Programmed Solution to a Problem - Evaluation

Porth-y-waen Silver Band Management System

Nia Hawkins: 7183
The Maelor School: 68146

Contents

Evaluation

Evaluation of the effectiveness of the programming language and justification of the tools and techniques used

The programming language I chose to develop my project was Visual Basic. The language is easy to learn and use, making it ideal for the project as my programming knowledge is limited. The language allowed me to use global variables so data stored in one subroutine could be accessed in another. This language is also widely used, so I was able to access forums to help with any errors and to learn how to create features in the program that I was unsure how to approach.

The IDE of this language, Visual Studio, was ideal for this project as it allows the drag and drop of built-in controls onto a form, making creating the user interface quick and easy. When writing the code, automatic code completion and formatting made programming much faster. Keyword highlighting made it easier to read and understand the code and helped me to spot errors and debug. Automatic line numbering made it easier to debug as error messages include the line where the error occurred. Automatic syntax error detection also helped avoid syntax errors during compilation by checking for any syntax mistakes as they are made when writing the program and saved time as the errors were quicker to fix than if they were encountered after the program was run. The IDE included various debugging tools such as breakpoints and variable watches to allow me to work out why the code does not work as expected. I could step through the code and watch the variable values as the code is run line by line. It also has code optimisation, notifying me when a variable has been assigned but not used.

Tools and techniques used

To handle data, which was a key aspect of the system, I used Visual Basic's structures to store the data. They made it easier to manage data as it allows several pieces of related information to be stored together. Each aspect of the system has a file which stores the relevant data. Structures were created which allow each record to be read or written to the file in a structured format. I used FileSystem methods such as FileOpen, FileGet and FilePut to read and update files.

I used linear searches in many areas of the system, some of which were terminated when the search data was found, and many involved reading each record in the file, and outputting the record if it contained the search data. This also required the use of the structure that corresponds to the file that is being read. I included two bubble sorts in the program, which are not visible to the user as they are involved in displaying group attendance in a chart. They are used to ensure that when a number of rehearsals to show is inputted, the rehearsals shown are the most recent and outputted in chronological order. Sorts that the user can initiate were not included as the DataGridViews used on all screens have built-in sorts so they were not as necessary to write. This allowed me to spend my time developing different areas of the system.

While writing the code, I added comments to each block of code to aid the understanding and readability of the code. This will help someone else understand the code that I have written, and it also helped me know what the code does if there was an error.

I managed the changes made to my system using GitHub. This allowed me to keep multiple versions of my program using different branches. If a new change results in the program not working correctly, I could restore a previous version and compare the files to see what had changed. GitHub also made it easier to transfer my work between home and school. The commit records and pull requests also created a timeline so my progress on the project can be viewed and managed.

Comparing and contrasting the completed system with similar commercially available systems

In my desk-based research, I found systems that had features that I wanted to include in my system, as my final program is for a niche area so there are very few commercially available systems that are similar.

Zenefits

This program allows the user to add, edit and delete employees and their details. The system stores details such as job title, department and work location. These details allow the staff to be searched for, allowing a manager to find who would be suitable to put on a task.

I have included some of these features in my program. The user can view, add, update and delete players from the system. When players are displayed in the DataGridView, the columns can be sorted to arrange the players by each field. I did not include any searching for players that play a certain instrument, for example. This feature could be added in the future.

Another feature that could be included in my system would be Zenefits' payroll system for managing the costs of events. This was discussed in earlier development but due to the complexity of other features of the system, and the roles of the users, the feature was not included in the final program.

Further improvements using ideas from "Zenefits" could be to improve the user interface of this area of the system. "Zenefits" has a card-based interface that gives a preview of the employee and a picture of them. This makes their interface more user-friendly and more intuitive. This feature was not implemented due to the timeframe of the project and a lack of knowledge of how to create this type of interface.

Bookedin

“Bookedin” is a time management system that helps organise work meetings and events. It has a calendar-based interface that displays the week’s events to the user. It allows the company to enter any events and edit them.

My project allows users to enter events and view them in a DataGridView. They can also view the events on a date by selecting the data in the calendar and all events are shown in the DataGridView.

However, this system has a colour-coded calendar-based interface which I did not include due to the complexity and the timeframe to complete the project. In the future, I would update the interface for this area as a calendar interface would be more intuitive.

“Bookedin” allows the user to create multiple calendars so that bookings can be organised. I did not include this feature in my system as it is fairly complex and the timeframe of the project did not allow for it to be included, although I would include it in the future. However, when an event booking is made in my system, the groups that are required for the event are inputted so the user can see the events for each group this way. Another way to allow the user to view the events for each group would be to allow the search of event details.

“Bookedin” allows users to view all customer details so the customers can be contacted if needed. As discussed in the investigation, an interface similar to the one in “Bookedin” was not included in my final project. However, the user can view the customer’s details when they select an event in the DataGridView.

Bookedin also has a login page that also allows the user to change their password. This is an important feature in my system as this allows different users to have different levels of access to the system based on their role in the band. This is managed in the players screen, unlike Bookedin which has a separate screen allowing the user to manage logins. This could be a feature that could be included in the future as it allows the user to view all logins in one place, instead of having to click on each player in the players screen to view and update their role.

Follett Destiny Library Manager

Follett Destiny Library Manager is a book management system. It allows customers to search for a book and find out details about them. It also sorts books based on their genre. My program includes the ability to search and sort music stored in the music library, and also instruments stored. It allows the user to borrow books from the library. This feature could be included in the new system as the band does rent music out to other nearby bands. However, with the timescale of the project, I decided not to include this feature.

The system allows the librarian to add books to the record, which will be included in my system as I will include the ability to add new music to the record. The system also records the number of copies of a book that are in the library and marks each book as available or not. This feature is not in the system.

Successful features of the system

The system has many successful features, based on my success criteria.

- Users can log in to the system using their password. They can reset and change the password if necessary.
- Users have different levels of access to the system, determined by their role within the band, to ensure confidential data is protected.
- Player details can be viewed, added, updated, and deleted from the system. Players are also able to update their details.
- The players in each group can be viewed. Players can be added and removed from groups, and one player can be in many groups.
- The attendance for a rehearsal can be recorded for each player in each group, and attendance can be updated.
- Attendance can be viewed for a chosen number of dates stored in a chart
- ‘Deps’ are stored and can be viewed if extra players are needed for an event.
- The user can view, add, update and delete music, and they can search for the music in the database.
- Instruments can be viewed, added, updated and deleted. Instrument service dates are coloured to indicate how recently they have been serviced. All available instruments can be searched.
- Events can be added, updated and deleted from the system.
- The user can view all events and the events on a selected date.
- The user can record and update their availability for events and they can view all players that are available to play at an event.

Less successful areas of the system

Several aspects of the program could be improved, which can be grouped into the three categories detailed below.

Improvements to existing features

The validation of inputted data could be more thorough to ensure data is as reasonable and as acceptable as possible. This will reduce the chances of errors in the data and could prevent a runtime error from occurring.

I could implement a SQL database to store the data. I do not have any experience using SQL, and it could take longer to create the database. However, overall, it could be more efficient as it would be easier to search and sort data in the final stages of development.

Efficiency of the program

Certain areas of the system could be inefficient, especially the more complex features. The algorithm that processes and displays group attendance in a graph uses two bubble sorts to ensure that the most recent data is displayed in chronological order. With large amounts of data, this could become very slow. The algorithms could be written differently to be more efficient or a different sort could be used that would be more efficient with large amounts of data, such as quicksort.

I could also use more efficient search algorithms. I chose to use a linear search as the data may not be ordered and often strings were searched for. Many searches are carried out in the program and they could make the program slow with large amounts of data. The linear method of displaying data from files also uses a similar algorithm to searching which could become slow with large amounts of data. I could implement a hashing algorithm to allow direct access to the data, and using an SQL database could make searching easier to implement and faster.

Achievement of success criteria

The files, especially those that contain confidential information, such as the players database, could be encrypted to ensure that unauthorised users cannot view people’s contact details. However, many details were encrypted in the file due to the file writing method.

Evaluation of the strengths and weaknesses in the design and development of the system, including personal performance

I have written code that is well-structured and easy for another developer to understand. I used self-documenting identifiers so another person could understand what a variable stores. I also annotated my code to explain what each line or section of code does. This was also useful to me when reviewing and testing the code. As the project was developed over a year, certain aspects may not have been tested for several months and I sometimes forgot what the code did, especially in the more complex areas of the program.

I produced a video showing each of the features in the program. The video is more intuitive and easier for a user to know how to use the program and it illustrates what each screen does. The video also saved time as it was quicker to produce in comparison to writing a comprehensive user manual for the program.

Using the Waterfall methodology made it easier when planning the timeframe for completing the project, as it created a structured way to complete each piece of documentation required, in order. This allowed me to plan my time so I could ensure that the project was completed on time.

One significant weakness was my lack of knowledge and experience using Github. I had no prior experience using GitHub, although I had used a different version management system before. It took a significant amount of time to learn how to use the system properly, and several issues occurred when using it as I did not know how to use it properly, and many features have to be understood.

I also often found myself spending a significant amount of time fixing small features that did not necessarily need to be fixed at that time. I could have prioritised the more significant issues in the program so that documentation and testing could be completed. This also led to the project being completed very close to the deadline, as I discovered lots of code that still needed to be tested and fixed.

Despite the advantages of Waterfall methodology, it also had some limitations. As this project was sizable and fairly complex, I often found myself needing to change the designs of the program to create the most intuitive interface that also was not too complex to create. The linear approach made it very difficult to change any of the design documentation as it had already been completed beforehand. More experience creating similar systems or a similar commercially-available system may have reduced this issue.

Another weakness was my testing method. Despite testing that a feature was working before moving on to developing the next, I still encountered errors later on in development. This is discussed in the ‘Developmental testing’ and ‘Testing’ documentation. This slowed down the development as I was having to fix many features across the system at times to test the newest feature I had created. To reduce the chances of this happening in the future, I will use more thorough testing techniques. This could include developing unit testing scripts that would test every aspect of the system with every possible scenario that could occur.

Specific changes to the approach that would be adopted in the future to avoid problems

A significant issue encountered was the need to change documentation during the development of the system. To avoid this problem in the future, I could use Agile methodology as it allows any documentation to be amended at the end of each stage of development. It also allows testing to be done as the system is developed so any bugs are found sooner in the development process. I could also use the methodology to create a plan for the development of the system and have “sprints” to complete each feature for a certain date, ensuring that the project is completed within the timeframe set for the project.

Using Agile also would help solve another issue encountered; discovering significant errors and bugs in features that had already been tested during their development in the latter stages of the project. The Agile methodology allows for frequent testing at the end of each stage of development. I would also create a plan for the testing of the project, to make it easier to carry out testing as a feature is created, to avoid errors and make the completion of documentation quicker and less laborious and to ensure it is completed on time. I could also create unit tests that simulate user interaction with the system to ensure thorough testing.

To avoid any data loss and confusion using version management software I have not used before, I will ensure I know how to use the software before using it. I could also ensure that I only use methods of file management that I am comfortable using.

Final review

Overall, the final program was successful as most of the success criteria were met within the timeframe. The following improvements that could be added in the future are summarised below.

- Allow searching for a particular field in the players and groups screen, such as instrument, to make it easier to find players within the band that could play in an event.
- Improvements to the user interface:
    - A card-based interface for players, that shows some key details of all players and possibly a picture to make it more intuitive.
    - Scroll bars on larger screens to make navigation easier.
    - Reduce the number of comboboxes used to select views, such as on the group screen. Instead, a screen with buttons for each group could be shown, and when a button is clicked the applicable group view is displayed. This would make the interface easier to use.
    - A fullscreen calendar interface to view events to make the interface more intuitive.
    - Allow the user to create and manage separate calendars for each group's events.
- Create a separate screen for the management of customers for the events.
- Include a separate screen for committee members to manage player logins.
- Allow rented or loaned music to be managed.
- Use SQL instead of SystemFile methods to read and write files.
- Implement more efficient searches and sorts to improve the efficiency of the program.
- Allow the user to 'Switch user', which would log out the user but not close the program and show the login screen. This would allow another user to log in without having to open the system again.
- Allow the events coordinator and any other committee members to record a player's response for an event, if they are unable to use the system.
- Include help buttons on each screen to aid the user if they are unsure how to use the system.

Features in the designs and success criteria not included in the final program which could be implemented in the future:

- Calculate the price of hiring the band for an event.
    - Allow the committee to set costs for different aspects of the event.
    - Allow users to input the events requirements.
    - Output the total cost to hire the band.
    - Test that prices can be entered and the correct costs are calculated.
- Encrypt player data so it is secure and cannot be read by an unauthorised person.

Appendix - Discussion presentation slides

I made a presentation to my class on my project ideas to gather feedback. The presentation slides are shown below.

- Records members and any relevant information about them on paper.
- Attendance in each rehearsal recorded in a paper register
- Basic paper record of the music the band owns.
- Spreadsheet of instruments the band stores, which has been recently upgraded from a minimal paper record
- Can book the band by emailing or messaging the band using their website
- The events coordinator has to check a list of bookings for available dates and then inform the members of the band in the following rehearsal.

Broad aims

- Add and edit member information - to store contact details about each member and instrument and playing ability to be stored to ease looking for extra players for an event
- Add members to the group - create a register and record members' attendance to rehearsals
- Create a record of members who will play at an event
- Add and edit music information - store details about the piece such as number of parts and category and location
- Search the music library for sets of music and specific parts to determine whether new copies need to be bought.
- Find dates the band is available and enter and edit booking details
- Calculate the price of hiring the band for an event and calculate costs of an event and profits
- Easy to use and follows current band colours and formatting.

Limitations

- My ability
  - I do not have extensive coding knowledge.
  - Further training and research
  - Complex aspects may not be able to be included
- Time
  - Short timescale to complete the project
  - Time used completing training and research for functions I do not know who to code
  - Ensure key features of the system are functioning before adding any additional functionality.
- User ability
  - Intuitive interface and ensure navigation around the system is logical.
  - Instructions to guide the user
    Must enter the existing information so will take time before some features can be used fully

Limitations

- Store copies of music in the system
  - Photocopying some music could be illegal.
  - Not possible to scan parts so an external system would be required
  - Difficult to store and display a .pdf file in the program
- Notification of booking
  - May require emailing the contents of the booking which might not be possible with my current skills.
  - Might not include a booking form in the program for the customers

Limitations

- Computer access
  - Need access in rehearsals to mark the members as present.
  - Cannot afford to purchase and set up a computer in the rehearsal rooms that is powerful enough so it can run the system without crashing.
  - Ensure that the band has a way to use the system before it is created.
- Suitability of a computer system
  - Marking each member of the band present may also be tedious so it could be more efficient for the conductor to mark attendance on paper.
  - I may decide to not include this feature in the finished program.