The record number of the record to be updated is determined from the DataGridView cell clicked and that record is updated in the file. A message is shown and the DataGridView is refreshed to show the changes.

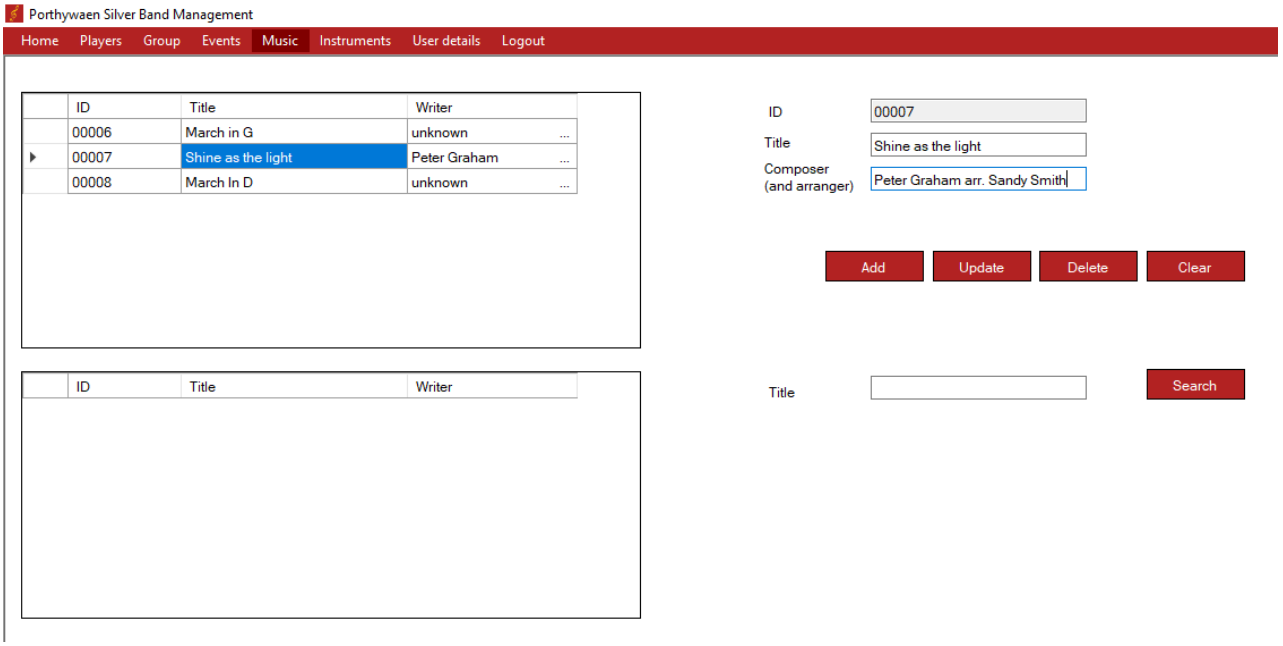

Figure 111: The "Music" screen while a piece is being updated.

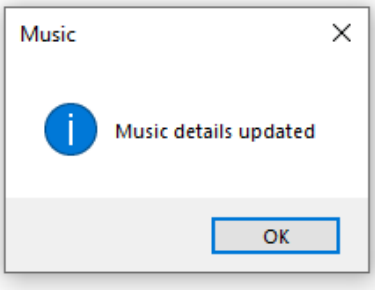

Figure 112: The message shown after "Update" is clicked and the music is successfully updated.

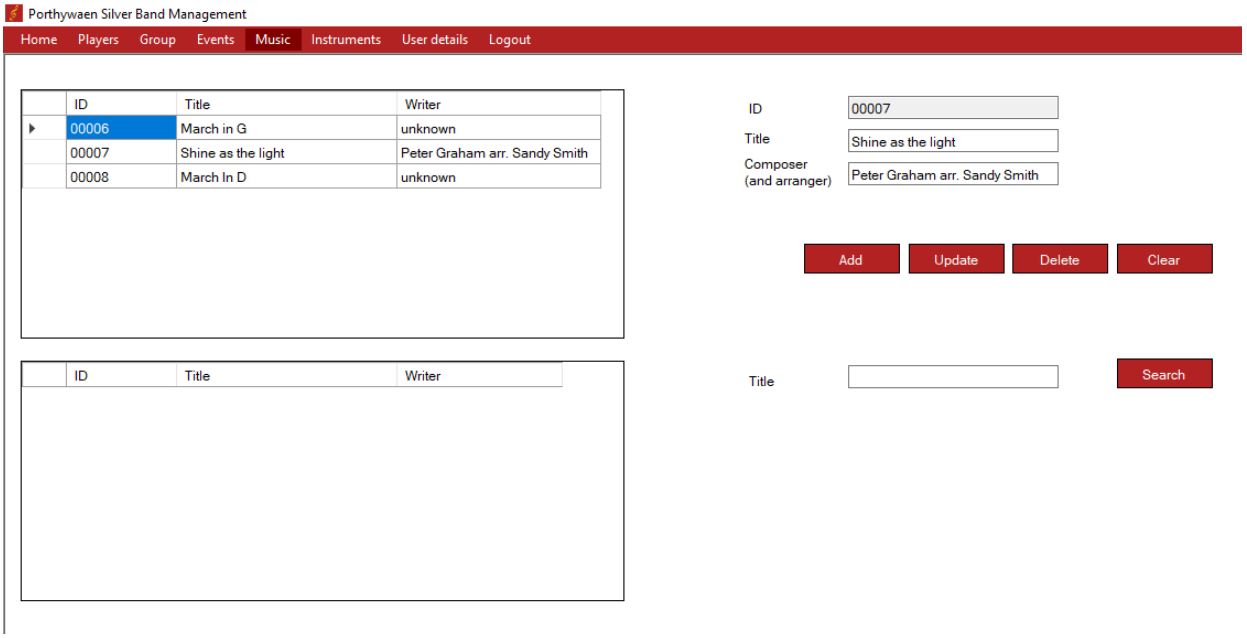

Figure 113: The "Music" screen after "Update" Is clicked and a piece has been updated successfully.

Test 43

This test checks that title is needed. When the user enters data in all input fields except the title and clicks "Add" or “Update”, a message is shown, informing the user that they need to enter the title of the piece.

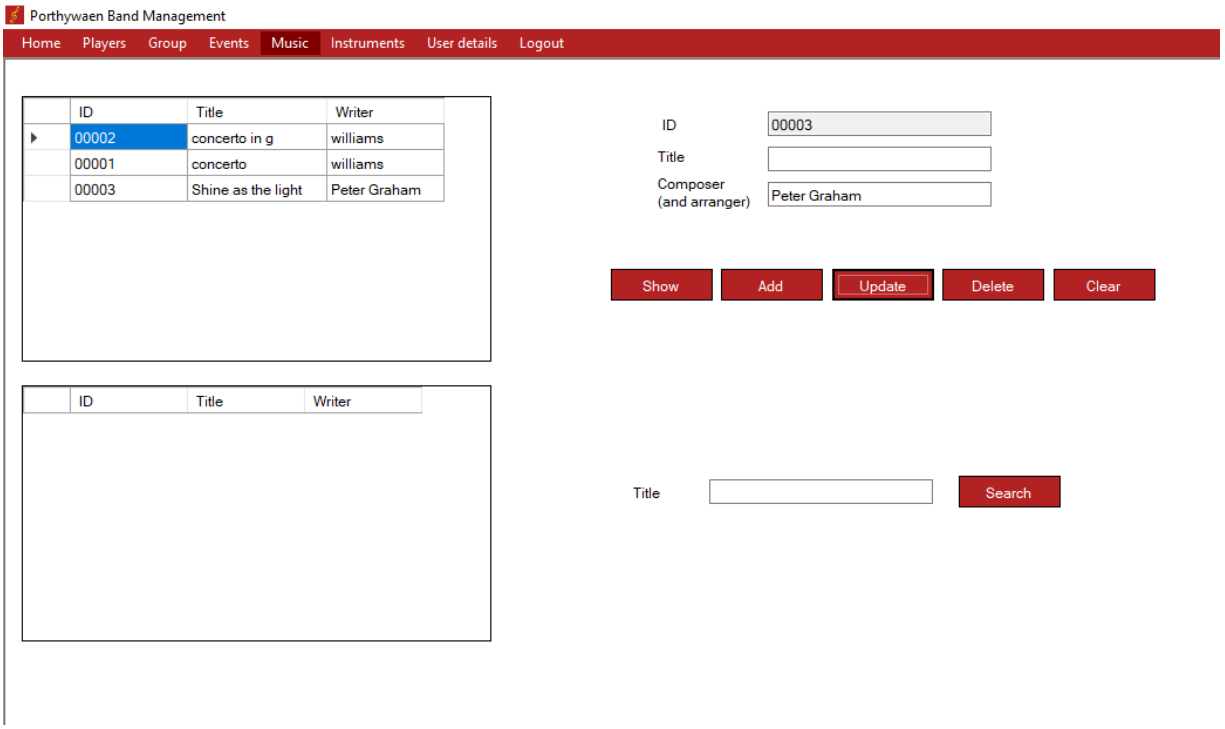

Figure 114: The “Music” screen with the title field empty before “Add” or “Update” is clicked.

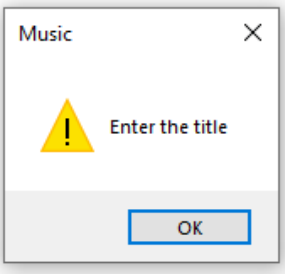

Figure 115: The message shown when the title field is empty and “Add” or “Update” is clicked.

Test 44

This test checks that a piece can be removed. When a piece is selected in the DataGridView and the user clicks "Delete" , each record in the file is compared to the record that has been selected to be deleted. If it is not this record, then it is added to a temp file. When the record that is to be deleted is found, the record is skipped and not added to the temp file. When the end of the file is reached, the temp file is renamed and the old file is deleted. A message is shown confirming it has been deleted and the DataGridView is refreshed to show that the piece is no longer in the database.

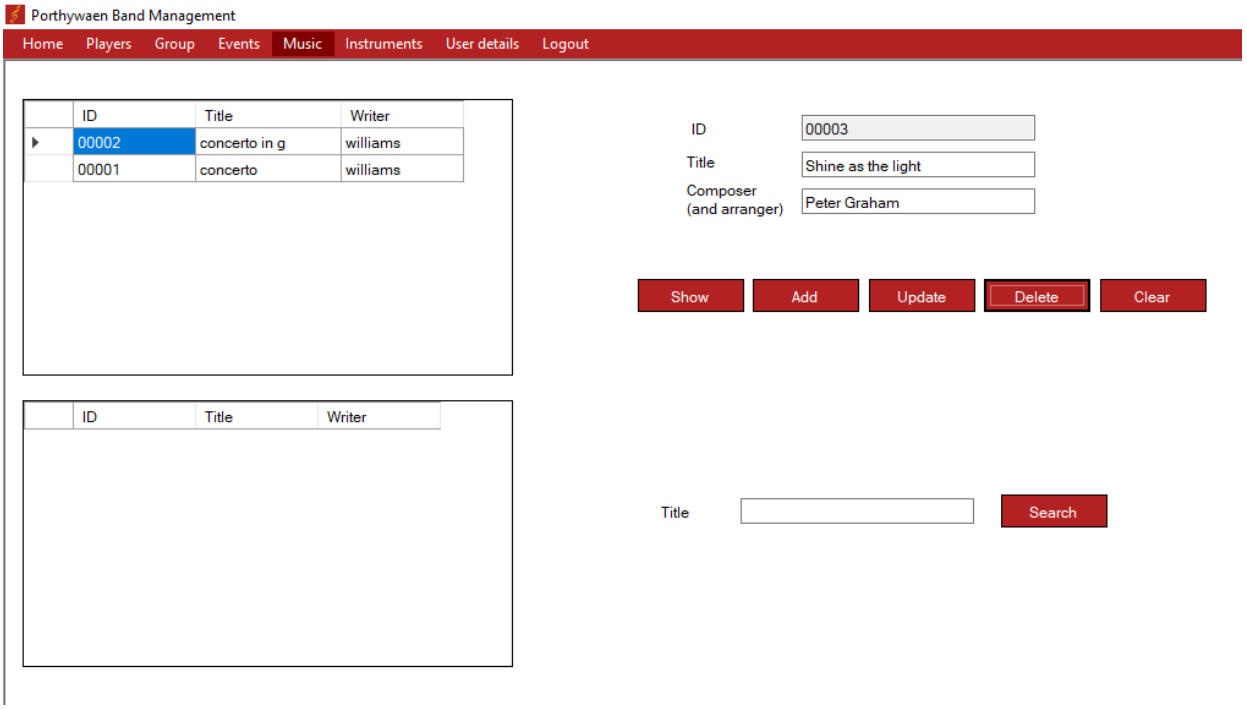

Figure 116: "Music" screen when a piece has been selected before it is deleted.

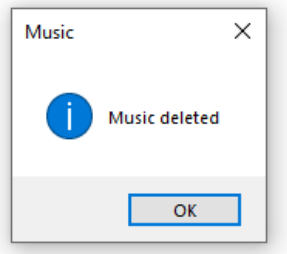

Figure 117: The message shown when a piece is deleted.

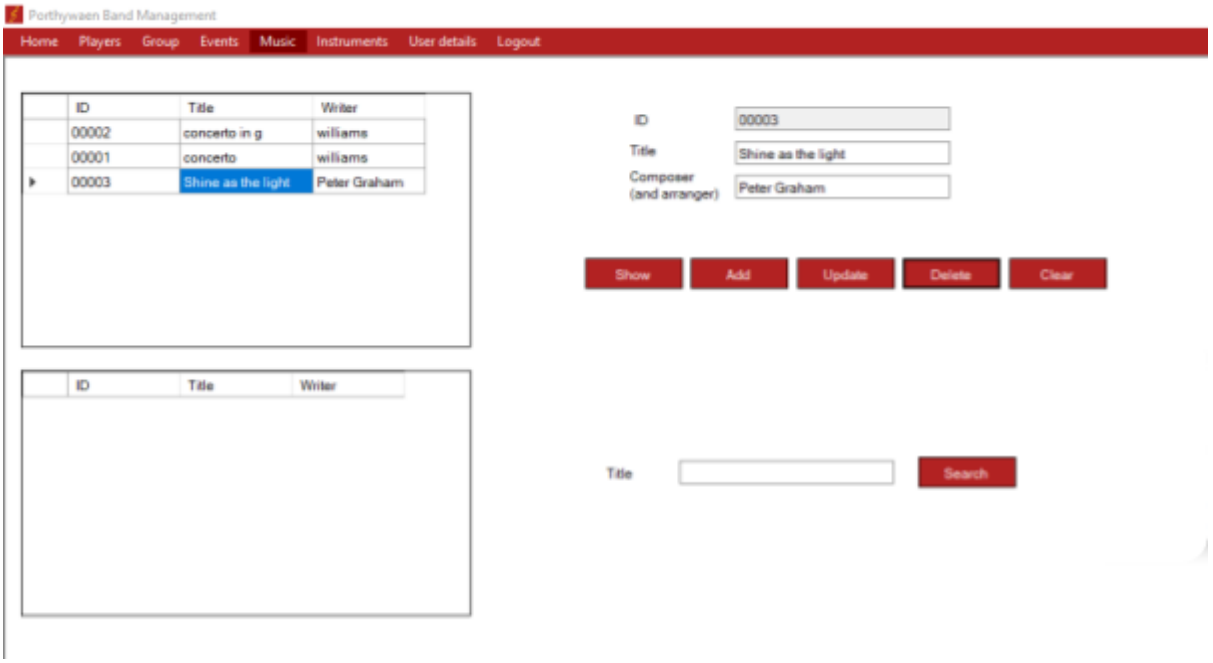

Figure 118: "Music" screen after a piece has been deleted.

Test 45

This test checks that a piece can be searched for. When the user inputs a title or keyword(s) into the box and clicks “Search”, the file is opened and each record is stored into a structure. If the title element in the structure contains the data inputted, the record is added to the DataGridView. If the end of the file is reached and no records are found that contain the inputted data, a message is shown.

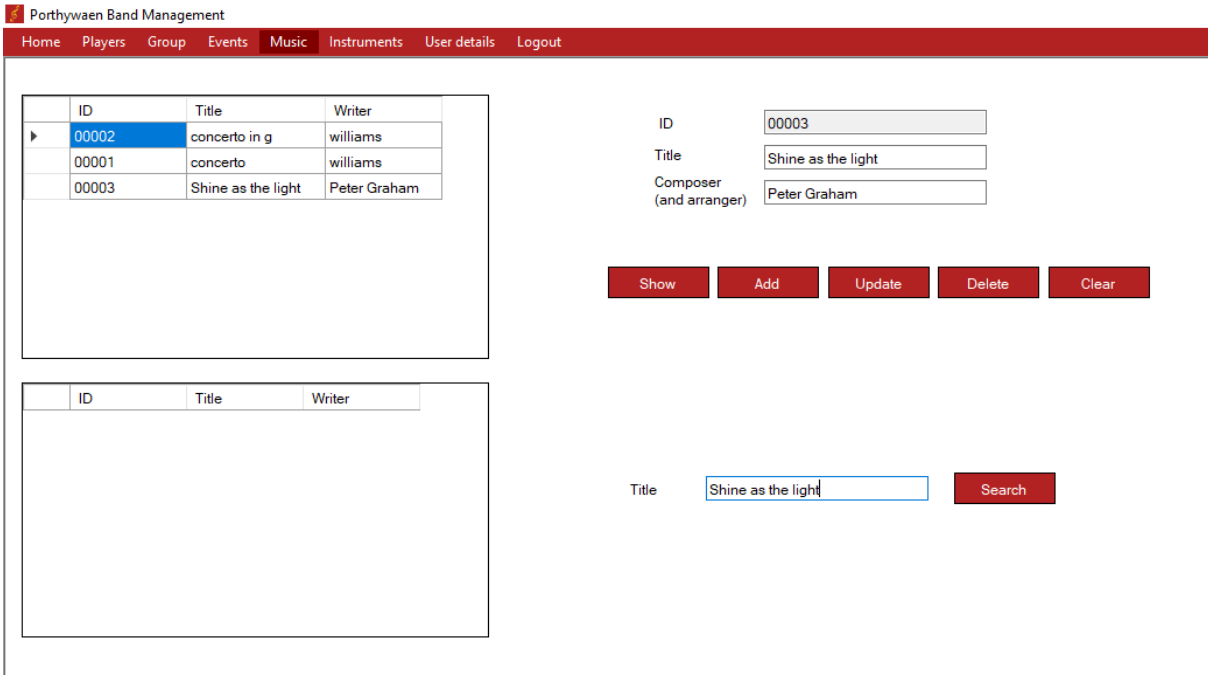

Figure 119: “Music” screen when the user enters a title to search for.

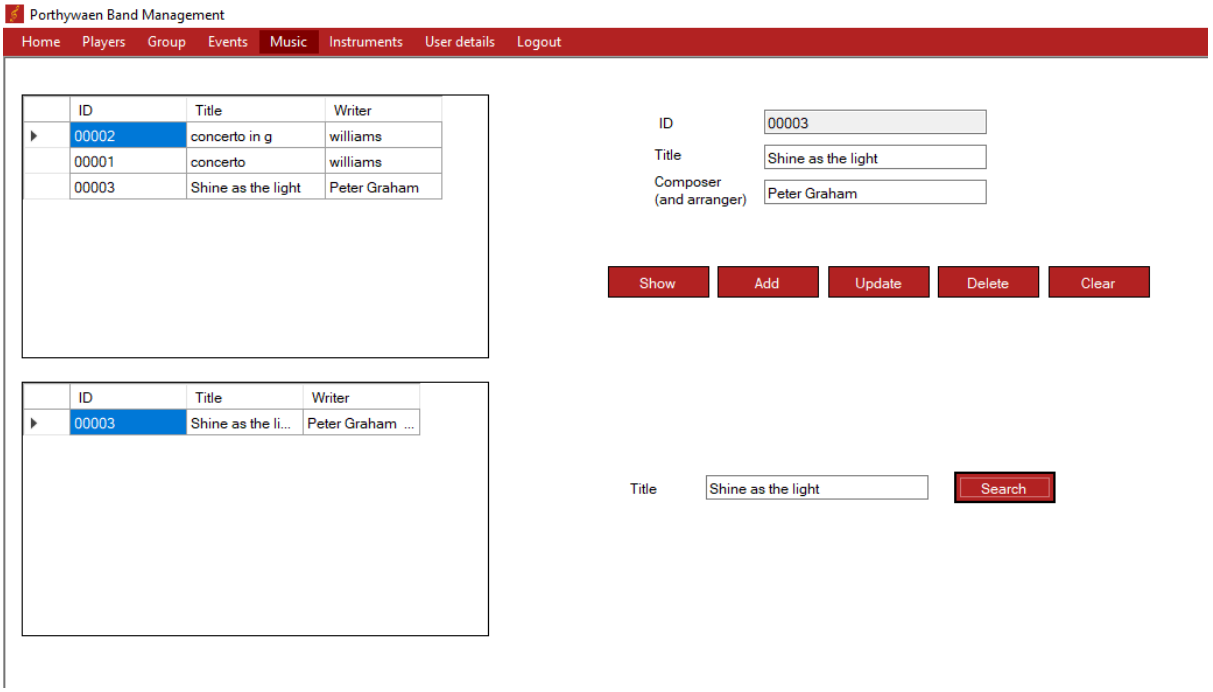

Figure 120: “Music” screen after “Search” clicked.

When testing this, I discovered that the search was case sensitive, meaning that if the user did not enter the title in the exact same format as what is stored, it will not be found.

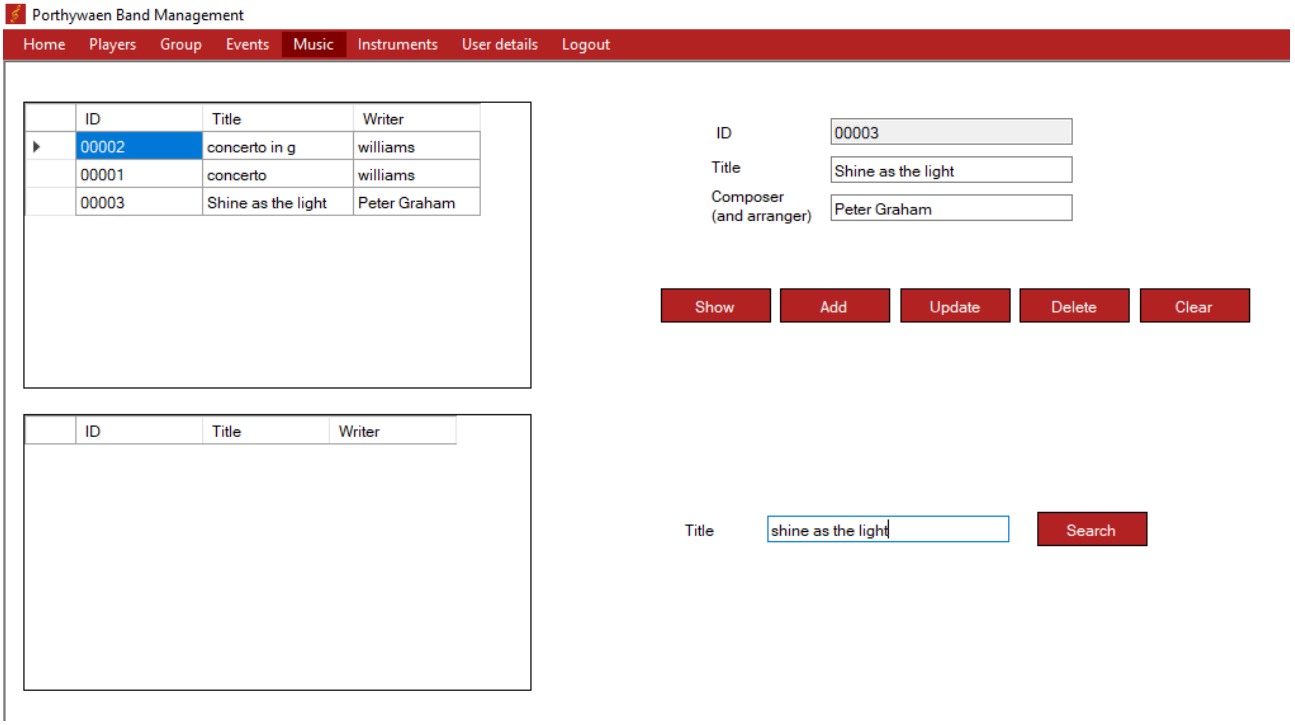

Figure 121: "Music" screen when the user enters a title to search for.

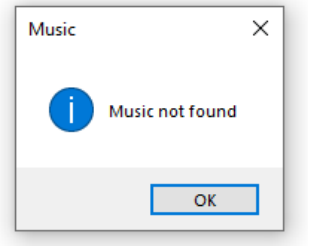

Figure 122: The message shown when the user clicks "Search" and the data inputted is not found.

I fixed this by changing the following line of code so that the title read is in the same format as the input. This code compares the title read from the database to the input. I amended the if statement so that the case does not matter when searching by adding the function ".ToLower".

```
dgvSearch.Rows.Clear()
FileOpen(1, "music.dat", OpenMode.Random,,, Len(onePiece))

While Not EOF(1)    'compare each item to piece searched
    FileGet(1, onePiece)
    If onePiece.title.ToLower.Contains(txtSearch.Text.ToLower) Then 'toLower used so search is not case sensitive
        dgvSearch.Rows.Add(onePiece.id, onePiece.title, onePiece.writer)
        found = True
    End If
End While
FileClose(1)
If found = False Then
    MsgBox("Music not found", vbInformation, "Music")
End If
```

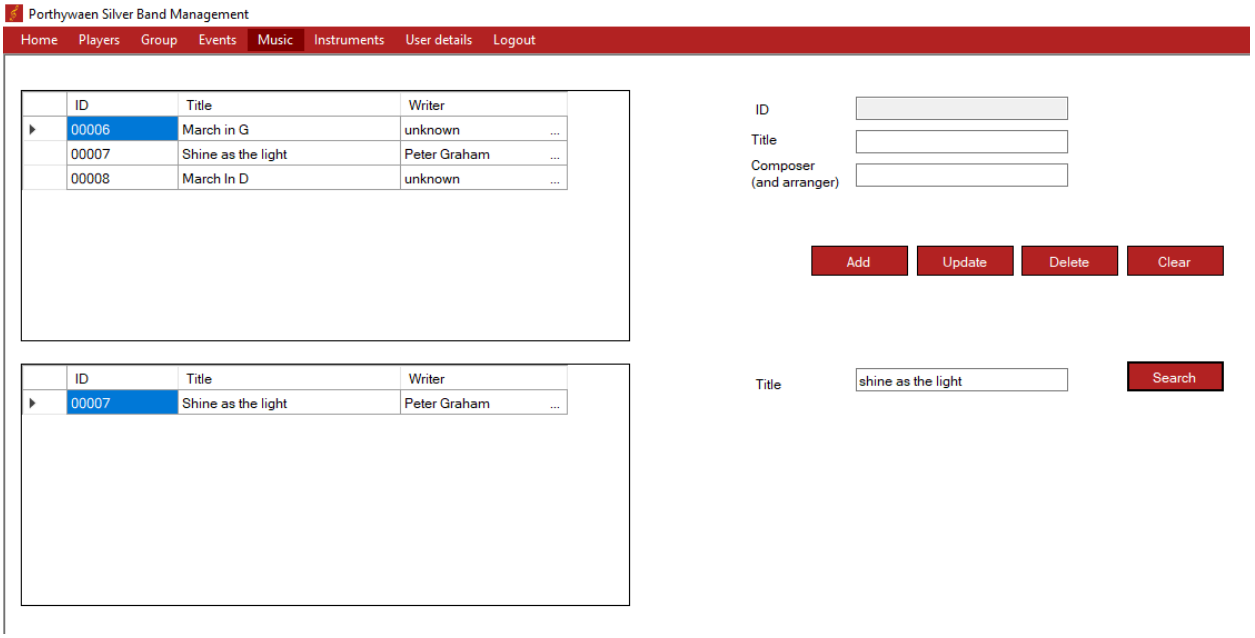

Figure 123: "Music" screen after "Search" clicked.

Test 46

This test checks that title is required when searching. When no data is entered into the search textbox and search is clicked, a message is shown informing the user that they need to enter a title or keyword to search.

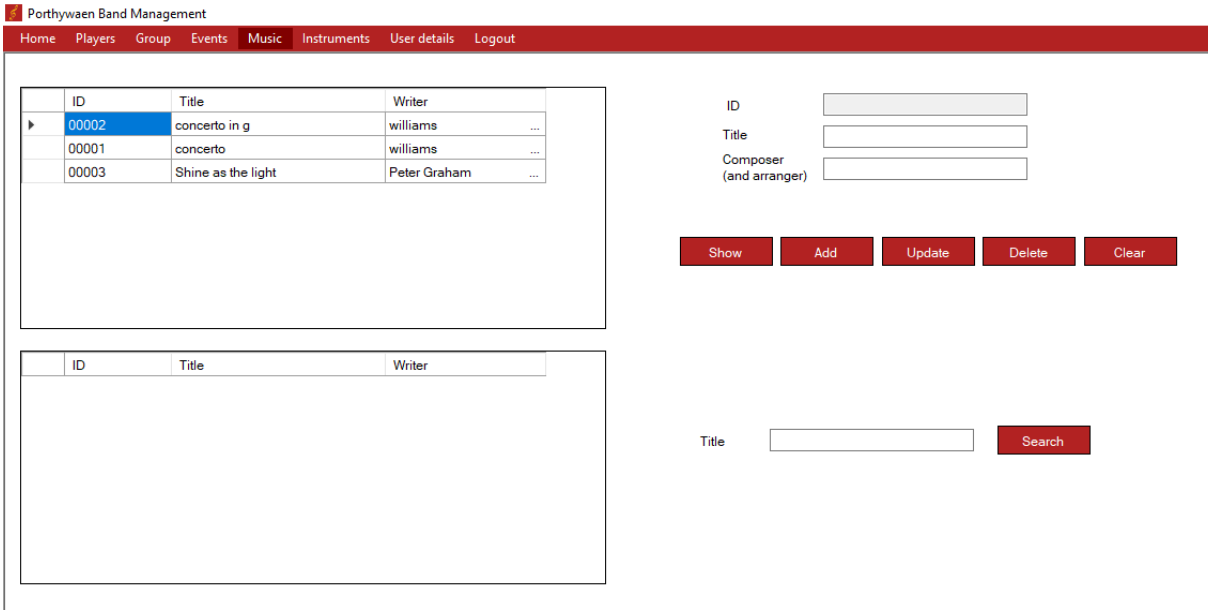

Figure 124: "Music" screen when the user clicks "Search" without entering data to search for.

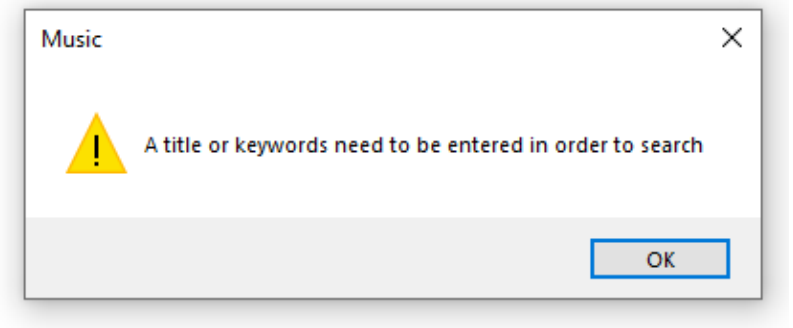

Figure 125: The message shown when the user clicks "Search" without entering data to search for.

Instruments

Test 47

This test checks that an instrument can be added. When the user enters data into the fields on the screen and clicks add, a subroutine is called which processes the inputted data. The data is stored in a structure and validated. The instrument ID is generated by searching the file for a possible ID and incrementing the ID by 1 each time an ID is found. The final ID is padded with zeros to ensure it is the same length as all the other IDs. The data is appended to the file and the DataGridView is refreshed to show the changes. A message is shown to confirm the instrument has been added.

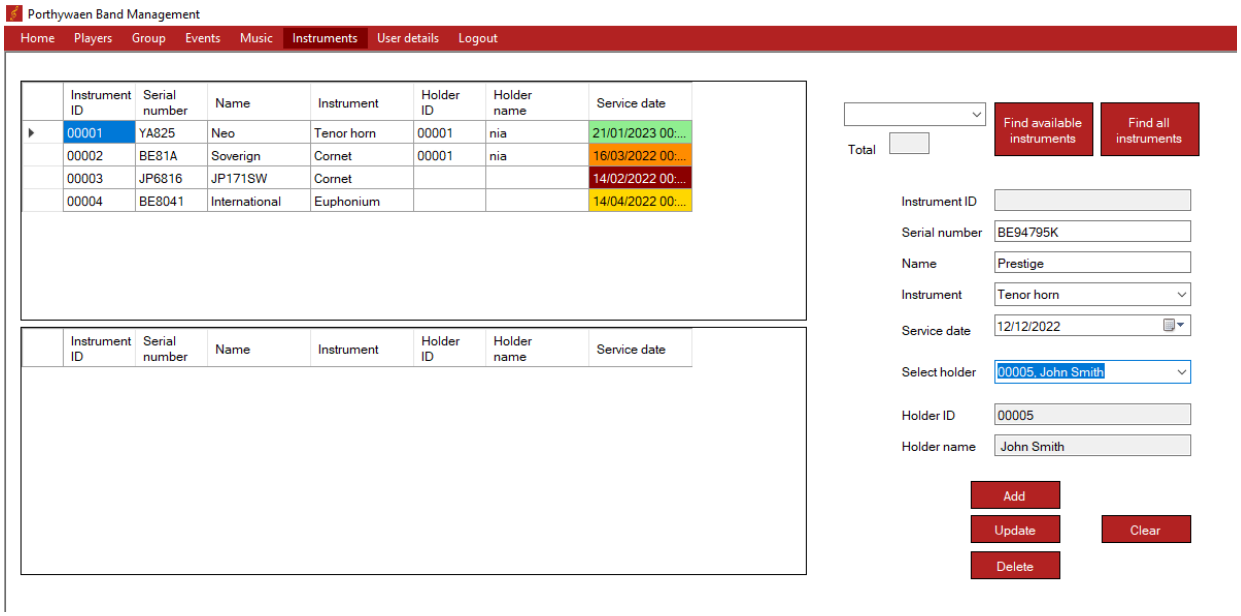

Figure 126: "Instruments" screen when the user is adding an instrument.

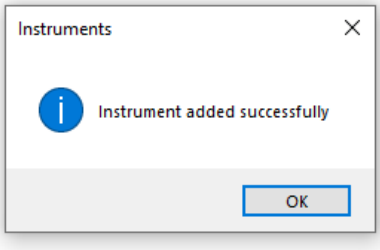

Figure 126: The message shown when the user clicks "Add".

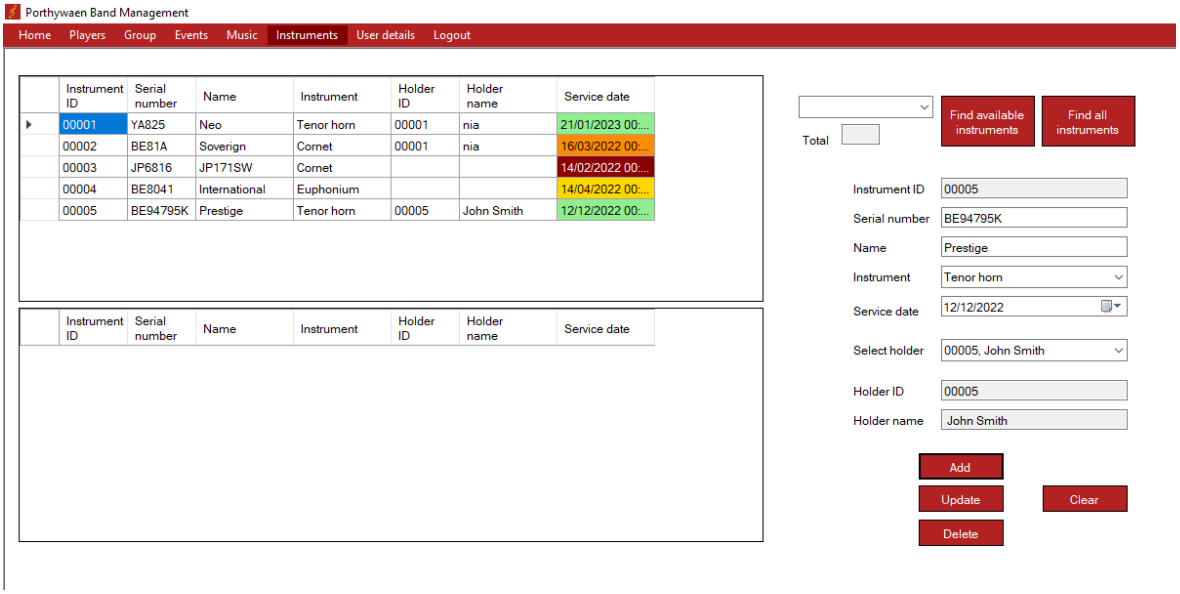

Figure 127: "Instruments" screen after an instrument has been added.

Test 48

This test checks that an instrument's details can be updated. When an instrument is selected the user is able to change any details in the input fields. When "Update" is clicked, the data inputted into the system is validated and then stored in a structure. The record number of the record to be updated is determined from the DataGridView cell clicked and that record is updated in the file. A message is shown and the DataGridView is refreshed to show the changes.

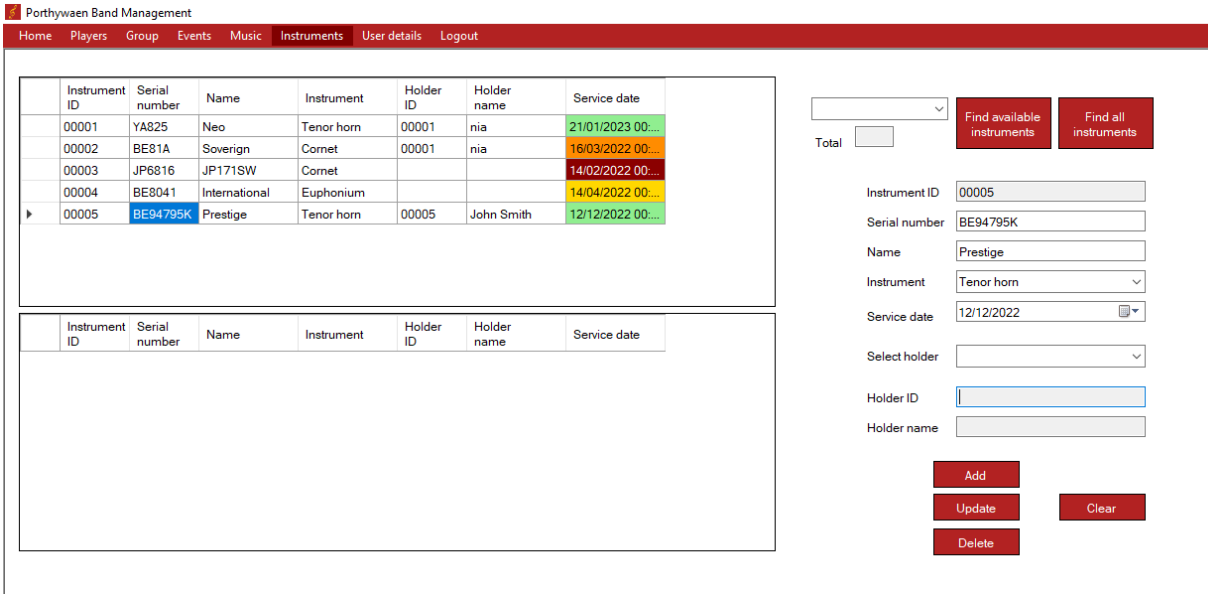

Figure 128: "Instruments" screen when the user is updating an instrument.

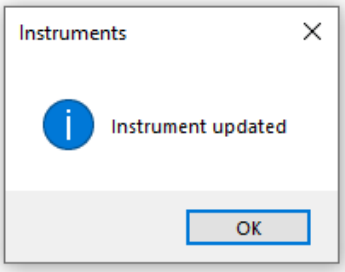

Figure 129: The message shown when the user clicks "Update".

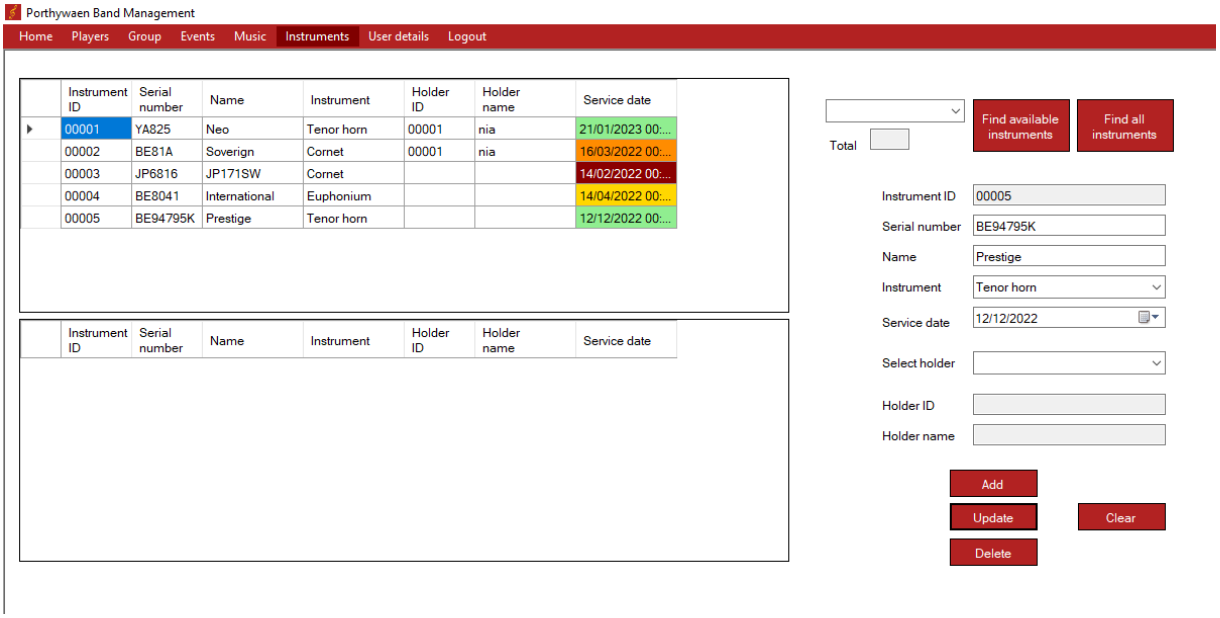

Figure 130: "Instruments" screen after an instrument has been updated.

Test 49

This test checks that an instrument can be deleted. Each record in the file is compared to the record that has been selected to be deleted. If it is not this record, then it is added to a temp file. When the record that is to be deleted is found, the record is skipped and not added to the temp file. When the end of the file is reached, the temp file is renamed and the old file is deleted. When an instrument is selected in the DataGridView and the user clicks "Delete", a message is shown confirming the instrument has been deleted and the DataGridView is refreshed to show that the instrument is no longer in the database. The instrument remains in the input fields, so the user is able to add the instrument again if they need to.

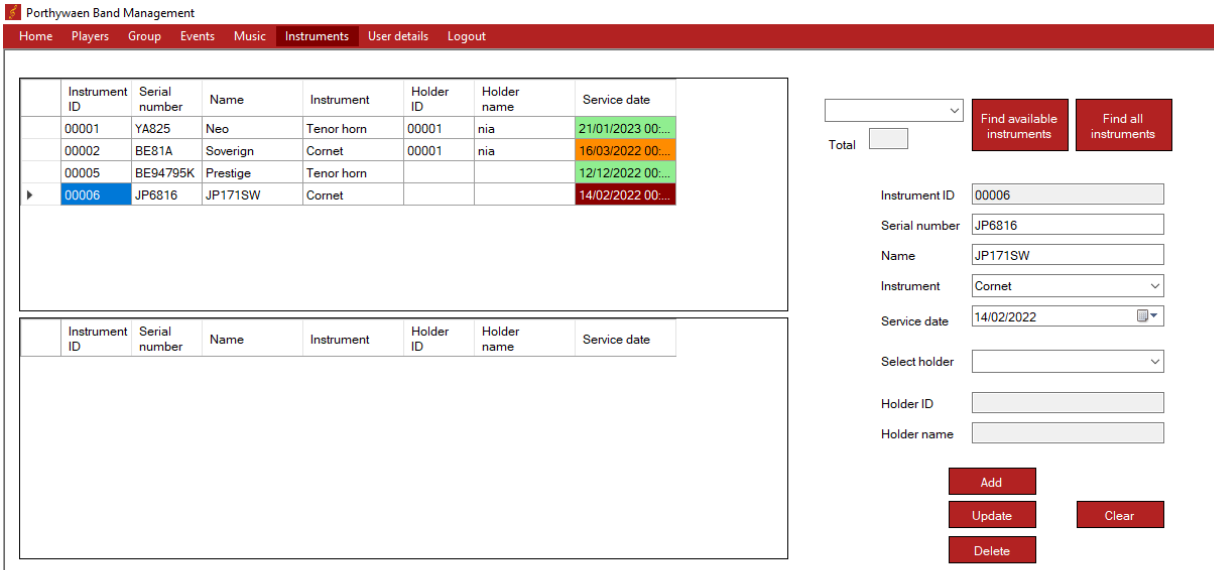

Figure 131: Before "Delete" is clicked, when an instrument has been selected.

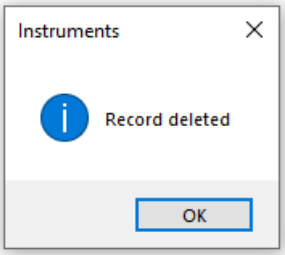

Figure 132: The message shown when "Delete" is clicked.

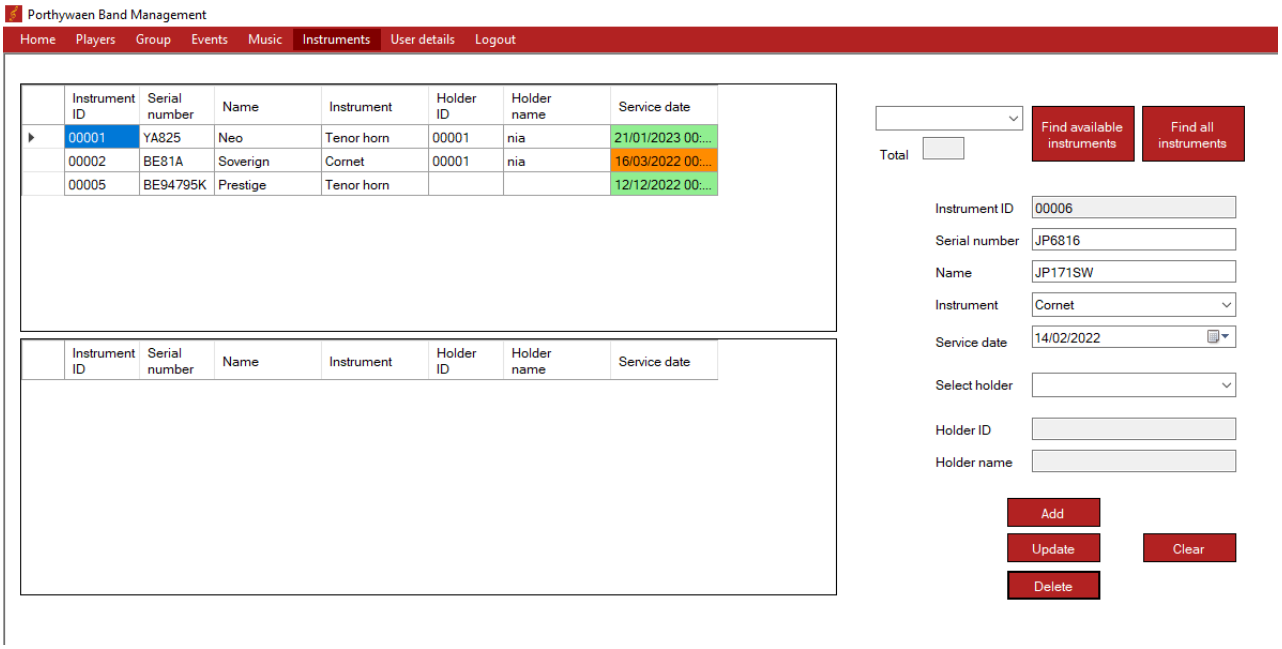

Figure 133: "Instruments" screen after an instrument has been deleted.

Test 50

This test checks that the serial number is needed. When the user only leaves the serial number textbox empty and clicks "Add" or "Update", a message will be shown and no data is stored in the database.

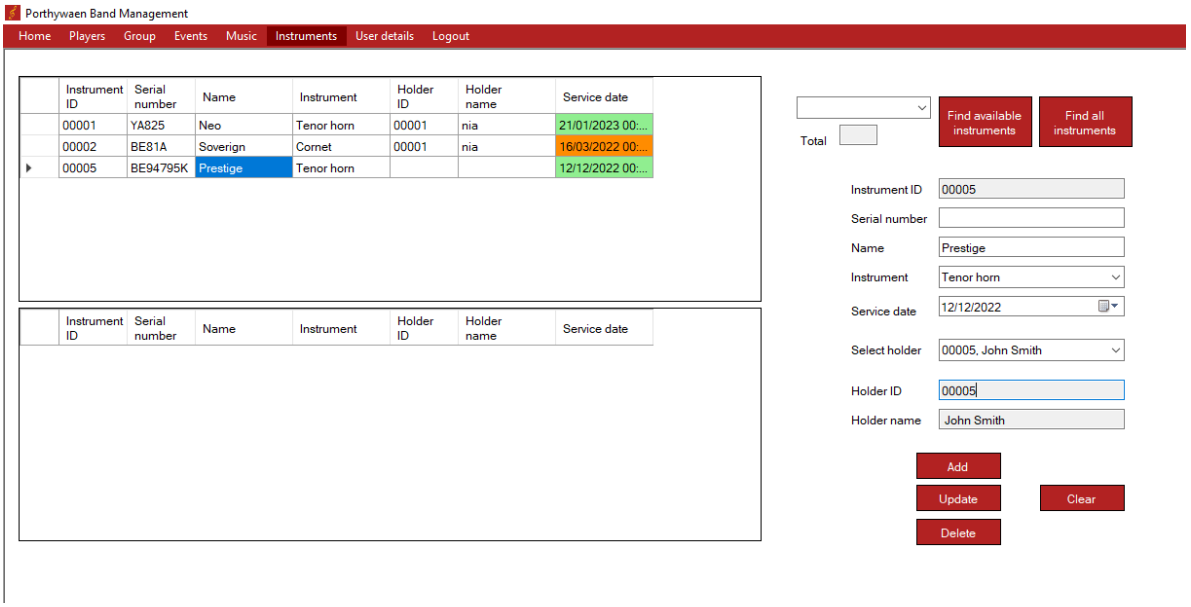

Figure 134: "Instruments" screen when the user adds or updates an instrument without entering the instrument's serial number.

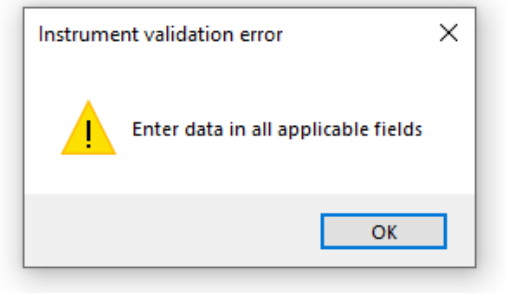

Figure 135: The message shown when the user clicks "Add" or "Update"without entering the instrument's serial number.

Test 51

This test checks that the service date is a valid date. When the user inputs data that is before 1900 and clicks "Add" or "Update", a message will be shown and the data will not be stored in the file.

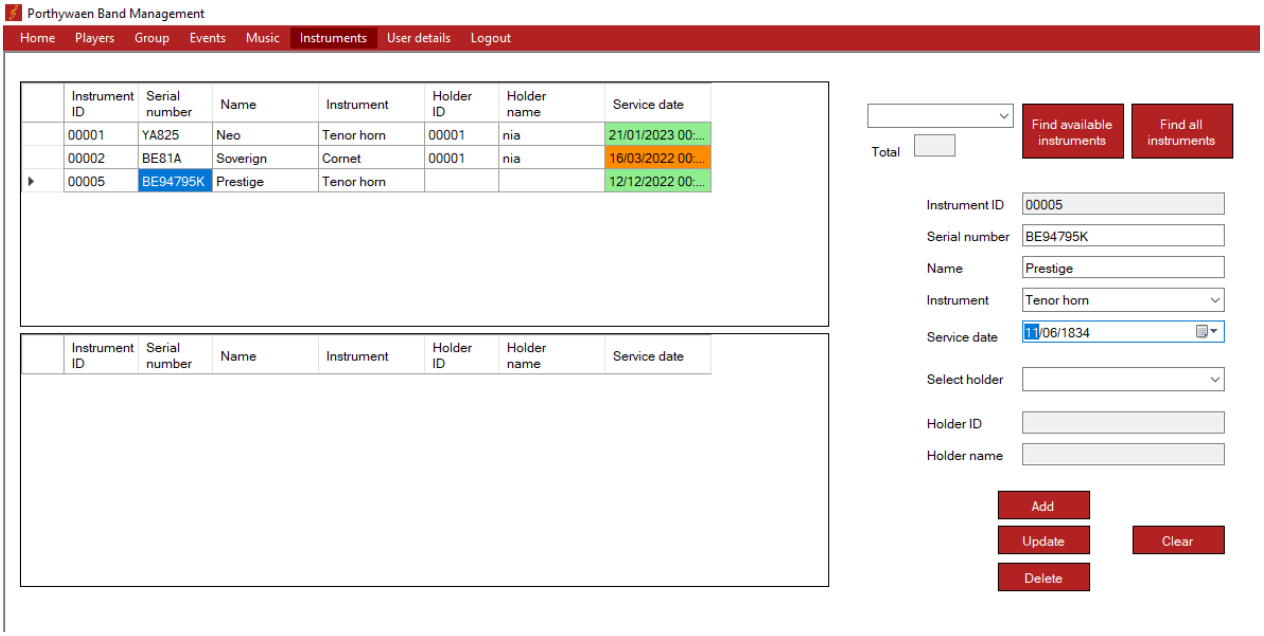

Figure 136: Before “Add” or “Update” is clicked, with an invalid service date inputted.

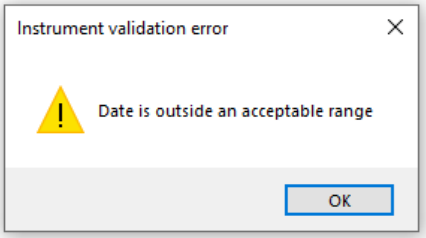

Figure 137: The message shown after “Add” or “Update” is clicked, when an invalid service date is inputted.

Test 52

This test checks that only players in the system can be instrument holders. When adding a new instrument or updating an existing instrument, the user can select a player in a combo box, and the applicable details from the instrument's record are shown in the textboxes below it. The data in the combo box is loaded into it as the instruments form is shown from the players database, by reading each record into a structure and then adding each ID and name element to a string which is added to the combobox's list of data items.

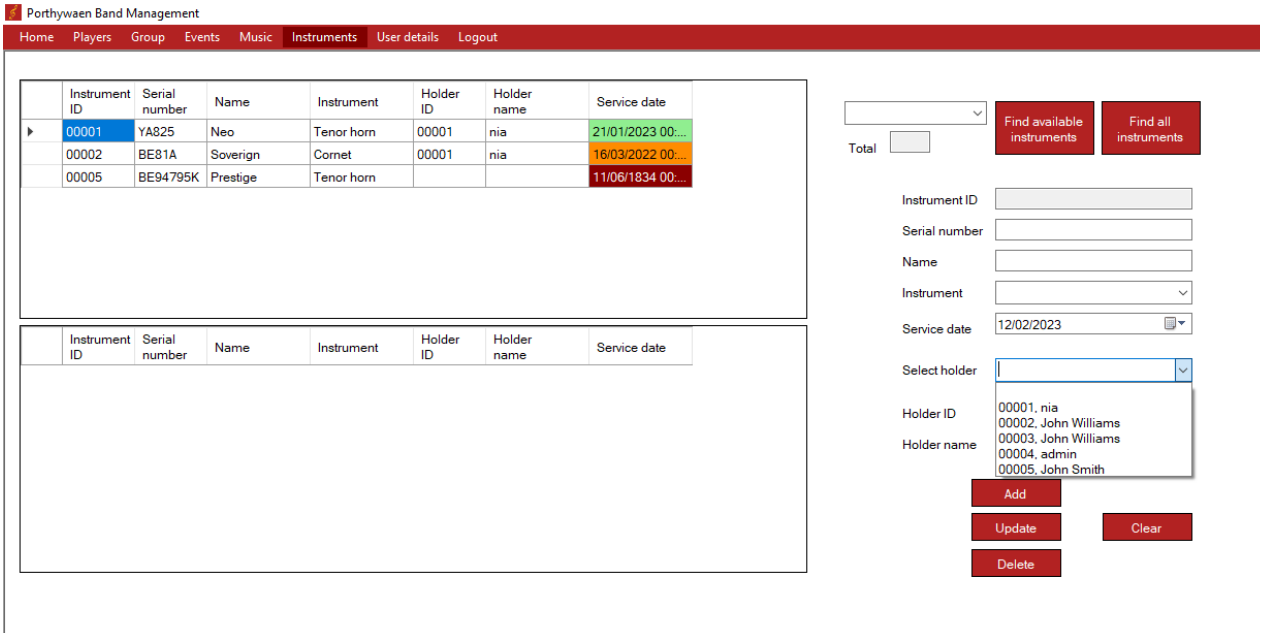

Figure 138: "Instruments" screen when the user expands the select holder box.

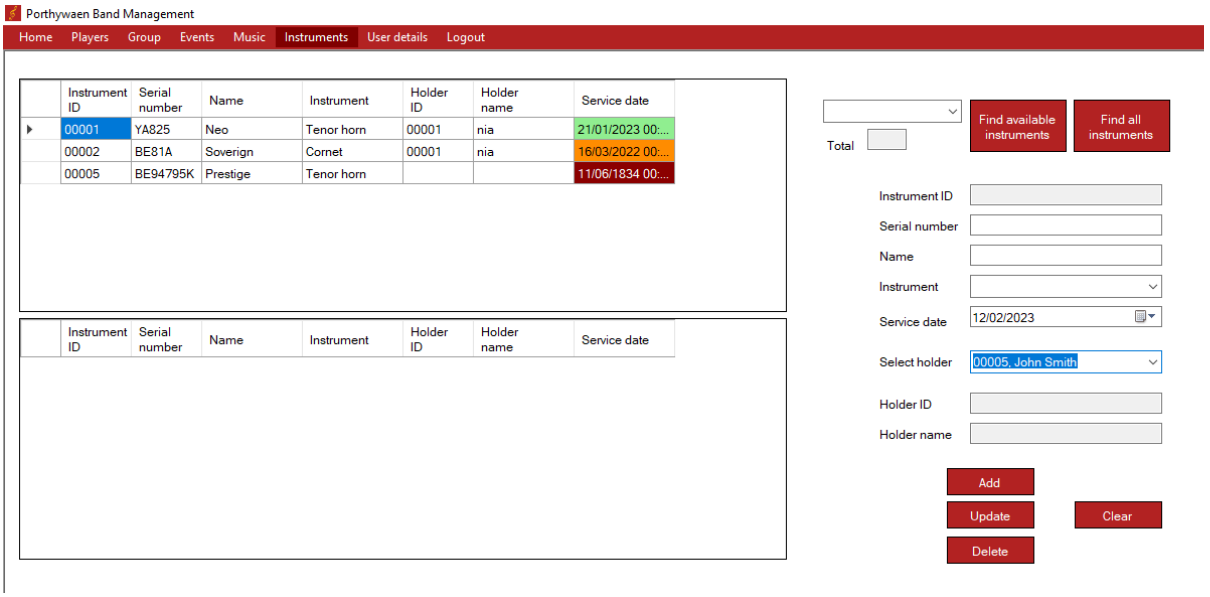

Figure 139: "Instruments" screen when the user has selected a holder in the select holder box.

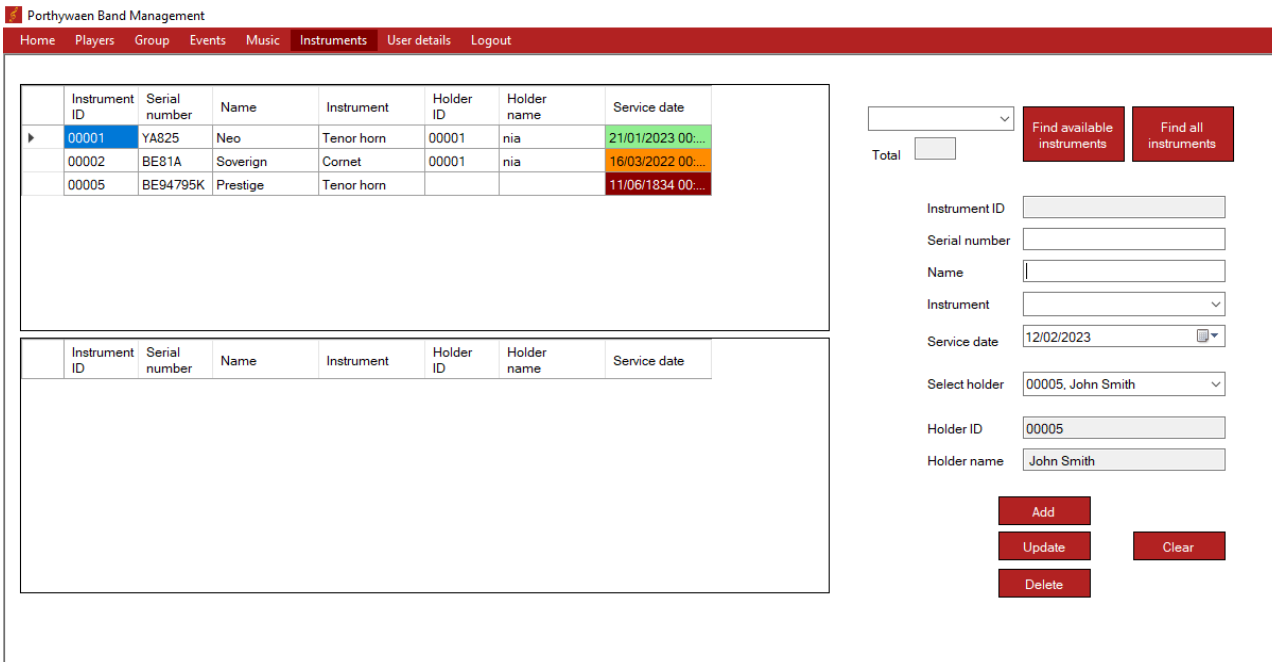

Figure 140: "Instruments" screen when the user has exited the select holder box.

Logout

Test 53

This test checks that a user can log out of the system. They can do this by clicking logout in the menu bar. This displays a confirmation message, to ensure the user does want to logout of the system. When yes is clicked, the program is closed and when no is clicked, the program remains open.

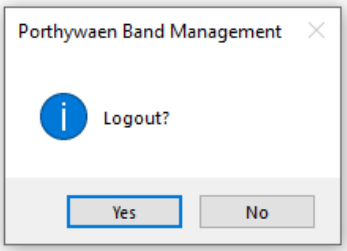

Figure 141: The message shown when "Logout" is clicked.