Programmed Solution to a Problem - Prototype

Porth-y-waen Silver Band Management System

Nia Hawkins: 7183
The Maelor School: 68146

Contents

Prototype

After completing the designs of the user interface, data structures and processing stages, I will develop a prototype of the system to demonstrate how the key features of the system will work.

Justification of the choice of areas to be included in the prototype system

I will create a prototype of the system to demonstrate the functionality the final program will have. I will include only the key areas of the system to test if the main functionality will work.

The prototype will allow the user to:

- **Navigate between different aspects of the system.**
    - Each file that makes up the database corresponds to a different screen which will be opened by clicking the menu bar
    - This makes the system more organised when developing it and also makes it easier for the user to use.

- **Add, update and delete records from the members, events, music and instruments files.**
    - These features are a key aspect of the system as the purpose of the system is to store the band's data.
    - New records may need to be added - e.g. a player joins or new music is bought; existing records may need to be updated - e.g. a player's phone number changes; and some may need to be removed - e.g. an instrument is sold.
    - Attendance is also regularly recorded at rehearsals, and the system must determine if an update is required for each input, or simply write it to the file.
    - Therefore, data must be able to be inputted, modified and deleted from the database.

- **Output members, events, music, instruments, players, groups and attendance at rehearsals.**
    - The user needs to be able to view the data stored in the database so they are able to use the data to find a suitable player or instrument, discover if the band is available on a specific date, or find a player's email address to contact them directly, for example.

- **Search for a specific field in the relevant files in the database**
    - This will allow the user to access the data they require quickly and easily without reading through the whole database.

- **Clear any input fields**
    - This allows the user to input new data into the system without having to clear each field as they do so, making it easier to use.

Areas that are not included in the prototype:

- **Different levels of access to the system**
    - They will include a selection of the screens already created and will not have any additional features to the conductor's view.

- **Login, logout or reset password**
    - These are not essential features of the system functionality as the purpose of the prototype is to demonstrate the data processing aspects of the system.
    - I do not need to have a login to the system as the developer, and this will also make it harder to carry out developmental testing as more processes are required to test the functionality of a feature.

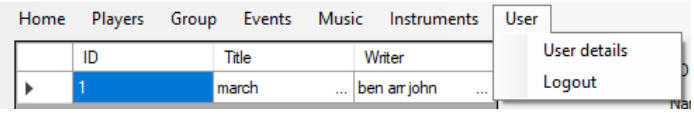
- **User details**
    - The user details screen will show the details of the player that is currently logged into the system.
    - Therefore, it will require the same data and processing as the players screen.
    - The processing will be duplicated so will not require extensive further testing.

- **Validation and verification of inputs**
    - The validation and verification of inputs is not required for data processing, as long as the data inputted is reasonable.
    - The prototype will be shown to a select group of people who know the inputs required and will be able to contact me if any significant errors occur.

- **Colours**
    - The prototype uses the default colours of Visual Basic controls and not those specified by the end user in the investigation.
    - The colours of the controls are not needed to demonstrate the functionality of the system.

- **Player event availability**
    - This feature contains the same data processing as attendance as it requires the same data input and stores it in the database
    - The data will also require a similar method to produce an output by reading the relevant data from the database.

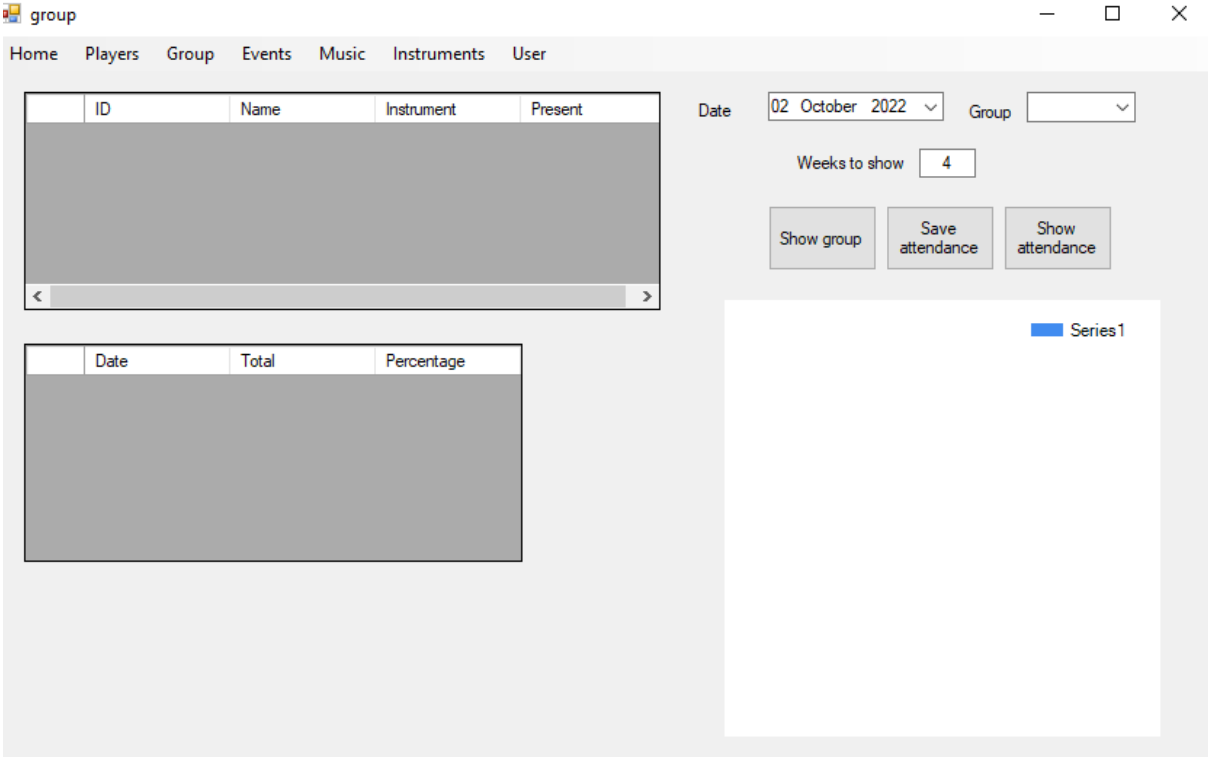
Screens and outputs for the chosen area of the solution



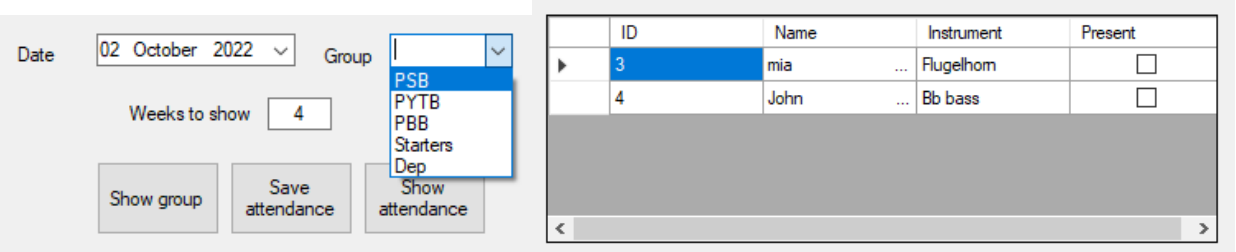
Each screen has a menu bar across the top to allow the user to navigate between each screen. When the menu is clicked, the relevant screen will open and the last screen will be closed.

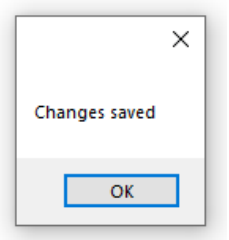
Group

This screen allows the user to view all members in a chosen group. They can mark each member's attendance for the rehearsal and can view the group's overall attendance.

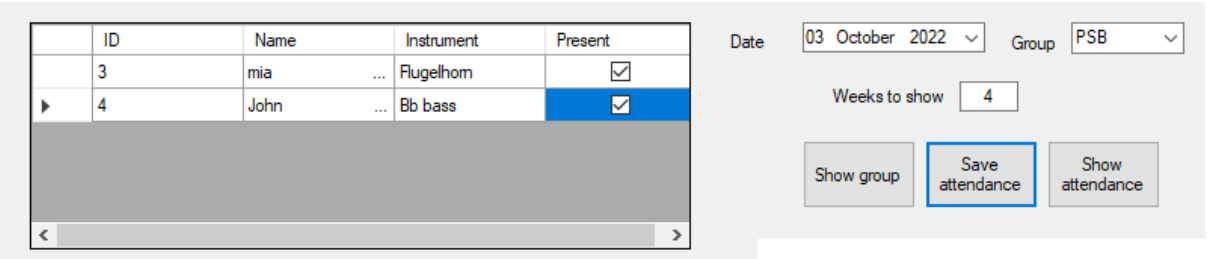


The user can select a group to view in the dropdown box, and all members will be displayed when the *show group* button is clicked.

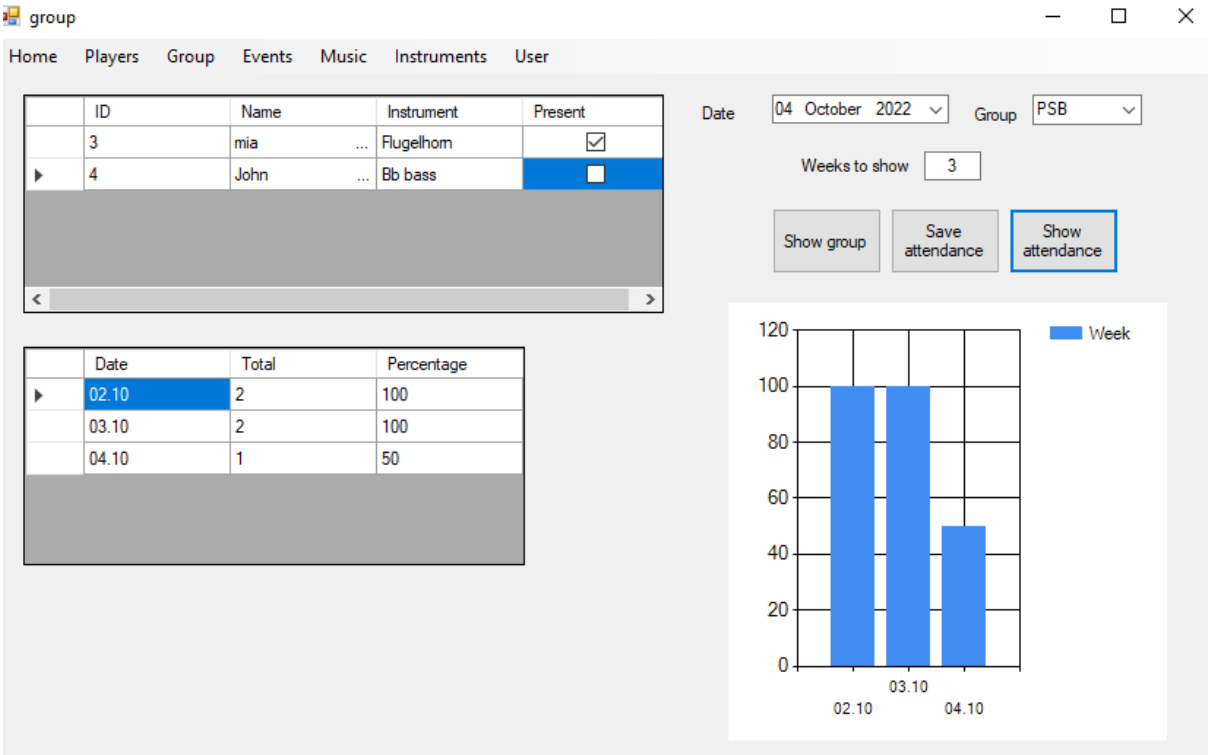




Attendance can be recorded by ticking the boxes next to the members. When *save attendance* is clicked, the marks and the date will be stored.
A message is shown to tell the user the attendance has been recorded.



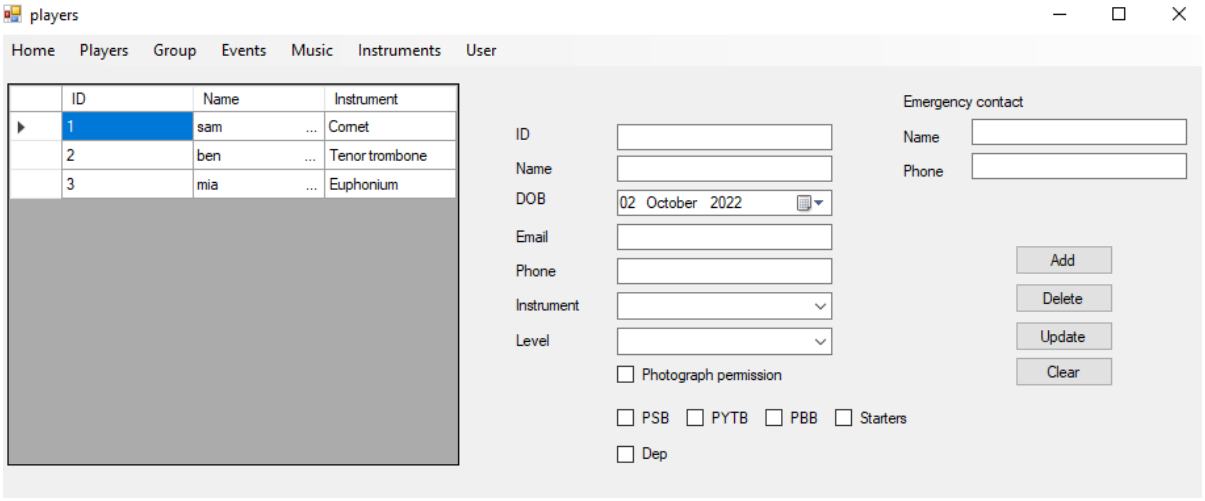
The user can select the group and the number of dates to show. When *show attendance* is clicked, the data will be displayed in a DataGridView and graph form.

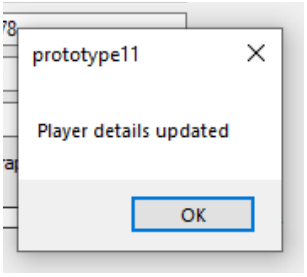


Players

This screen allows the user to view all players stored in the system, view each player's details, add a new player, update a player's details and delete a player.

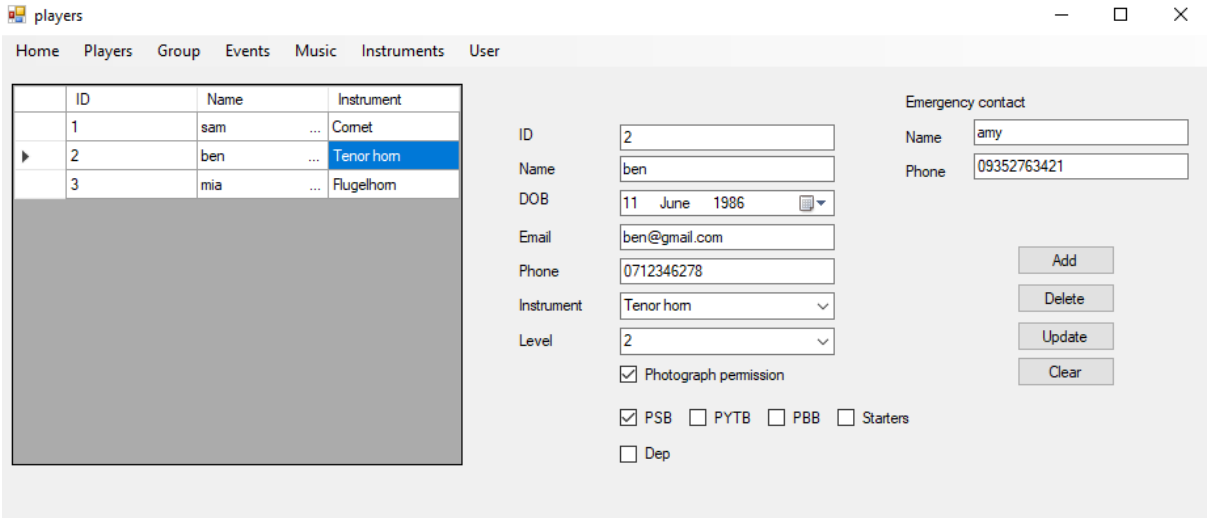
When the screen loads, all players stored in the database are displayed in the DataGridView.

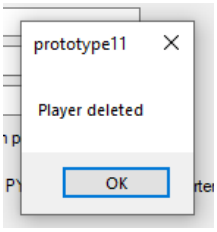




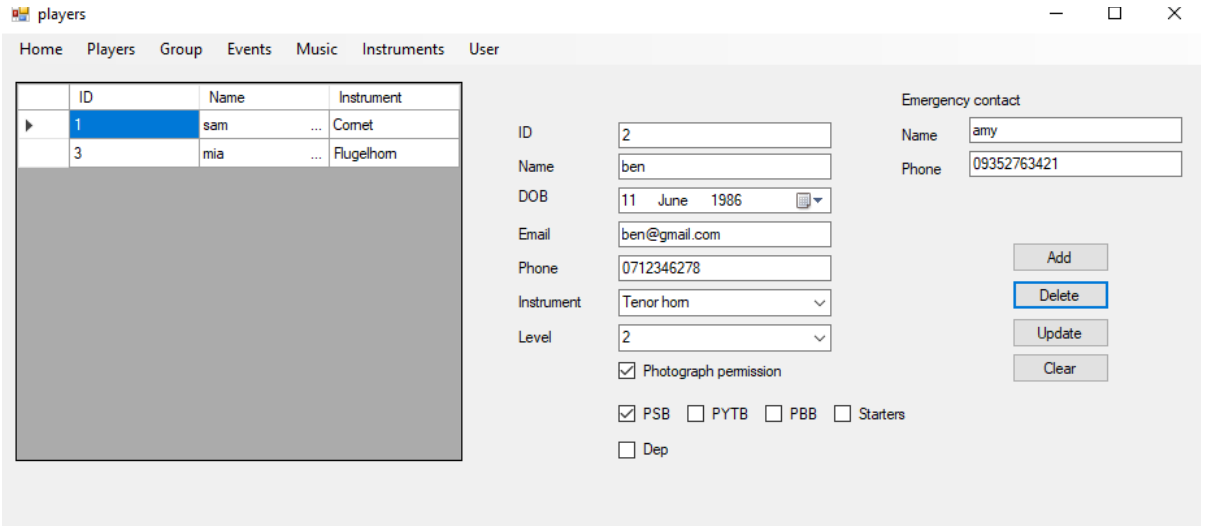
When a cell in the table is clicked, the player's details are displayed in the boxes. This allows the user to see all the information about one player. They can then edit the player's details or remove them from the system.

When *btnUpdate* is clicked, the player's details will be updated in the file and the DataGridView will refresh to show the changes. A message will be displayed to confirm that the record has been successfully changed.

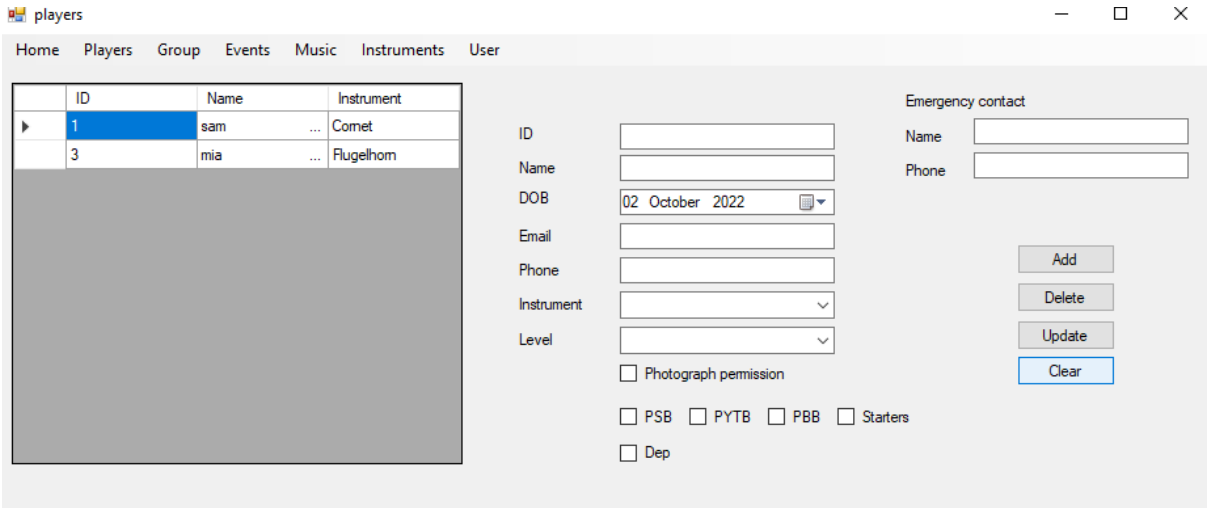


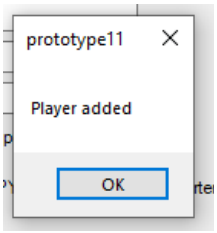


When *btnDelete* is clicked, the player will be removed from the database. A message is displayed to confirm the record has been deleted and the DataGridView will not contain the member.

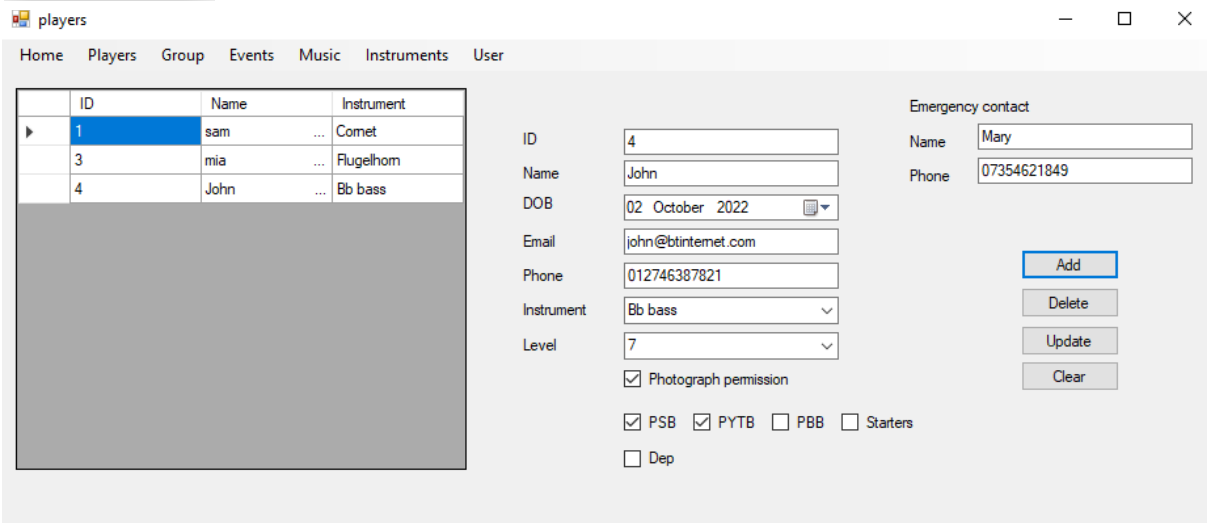


btnClear resets all the boxes on the screen so a new player can be added to the system.





To add a new player to the system, the user can enter their details into the screen and click *btnAdd*. The new player will be displayed in the DataGridView. A confirmation message is displayed.

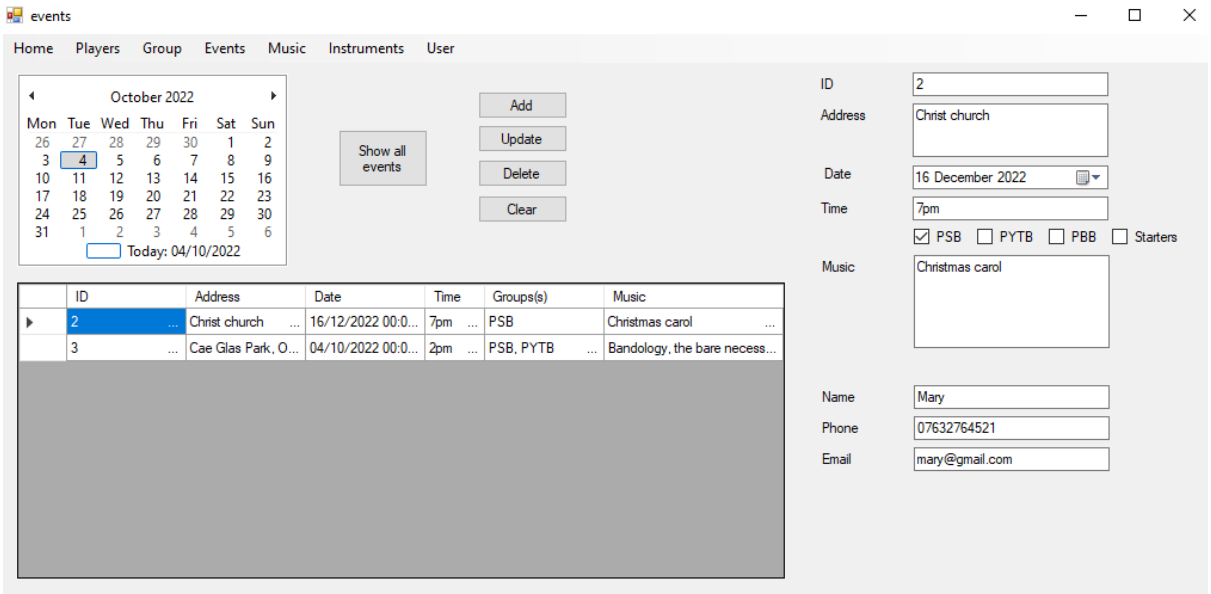


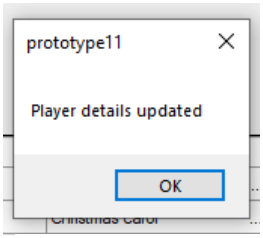
Events

This screen allows the user to view, and manage event bookings.
When the screen loads, and *show all events* is clicked, all bookings stored in the system will be displayed in the DataGridView.

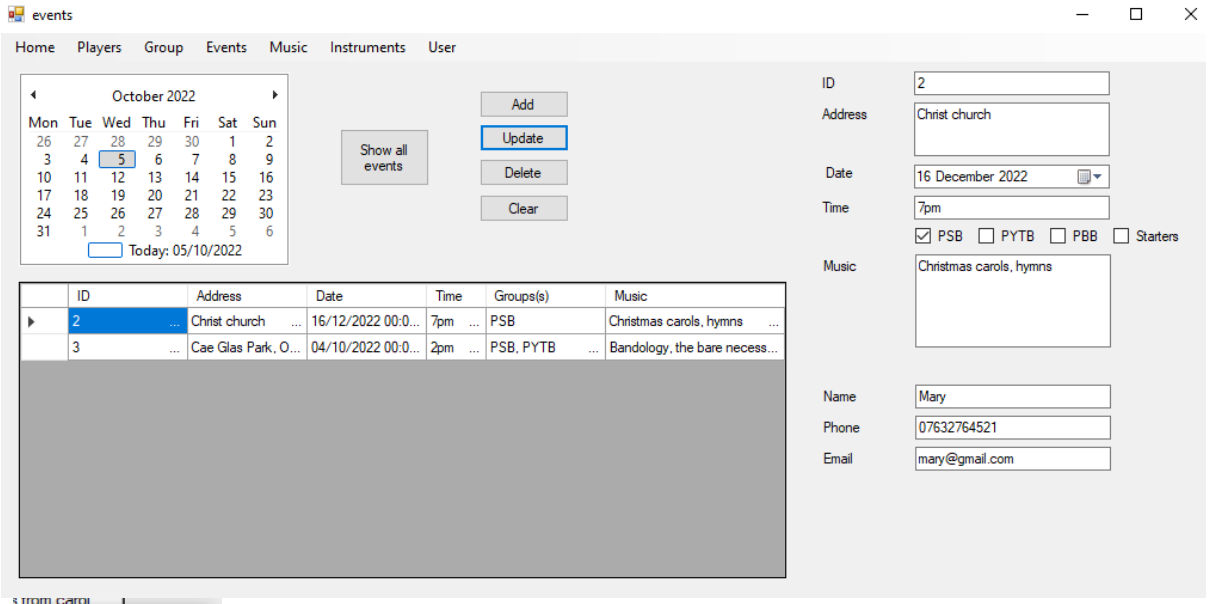
When *clear* is clicked, the inputs will be set to empty or their default values.

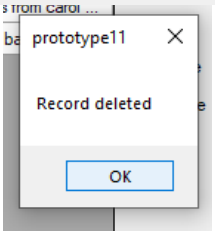
When a DataGridView cell is clicked, the details from the booking clicked will be displayed in the relevant input section. This allows the user to amend or remove the booking.



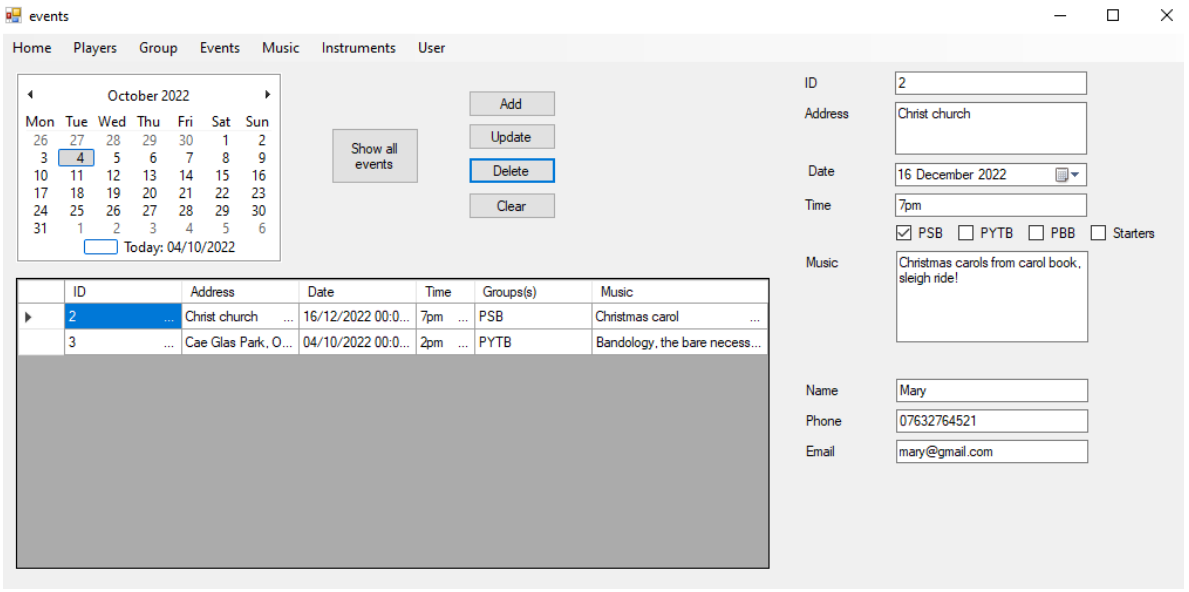


When *update* is clicked, the changes to the booking will be saved in the system. A message is displayed to confirm the changes have been saved successfully.

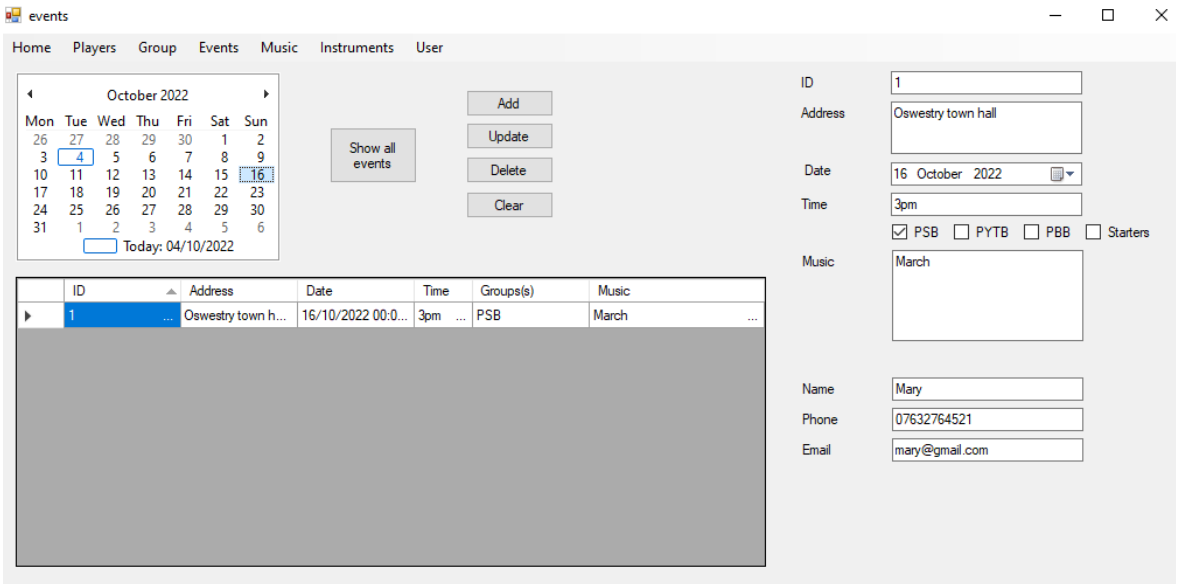


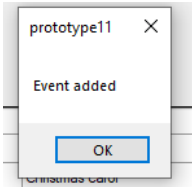


To delete a booking from the system, once the booking has been selected, the user clicks *delete*. A message is shown and the DataGridView is updated to show the booking has been removed.

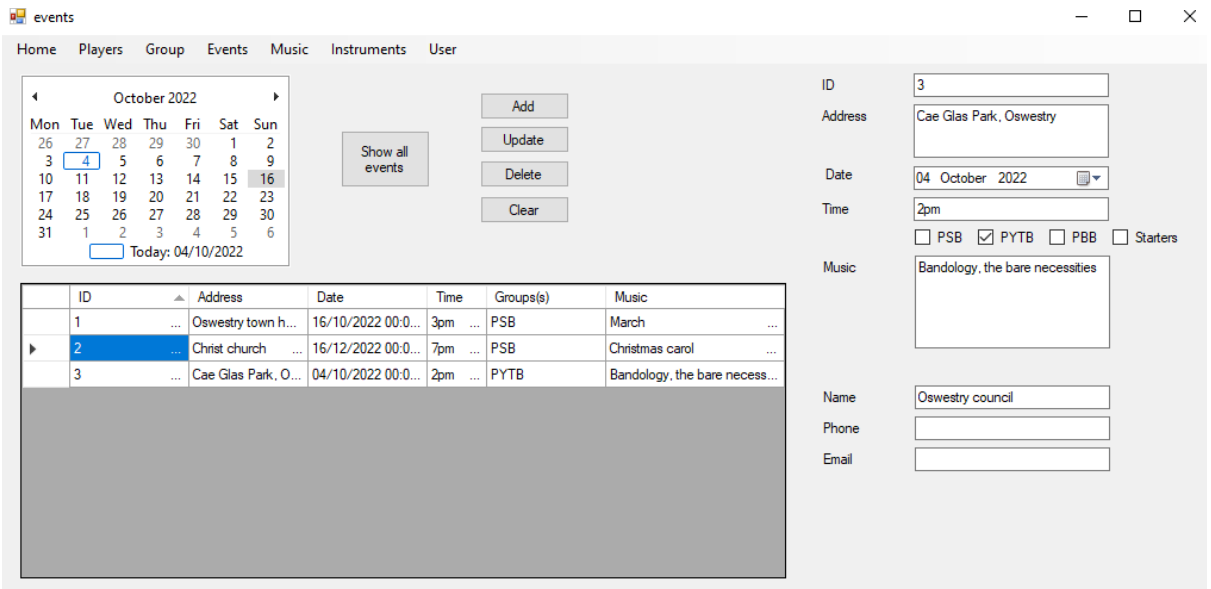


When a date in the calendar is clicked, any bookings on that date will be shown in the DataGridView.



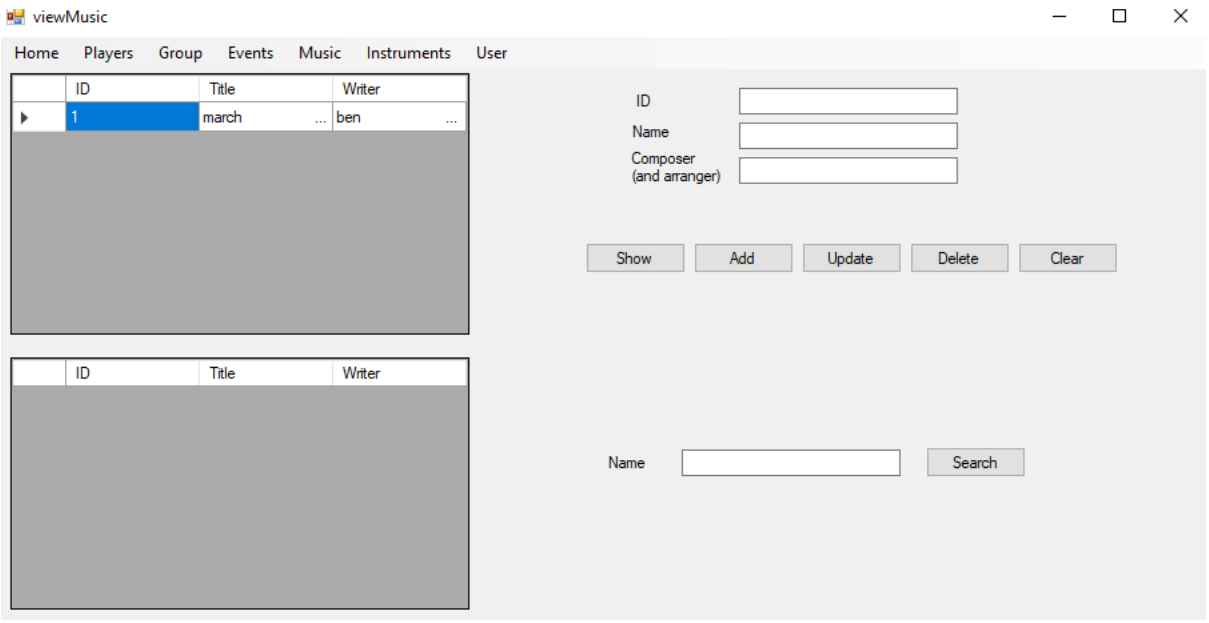


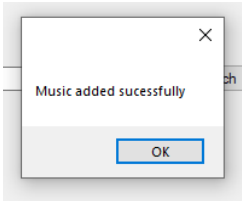
The user can add an event to the system by entering the event details into the screen and clicking *add*. A message will be displayed and the DataGridView will be automatically refreshed to show the new event.



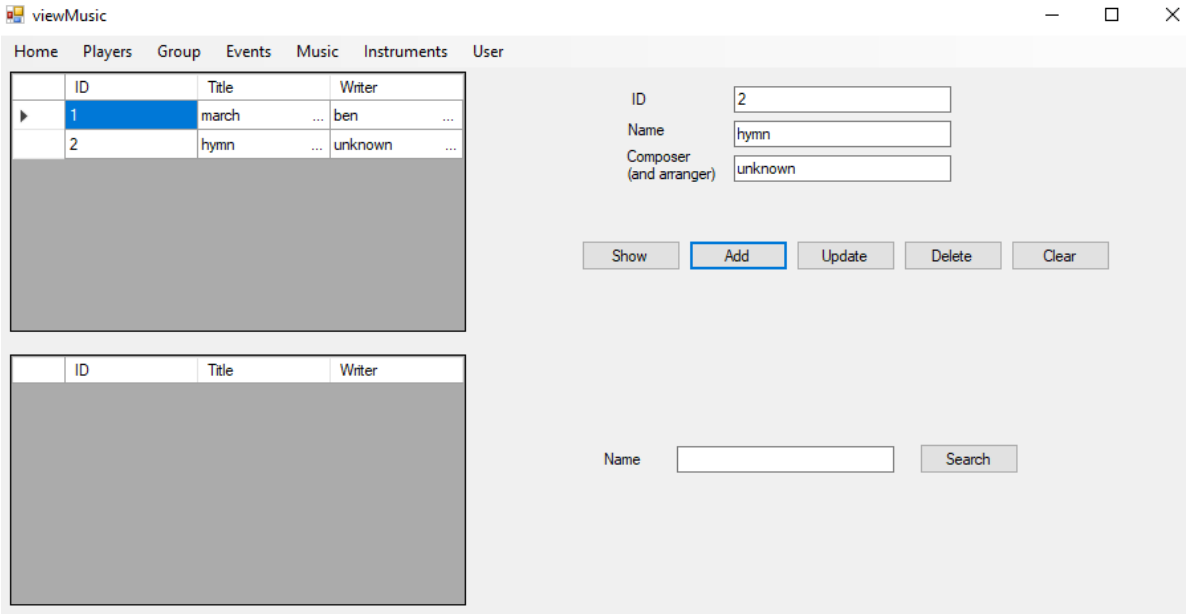
Music

This screen allows the user to view and manage the music library.
When the screen loads, the music stored in the system is displayed in a DataGridView and all inputs are empty. When *clear* is clicked, the inputs are all reset.

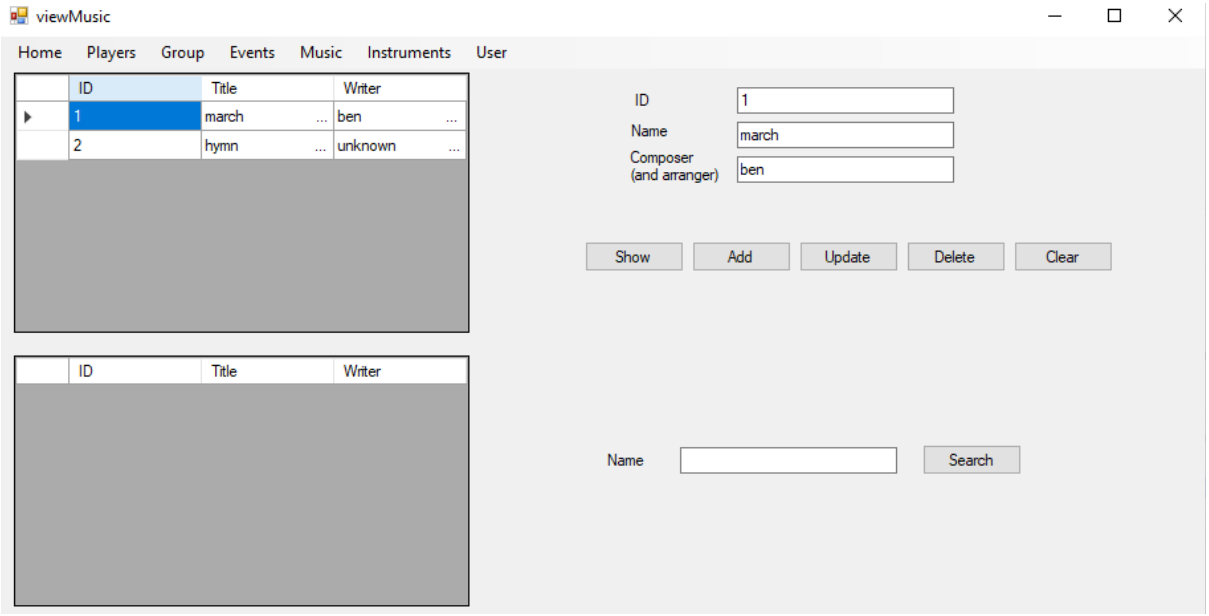


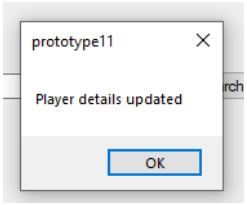


Music can be added to the system by inputting the piece's details into the boxes and clicking *add*. A message will be shown and the DataGridView will refresh automatically to show the new music in the system.

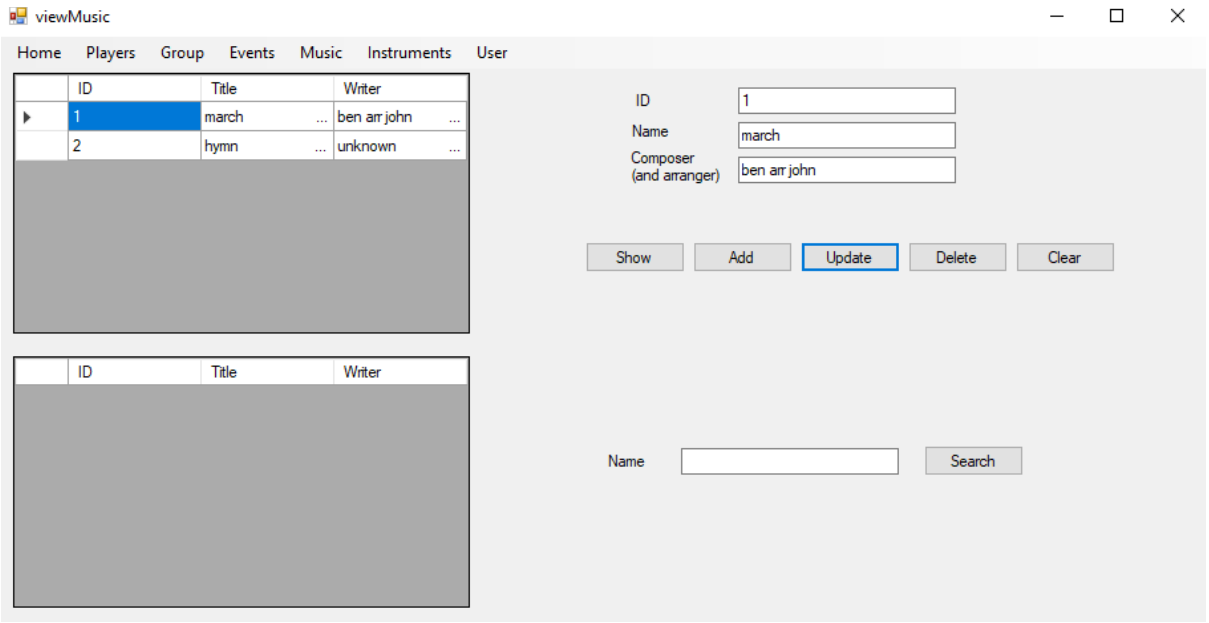


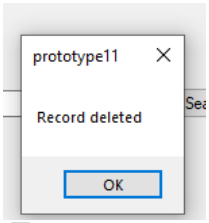
When a piece in the DataGridView is clicked, the details are displayed in the input boxes so the piece can be edited and deleted.



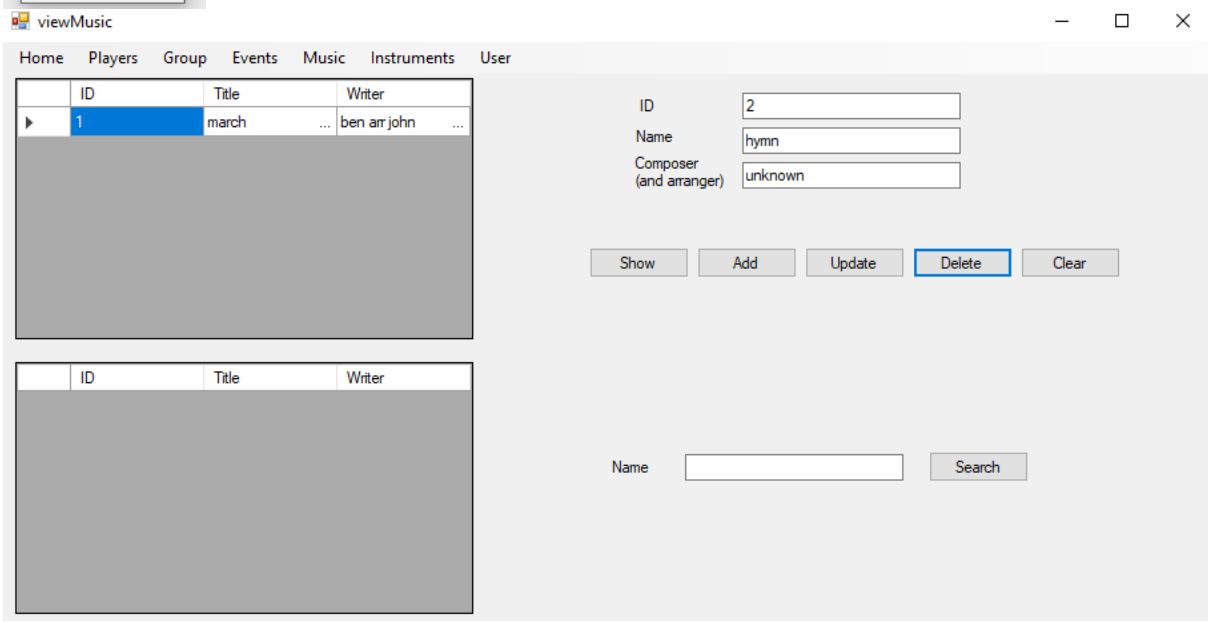


When the user has changed the details, *update* is clicked to save the music to the system. A message is shown and the DataGridView is refreshed to show the changes.

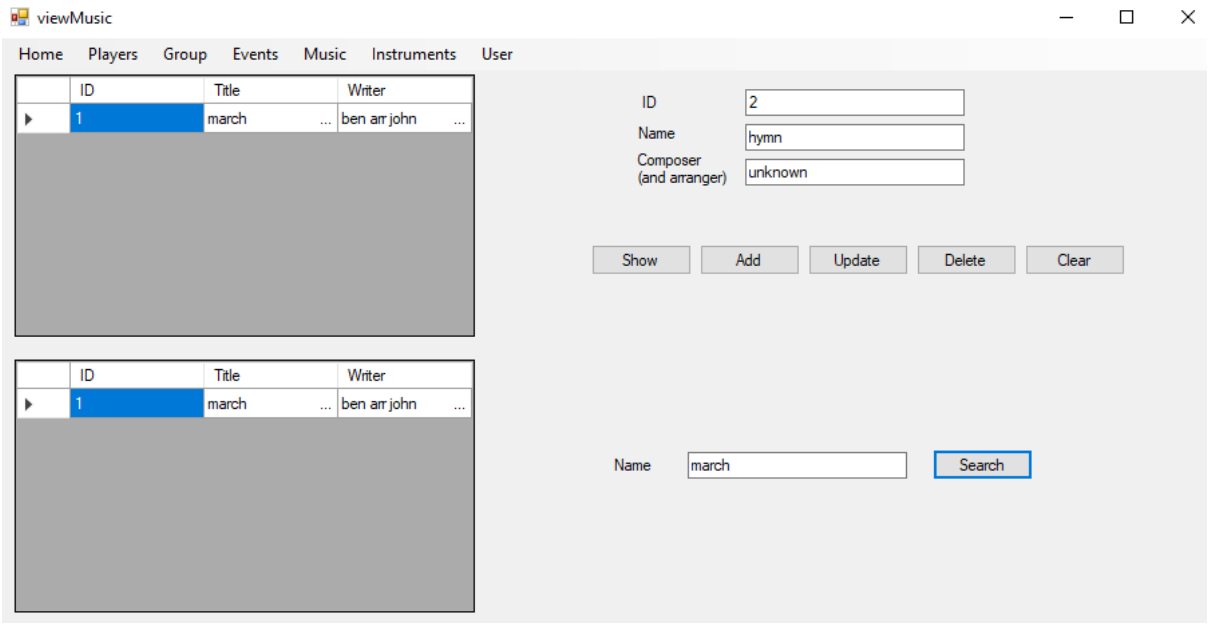


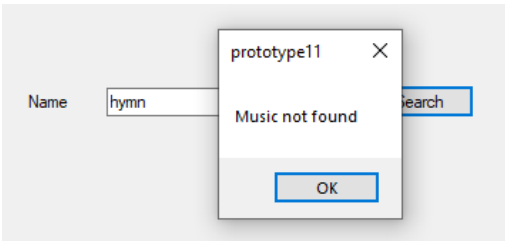


If a piece needs to be removed from the system, once it is selected, the user clicks *delete*. A message is displayed and the DataGridView is refreshed to show the music has been deleted.



To search for a piece, the user enters the name into the search box and clicks *search*. If the music is found it is displayed in DataGridView. If the music is not in the system, a message is displayed to tell the user.



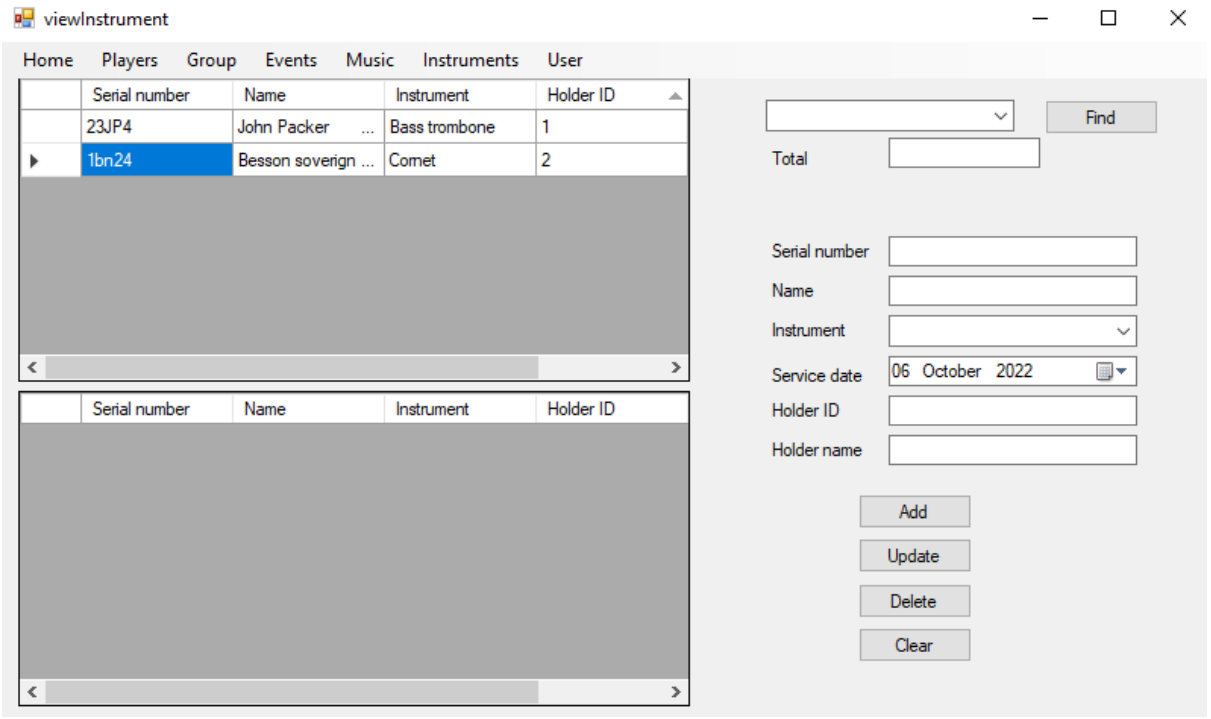


'Hymn' has already been deleted from the system so it is not found when the database is searched.

Instruments

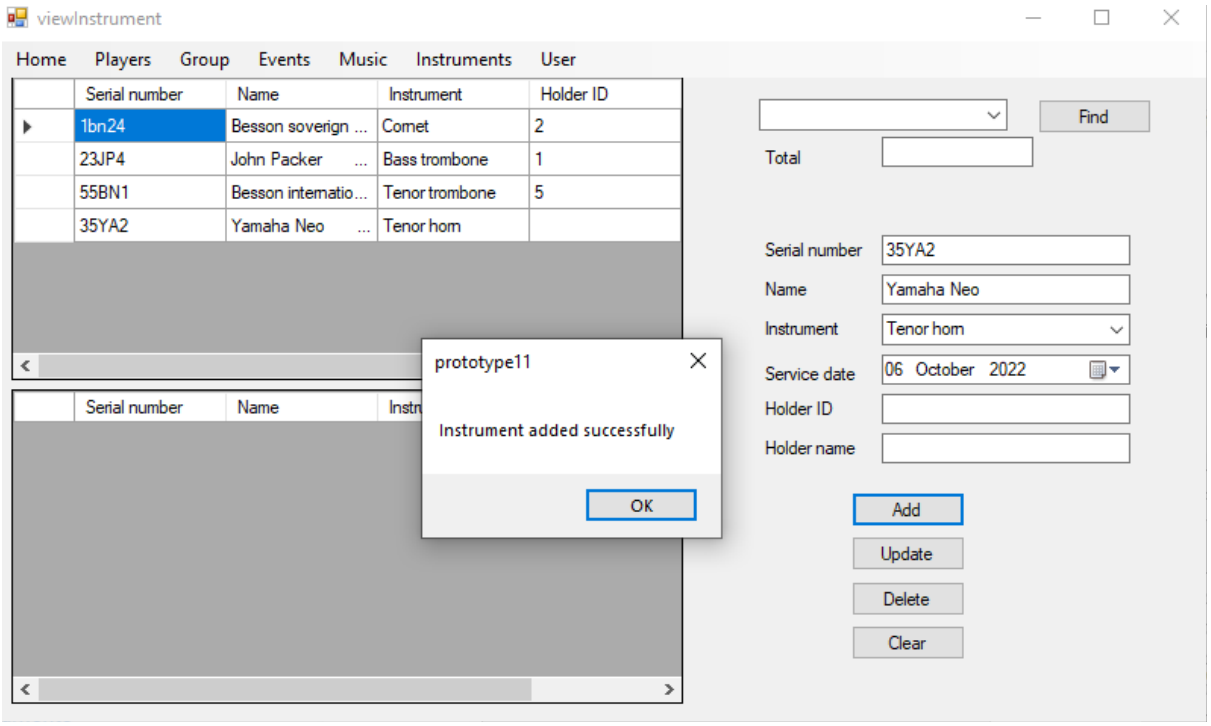
This screen allows the user to view, manage and search for available instruments.

When the screen loads, instruments stored in the system are shown in the DataGridView and all inputs are set to empty or their default values. The serial number, name, instrument holder ID and name are shown in the DataGridView. This is also what is shown when *clear* is clicked as it resets all inputs.

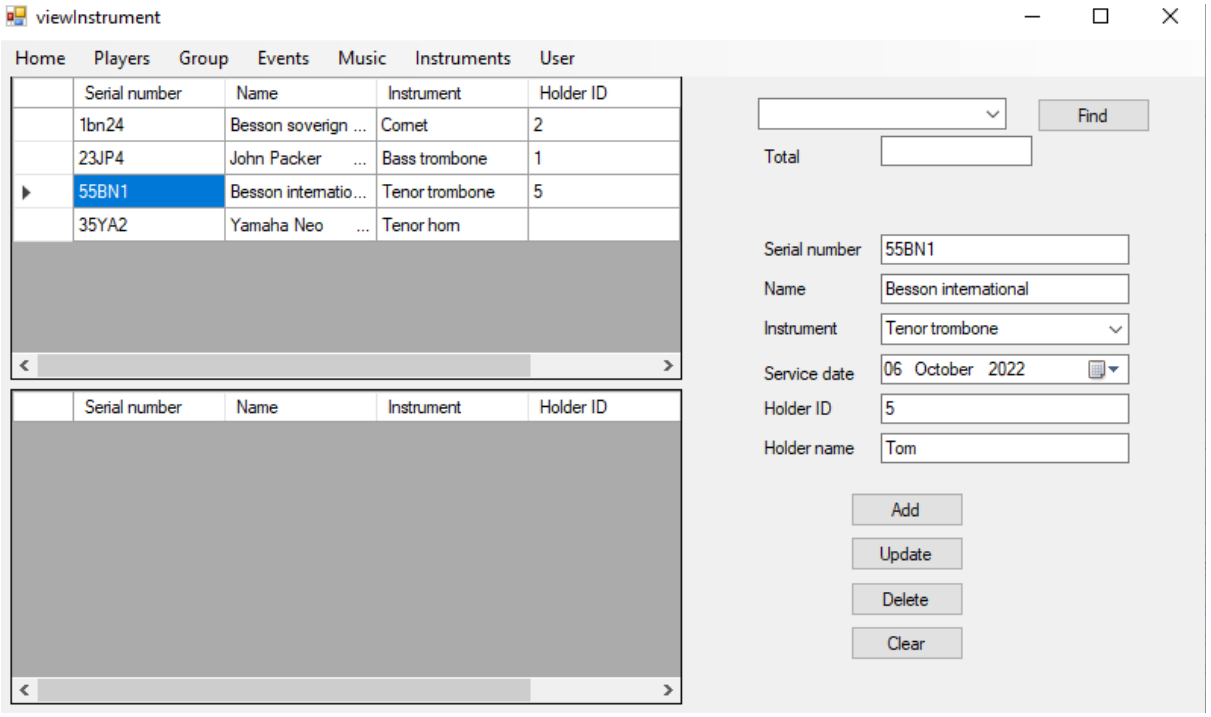


The user can enter the instrument's details and click *add* to store the instrument in the system. A message is displayed and the DataGridView is refreshed to show the new instrument.

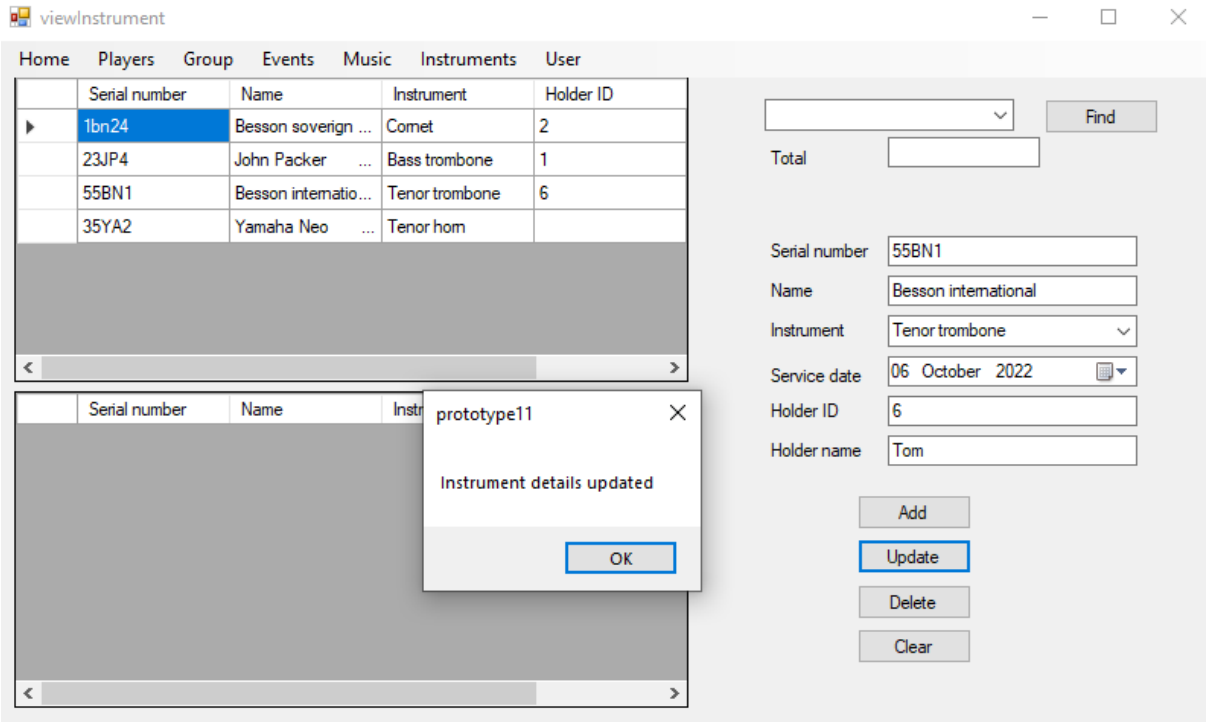
Holder ID and *Holder name* can be left blank as not all instruments will be in someone's possession.



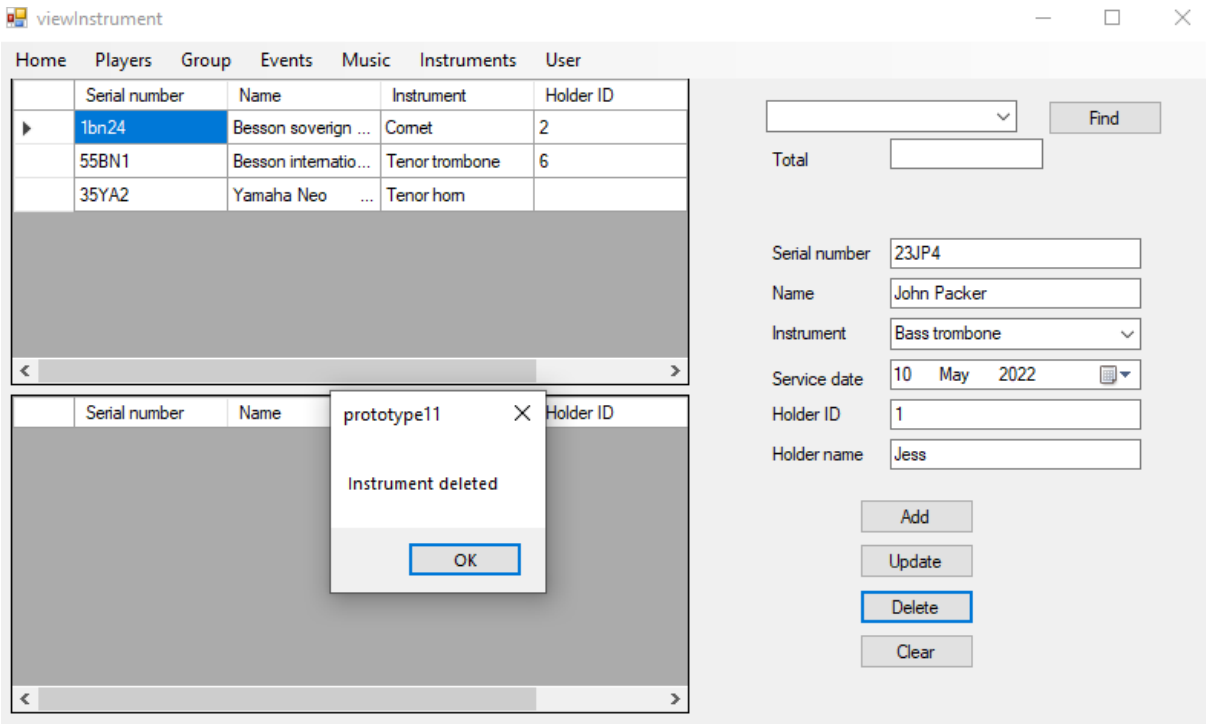
The user can see all details about an instrument by clicking a cell in the DataGridView.



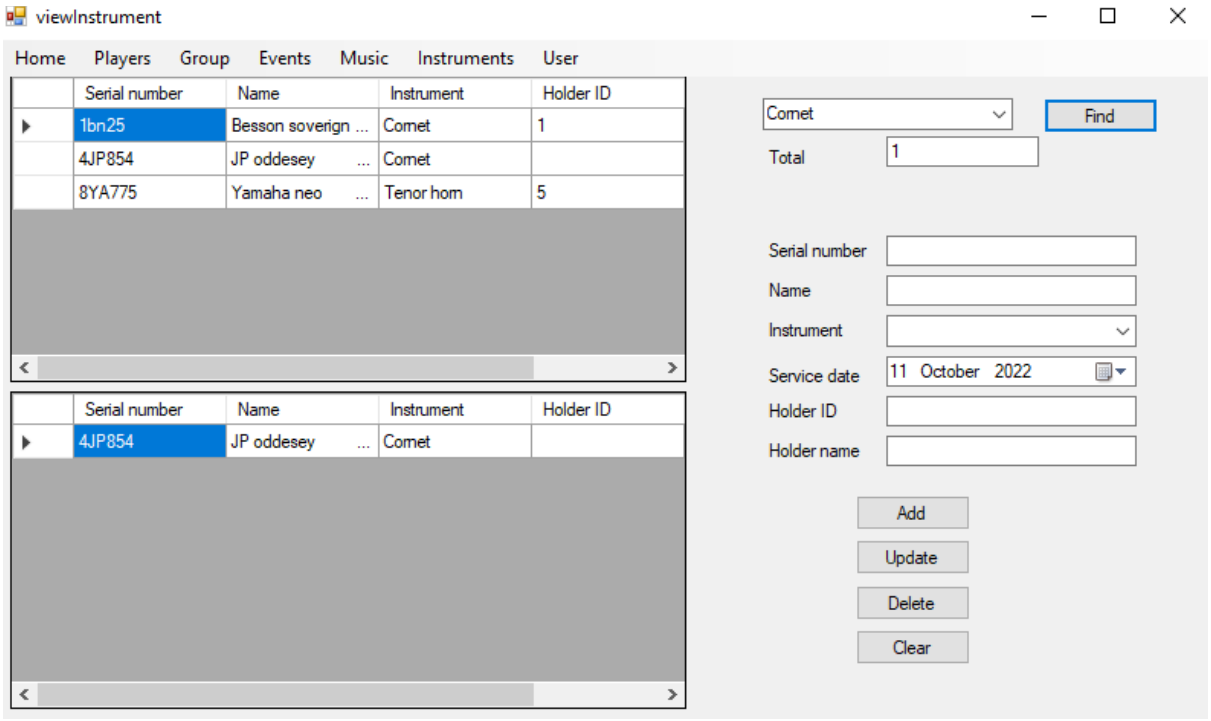
Once the instrument has been selected, the user can update the information about the instrument and save the changes by clicking *update*. A message is shown to confirm the changes have been saved successfully.

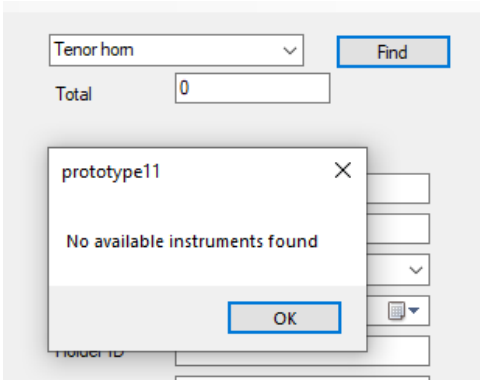


An instrument can be deleted from the system by selecting the instrument and clicking *delete*. A message is shown to show it has been removed from the database. If the user has made a mistake, the inputs remain filled in with the details of the selected instrument until another instrument or *clear* is clicked, so the user can add the instrument back into the system.



The user can find the number of available instruments by selecting an instrument and clicking *find*. If any instruments that do not have a holder ID stored are found, they will be shown in a DataGridView. If no available instruments are found, a message is displayed.





Evaluation of the prototype including justification of the good features of the prototype

The prototype contains the key features of the system that are required by the user. It allows the user to view and manage a selected group. They can make and view attendance for the group at each rehearsal. They can view all players stored in the system, and add, edit and delete players. The user can view all events and any event bookings on a selected date and can add, edit and delete bookings. Music stored in the system can be viewed and searched for and music can be added, edited and deleted. The user can view all instruments stored in the system and add, edit and delete them. They can search for available instruments to find how many instruments of the chosen type are available and view the details about these instruments.

The fact that the above features in the prototype are working correctly shows to the client that a working solution to the problem can be produced. It also shows how the user interface will look like, and allows the client to test the navigation across the system. The prototype has allowed the development of the user interface and the data processing, aiding the planning of the completion of the project, as time can be allocated to each aspect to test and validate.

The shortcomings of the prototype and suggestions for improvement

During the development of the prototype, there was limited testing on the features to ensure that they work as expected for all possible real-world situations. This results in a reduced amount of test data to get an accurate representation of the system. Further testing will be carried out before the final project is completed.

When developing the prototype, I did encounter some problems when testing that could become an issue in the final program. Any ID fields had to have consecutive values. If they don't, searching and displaying a record when a DataGridView is clicked fails. As all IDs must be entered manually by the user, this could create a large issue when the program is being used by the end user. To fix this, I could automatically enter the ID’s into the file and make the ID textboxes read-only so the user is unable to edit them.

The system also had problems handling empty combo boxes, producing an error when a blank combo box is trying to be saved. This is something that can easily occur when the program is used by the end user. This error happened because there is no validation in the prototype. In the final program, all inputs will be thoroughly validated to ensure no errors occur as a result of unexpected data inputs.

Currently, all data is sorted on screens using the built-in sort feature of the DataGridViews. This is only displayed to the user and is never stored in the system. This means that if a user sorts the data and closes the screen, they will have to sort it again if they need it sorted. I could include an option to sort the file in the final system, but this depends on the time I have remaining as this is a minute issue.