# 8 Evaluation

## 8.1 Evaluate the system; Produce an evaluation of the programming language used to create the solution.

To develop my solution, the languages I used were Python and SQL. I choose Python for a number of reasons including my previous experience from practical lessons in class and through my GCSE coursework. I also chose python because of its simplicity when compared to other languages that I could have used to develop my system. Python was developed to be simple to read and write, it is therefore very like the English language, with keywords being easily understood and interpreted by even programmers with minimal previous experience with other languages.

I began to teach myself C# by subscribing to Team Treehouse, an online platform to learn many different languages. I could've potentially used this language along with WinForms to design the interface for my application. This combination of C# and WinForms would have allowed me to "drag-and-drop" widgets which would have made the implementation of my interface a lot quicker. However, as I did not have as much experience of the syntax for C# to implement an extended task like this, I did not want my inexperience to affect the development of my program. Most of my peers choose to use Python as well, so if we had any issues, we could help each other out.

I decided to use Tkinter, Python's GUI toolkit. I choose this package because Tkinter is Python's most popular GUI library, so there were plenty of resources for when I needed to figure out how to implement a new feature and I wasn't quite so sure on the syntax or simply trying to debug an error in my program. I knew developing the GUI using Tkinter would take longer than with C# using WinForms however I feel like I would have lost this gained time using WinForms when trying to write the logic of my program.

When I started to develop my program, I really struggled to implement my interface using Tkinter, this was my first time ever creating a GUI, all my previous programming experience was using the Command Line Interface. As documented in my developmental testing, I struggled with the layout of frames, and the ability to switch them to display different screens on action driven events ie. Button presses. Within Tkinter there were three different methods that could have been used to insert widgets onto frames, these are; .pack(), .place() and .grid(). The choice in the method used depends on how complicated you wanted interface to be. The .pack() method is the simplest however, this does not allow you to place every object exactly where you want it on your frame. You have the option to "pack" a widget left or right and it just pushes the widget to the left of your frame or right. I wanted to create a very intuitive interface, so this method wasn't going to be the one for me. So then I decided to use the .place() method. This method takes the coordinates of where you want to place the widget. I thought this would be the best option because this would have allowed me to exactly place the widgets where I wanted them to reflect my design. This method became an issue when I tried to line up buttons and labels on my frame as I wasn't sure exactly where coordinate was being taken from. By this I mean when I tried to line up two labels which held

different data, I wasn't able to exactly align them up below each other as the length of the text had changed which changed the x-coordinate in needed to use. I ended up using .grid() which allowed me to place all my widgets into rows and columns, This made development a lot easier as I was able to place a widget into a particular cell and then align the widget inside the cell ie. East or West.

I was able to effectively use many programming constructs within Python. These are included lists, two dimensional lists, dictionaries and tuples which were produced from the SQL query - of which I had no experience using before. Dictionaries allowed me to store allow the scores for each band in a grade and using the Band ID as the key and the score of the band as the Value, this construct ensured the correct score was being committed to the database for each band after the judge has entered their results and they had been checked for validity.

I also tried to limit the number of global variables I used, however I could not get away without using none. Global variables differ from local variables as these are variables which are made available to every sub-routine through a program. They should be avoided because they are very hard to debug if an error is thrown. They also limit the amount of code that is reusable and therefore very hard to maintain after development. I used global variables in instances where I could not pass them as parameters into a subroutine as they weren't necessarily accepted into every instance of the subroutine. I also used them to my advantage as I was able to make global the type of user logged in. This allowed my program display different widgets depending on the user. For example, in the judging competitions button, if the global variable admin was true then get the admin to select a judge to override, if admin was False then don't ask the judge to select a judge instead, get the global ID of the judge to be used in the subroutine.

In terms of file handing, I used the sqlite3 library for Python to save and retrieve data from my database, along with the DB browser application to view the state of my database. I found that it worked very well with Python. I never had a major issue where I couldn't save or retrieve any data from my database. I found the syntax also very easy to learn as again it is very much like English with keywords in block capitals, it also helped massively having to learn SQL commands for theory and it was good to put some theory into practice.

Within my program I made use of a few libraries. My system required me to produce an output of results from each competition. I began to use a library called ReportLab which generated PDF documents. I found this very difficult to use as again placing text on the page was through coordinates. I had the same issue where I couldn't line up the text. I then searched for a better solution and came across Python docx. This library produces word documents instead of PDFs. I found this library so much easier to use. The documentation that was given along side this library was so easy to follow and understand. They also gave an example which made it so much easier to adapt and change to suit my required needs.

In the early stages of my development and previous to my coursework, I had just been using IDLE as my integrated development environment (IDE). It was perfect for solving small tasks however, I soon realised that this was not the IDE to be developing a bigger system with as it lacked many developments and debugging tools.

A friend of mine had recommended using PyCharm, an IDE designed for Python. From when I began using it, my programming experience improved drastically. PyCharm had many nice features which made development so much more efficient and less stressful as it took less time to debug. A couple of features I loved for development include auto-completion of variable name, automatic syntax check where unrecognised syntax is underlined before compiling and my favourite, bracket matching. Bracket matching closes any set of brackets that you open without you having to. This is a very nice feature as I would occasionally forget to close a set of brackets and spend ages looking for the missing bracket. PyCharm also allowed me to set breakpoints and then step through to debug the logic of my system as this allowed me to view the state of every variable throughout execution.

I made use of GitHub for version control. This cloud-based service allowed me to work on my project from many different computers without having to physically hold all my files needed on a flash drive. GitHub was also very useful for following the changes I had made each time I developed as I kept a log of commits and updates. It also meant that if I tried to develop something and it didn't not work, I could very easily revert back to the previous stable version of my system.

## 8.2 Compare the solution with similar commercially available systems.

I believe that my solution has been developed to a high standard and if pushed out to the association, they would be able to use it effectively. Any commercially available system is built as so multiple users can be using the application at any given time. Before I could make my system commercialised, I feel like it would need adapt my solution in a way that means the database is hosted on the internet and each device using the system must be connected to the internet to access the application. This would allow each judge to log in and be judging the bands "simultaneously".

As my system is very specific to the needs of the RSPBANI, there are no similarly available systems on the market. As I don't have any commercially available systems, in my desk-based research I found a few examples of what data I would need to collect for certain features in my program and how to lay them out. I also have shown a few examples which I do not like and explained why I did not implement these in my solution. I found the desk-based research very hard, however I think it was solely down to the project I choose in the first instance as my chosen problem is so abstract.

Like all commercially available systems my system is able to send generic emails to users of my system when they carry out specific features. I really like how my system emails for instance a bands member when the Pipe Major of the band has edited their details or emailing the Pipe Major to confirm he has just entered a specific competition. I said in my desk-based research that I would like to email the bus company when the Pipe Major with their quote. I believe my system has been able to achieve more than in my desk-based research.

## 8.3Identify the successful features of the system and make specific suggestions for improving less successful areas of the system.

To evaluate the good features of my system I will colour code my objectives based my how successful I think they are and below I will talk about a few of the successfully features.

Green means that I think the feature has been successful

Red will mean I don't think it has been as successfully as I had initially intended

**Objectives:**

1. Appropriate layout, font and colour-scheme to follow associations colours
2. All users of the system on launch will be brought to the home page.
3. All users have the ability to view and search for a band on the bands page.
4. All users have the ability to view and search for a competition on the competitions page.
5. All users have the ability to view and search for a grades' result in a competition on the results page.
6. All users have the ability to login if they have an account.
7. All Account holders have the ability to reset their login credentials.
8. Different permission levels for band, judge and admin accounts once logged in.
9. Update and set password, security question and answer when accounts are logged in for the first time.
10. Band accounts will have the ability to update the Bands details & their band members details.
11. Bands will have the ability to enter and withdraw from competitions.
12. Bands have the option of booking transport.
13. Bands have the ability to register new players and transfer existing members.
14. Admins have the ability to carry out these band features above on behalf of the band.
15. Admins have the ability to add and remove band accounts, judge accounts and admin accounts. When a band account is created an email is sent to the Pipe Major to include the initial log in details.
16. Admins have the ability to create, update, delete and draw individual competitions.
17. Judges have the ability to write comments about each section and the score bands in competitions.
18. Admins then have the ability to collate the judges results and produce the results and send them to all bands which then updates the latest results feed on home page.

***Good features:***

As seen from above my solution contains many features which I am pleased with. I am satisfied with most of my features as I have been able to develop them both functionally and aesthetically. I love the way I have been able to develop my system so that only specific buttons are given to the user depending on their access rights. (Access rights depend on the table their credentials are found in). This means that all groups of users cannot see all the available functionality. This provided extra security as it does not tempt a user who doesn't have access to certain functionality to try and gain access to it.

I think basing my main form of navigation through all features using list boxes was a really nice. This idea worked very well for my system as is pretty much all cases, for the user to carry out any functionality they had to select a competition or a band. This use of list box I think will really improve the users experience – they make all functionality so simple to use.

When the user wished to edit band or members details, I love the ability for my program to display the same registration form however just populated with the current band or members details. This allows the user to easily see which fields they would like to change.

I also really like how my system is able to produce emails every time an important change is carried out. I think this is a great idea as this allows the band to keep track of when changes were made. This feature is very useful because admins also have the ability to edit and update band details on their behalf. So, when an admin updates information about a band, the Pipe Major of the band will receive confirmation of this change.

I am most impressed with the Collating results function. At first there doesn't seem like much is going on as there is only one click of a button, however the logic powering this feature had me rough drawing diagrams allowing me to see where I needed to get each piece of information from and then saving it back to the database and then producing the outputs.

***Short comings and improvements:***

I had wished to be able to allow the judge to select the section of the bands they are judging after selecting a competition. However, during development I was unable to implement this in this way due to time restraints of the project. Unfortunately, the judge has to select the the section each time they enter any results about a band. Of course, this is an issue as this can lead to human error causing issues if the judge selects the wrong section to judge and overwrites another judges score. This issue can be looked at after the initial release of the software and come as an improvement as it does not affect the stability of the system, it is just possible to reduce the chance of human error.

For the system to be used to its full potential, the system must be hosted on an online platform to allow multiple users to be logged in at the same time and constantly updating details and entering

scores. This would mean that the system may need to be tweaked a bit to allow for this and tested to ensure that the program can still function as required as mentioned above in the comparison of commercial systems.

If I had more time and knowledge, I would have implemented some way for bands or members for confirm that they agree with the changes that have been made about them before their new details are committed and updated, this will require another table of pending changes which if accepted are updated into the main file or discarded if decided by the data subject. This could be done through a link in the current email the members are already receiving. This would be a significant improvement as this will ensure no incorrect personal data is being stored.

Going along similar lines, when transferring members to a different band, I would have liked to implement a feature where by the receiving band must have the option to accept the transfer request. This ensures that members can not be transferred by mistake or incorrectly.

Another major shortcoming of my program is down to how data in list boxes is manipulated. I used list boxes as they are a very simple and intuitive way to allow the user to only select valid data from a list of possible data. This feature of my program makes it very easy to navigate and it also reduces the chances of the user entering invalid data and causing the program to crash. However, one major downside to this feature is that I must display the name of the band in the list box for the user but to use the competition in the logic of my system I must use the competition ID which is unique to the band and creates relationships in different tables within my program. This means that I can only retrieve the name of competition or band and I must query the database for the primary key. This means that I can not have any bands or competitions with the same name otherwise I will get an error as two bands or competitions will be query where theoretically there should only be one. To solve this program, it would have been nice if the language would have allowed the name of the competition or band to be displayed in the list box but on selection pass the ID to be used in my program. This would prevent me having to query the ID and means that multiple competitions could have the same name. This will become an issue when an admin creates the same competition for the next competitive season, where the name of the competition may be the same as the previous year. If I was able to get the list box ID functional then another improvement could be adding another screen to view previous competition results while still being able to enter and draw competitions for the currently competition season.

Currently my program is of a fixed size in terms of the aspect ratio that is of my computer (1280x800) As an improvement I would like to be able to expand/minimise my interface to suit the aspect ratio of the users monitor. This would mean that my program could be used on many different devices and there would be no compatibility issues where the users screen might be too small to fix the application. This would be very hard to implement as all my widgets and text would have to be scaled to ensure my program would still be displayed correctly and there were no formatting issues.

On the Bands page I did not display the bands details is a professional manor. I simply did not have enough screen space to be able to display all the information of each band. I am not pleased with this

feature and I would have liked to fix it however I felt this shortcoming does not affect the running of my solution so therefore it wasn't at the top of list of issues to fix. In the future I could potentially limit the number of characters that is allowed for each field at registration/updating. This would then allow me to keep the layout I have which would be the easiest solution. Although I think the most suited option would rely on a complete redesign of this page and possibly only shown keep important information of the band when searched for and allow the user to select a band to view all the information about the band on an additional page.

## 8.4 Describe the strengths and weaknesses of own performance in the design and development of the solution.

***Design and prototype:***

I am overall very pleased with the design of my system. I was able to design my whole system before development and know where I wanted to place everything on my interface. I knew which buttons I needed to include and to which user they must be made available. As a class we had to present our design section to each other. This was a great opportunity to gather their thoughts and opinions and they were able to point out a few issues which had not stood out to me as the developer. This allowed me to resolve these problems before I began to prototype my program.

One aspect of the design section I found challenging was the processing stages of each objective. I struggled to effectively create my pseudocode to solve the problems and at this stage I had very little experience using Tkinter, so I wasn't exactly sure how to implement widgets at this point.

My prototype in my opinion was over developed in many features and underdeveloped in others. I feel this may have been down to the fact that I was unsure with Tkinter at the beginning so as I got small things to work, I decided to continue on, and I ended up with some features fully developed and others not at all. I would have to admit I think I spent too much time on developing my prototype however looking back it might have worked for the better. It definitely allowed for a quicker development as I had gained a lot of experience using a GUI and using widgets.

***Time management:***

Throughout the whole project, time constraints had always been in the back of my mind. At the later stages of the coursework I felt very pushed for time struggling to meet deadlines. At the beginning, during the discussion, investigation and design phase my time management was very poor, I spent too long on these sections. I found myself beginning to complete these long sections with not enough time until the final deadline.

***Overcoming errors:***

During development, I had many problems which I had to diagnose and debug. The complexity of these bugs and then the time taken to find a fix for them depended on whether they were logical or syntactic errors. I have found that logical errors require a lot more thought and are much harder to fix than syntax errors. Many logical errors that I had dry run the logic on a piece of paper and draw diagrams if necessary. Syntax errors may have been easier to fix however if I did not know how to fix them, I had to use websites such as W3Schools or Stack Overflow. These websites were very good for finding and solving syntactic bugs as other programmers will have came across them as well so once I found what I was looking for I was very quickly able to resolve my own problem. As I gain more programming experience, I hope to rely less on web resources like this.

## 8.5Describe changes of approach that would be adopted to solve a similar problem.

If I were to solve a similar problem again. I would re-consider using Tkinter to develop the GUI. Instead of using Tkinter and Python, I would use a more popular language like C or Java#. There are many advantages of using C# over Python. C# is an object orientated programming language. Developing a similar project using OOP instead of procedural programming has many advantages in itself. Using OOP allows for the creation of objects which inherit attributes from a class. This structure would have allowed me to re-use more code more efficiently which would decrease the developmental time. This re-useable code means that less debugging will need to be done and the code will be easier to maintain in the future.

In terms of development, in the future I will create the main functionality without the log in feature first. I realised during my prototype that because I had implemented my log in functionality, every time I wanted to test or dry run my system - I had to log in. This increased my development time as it took much longer to get to the page I wanted to test and got very frustrating at times when you kept getting the credentials wrong. I found it very to force myself to write annotation on my code as I went along because I got so into programming I couldn't stop until I had implemented the feature, this meant sometimes when I went back to write my annotations, I found it difficult to understand which I am actually doing in certain parts of my code.

My balsamiq designs really helped me with the layout and feel of my application. However I feel like it would have been a lot easier to develop my solution using individual flow chart diagrams for each feature as I would  have been able to see and layout my system on paper. I found pseudocode very hard to write in the design section as I wasn't sure exactly how I was going to end up developing my solution. In the future I must create and design less complex and more understandable pseudocode in my design section as did not end up following fully in my development.

If I were to start the write up again, from the beginning I would ensure I have thoroughly planned what I was going to write in each section and limit myself to how many hours I should put into each. This would allow me to spend more time on the latter sections. The testing section of this, was

the most time consuming. If I had planned this section by creating my test plan as I developed functionality, I could have saved some time when each feature was fresh in my head. Doing this would have allowed me to go straight into testing instead of having to create my test plan for each objective from scratch before testing it. I also think creating the test plan after development makes it a lot easier to forget and miss a few tests in the test plan.

In conclusion, I really enjoyed developing my solution as part of this coursework piece. I have successfully completed the task I set out to solve and have been able to meet all my objectives, most of which to a commercial standard. I think this coursework has been invaluable to learning in depth how to program an extended task and the skills that are required in many fields within the software development sector, from systems analysis, software developing and testing, all which are vital to ensure the success of a project.