Candidate Number:
3097

Evaluation:

9.1: Evaluation

I will be giving an evaluation of all the sections I have completed throughout this program as well as some additional areas. These sections will be:

- Discussion
- Investigation
- Design
- Prototype
- Post-prototype
- Software development
- Developmental testing
- Testing
- Effectiveness of programming languages used
- Successes, shortcomings & potential areas for improvement
- My own strengths and weaknesses
- Comparison to success criteria
- Comparison to commercially available systems
- Changes that could be made to avoid problems experienced
- Overall

I hope that this will be a suitable look at the program I developed in addition to the development process of the program. This also looks at the main issues I had with the program and how I would act differently if I approached the project again for a second time. The evaluation will also show how the project met the overall specification that I originally laid out. I decided to analyze each section separately as each evaluation would be more specific, well laid out and in-depth.

9.2: Discussion

This was the first section of my project. I was contacted by a clinic who asked me to develop a database for them that would replace their old paper-based system. Initially I thought that the project wouldn't have enough scope with the clinic but after learning more about them I learned that there was a lot available.

Due to the large scope I decided to use the clinic as the company for my project, the clinic asked me to try and rehaul their current system and fix it up. Initially I came in and did a presentation to the staff at the clinic and left sheets of paper, so staff members could reply. I felt like this presentation was a success and I successfully managed to get a range of feedback about what the staff want in their project. These replies were vital and extremely helpful and helped me much better understand how the program should be developed?

I then laid out some rough aims based on the response of the presentation and finally I looked at the possible limitations of my program. The discussion overall was really important in understanding how the project would eventually looked and what the main focus of the staff of the clinic was.

9.3: Investigation

I then moved on from the discussion to my investigation. This is where the system was starting to shape, and I could gain a much more realistic understanding of what my solution would eventually end up like. I did a variety of sections within my Investigation to gain an analysis of what people were looking for with the program, how the program should end up and the possible shortcoming that may occur later on.

The first of these four investigation sections were the interview with the owner of the clinic, Mrs. Frener. This was a structured and in-depth interview and allowed me to have a significant insight of what their current system is like and what improvements they are aiming to look for in the project. I found this to be the best of the four methods as it allowed me to have a personalized look of the system from a person who is very knowledgeable about it. I found this to be very helpful in seeing the main issues that the clinic had with their own system and to find out what I should focus on in my programmed solution. I was allowed to see the specific ways that their paper-based system functioned and the exact shortcomings of each. This was an essential part of my interview and I found this section very vital.

I then sent out a questionnaire to some of the staff members at the clinic, these were questions which asked for agreeability on parts of their current system. This was helpful in finding out the general consensus on what most of the staff members feel about their current system and what they feel is most important about it. I also created graphs in my program to see the trends of opinions by most of the staff at the clinic. This was an important stage of the investigation as it allowed me to understand the opinions of the majority of the staff using it instead of just the owner of the clinic and allowed me to cater towards the issues of the staff members at the clinic more. The third of these stages was a visit to the clinic. This allowed me to see the issues with their system in real life terms. Although this was useful, I felt that my previous investigation sections already covered a variety of the issues with the program, so I didn't really gain much more new information about what I should look for in the system. This also has the possibility of staff members acting differently knowing that they were being observed which may make any conclusions in this section inaccurate.

The forth, and final stage was looking at existing desk-based solutions. I looked at existing examples of databases from clinics to understand how to structure my final program more effectively. This was useful as it gave me the idea to keep everything within the same frame and it also helped me make it so navigation throughout the program was much easier. I also looked at some data flow diagrams to see the information that existing clinic held so I could see which tables and columns may be required in my project in addition to the links of the data. I found this to be an effective section of my investigation.

After I completed all the sections of my investigation I then created the requirements of stakeholders, both functional and non-functional in addition to a success criteria to better see my aims within the project. The functional requirements were useful in creating the features that I needed in my program. It allowed me to better gauge how the final program would look. I also created non-function requirements to see my other aims within the program. I then chose the methods that I would like to use within the program which were useful in seeing which modules I would use to create my final solution. Finally, I created a success criteria on what I aim to meet with my final project. These were specific to each of the tables and the overall program and I used this success criteria when developing the program and attempted to tick off each requirement.

9.4: Design

The design stage was one of the most important parts of my project. It was vital in seeing the exact design of my system in both interface and functionality. The design allowed me to create the base of the system that I was making and was a foundation on all sections after. I believe that without this section my project wouldn't have been as polished or complete and if I hadn't made one the development process would have been much more difficult.

Candidate Number:
3097

I started my design by setting out the modules that I was aiming to create, I decided what features I aimed to have within each table and frame. This allowed me to split up all the aims I had with my program instead of having a long and complicated list and also gave me pointers on how all the pages are supposed to stick together.

Working with the modules, I created a menu diagram on how the program is stitched together. This would show you the navigation throughout the program in a simple and visual way, this was effective as it was much easier to understand than text. I created an ERD to see how the entities within the program relate and so it would much significantly easier to normalize the database. It also showed me how each of the tables connect with each other. I then created a data dictionary for all the tables within my program, this allowed me to see what data I'm storing in each table, and the data type and validation of each. I found this extremely helpful when developing my program as it made the database creation in SQL much more fluid and easier. Below the data dictionary I also showed the methods of access I would use to connect the database to the GUI and some brief lines on how the table was normalized based on the information given previously.

Data flow diagrams were created with an online diagram creator and I created Level 0 diagrams for the whole program and the login page and Level 1 DFD's for each of the separate tables. This allowed me to easily visually understand the flow of data within my program and how the data requested or sent from external entities effected the overall database. I did however find that the data flow diagrams were quite difficult to initially make though I do feel it is worth it and will also show any future developers' important information about the program that I developed. I created screens of the program within Photoshop. I used these, so I could see roughly what the GUI will eventually look like when I make the program. Looking back at it they were very close to the eventual program and I used the diagrams a lot when developing my system. I feel like the GUIs that I developed are both intuitive and aesthetically pleasing. In addition to creating the GUI I also designed my aims for the look of the text outputs. This was quite easy to replicate as the text output screens were designed in Notepad.

The last section that I created in my design was the pseudocode. This was a way of seeing the logic I aim to use when developing the program and in retrospect it was very close to what the code within the final project would eventually look like. It also allowed me to much better understand how I would manipulate the data in the program and connect it with the database and the methods that I would use to validate and output data.

Overall, I feel that the design was an extremely important part of my overall project and I believe that it matched very closely to the finished program.

9.5: Prototype

In this stage I had to create a rough program of what the eventual program would look like. When thinking about what I would contain in my prototype I felt that basic features like adding, searching and deleting records were vital in addition to different types of outputs. The outputs I used for the prototype being text and treeview. I also decided to use two tables to see how multiple tables would work together in the program and views with a suitable login page to see if the methods of views that I had planned would work effectively. I omitted the remaining tables, validation and some functionality like editing a record as I felt that the prototype that already existed would be a good enough look for what I aim to create without taking a significant amount of time to develop.

Initially I created the database of the program, I found this quite easy as I already had experience using SQL after doing GCSE OCR Coursework. I then added records to each of these tables, so I could see what the program would look like with dummy data. After this I created the login page, I had no experience with tkinter prior to this coursework so this was a significant learning experience on how it worked. I looked up a variety or resources online on tkinter. I created a page with two frames with one on top of the other, when the login button was pressed it would delete the top from and then it would create suitable tabs. I'm really proud to have achieved this and it took a lot of trial and error to make it work. Since I knew SQL already, I found that the functionality behind logging in to be quite simple to program. I then connected the tabs to other python files, I did this, so all the code would be easier to adjust instead of being in one long file. I created the frame, with treeview, which I am happy to have selected to display the data and added buttons for each.

At the end of my prototype I did a summary of everything and decided what improvements that I aim to make in the future. Overall, I am pleased with my prototype and this section.

9.6: Post-prototype

This was one of the smaller areas in the project however I did find it quite significant, it gave me a good insight on what most users thought about my program and it helped me design the future plans I had for my software development.

I started my post-prototype by creating a questionnaire, this was sent out by email to 20 random users along with the prototype. I then put all these results into a table to better analyze them. I found this section of the post-prototype to be very useful as it gave me a good insight on what users thought about my program and where I should focus on from then on, I also added a comment section at the end of the question are which showed me what most of the users thought about my program and what the main advantages and disadvantages were in it.

After I completed the questionnaire I created all the tables I planned to add, I used my tables in the Design stage to help create this. This was useful in seeing what information I should add to my program in the future. Below this I also added all validation I planned on adding to the final program along with all the pseudo code required for each method of this validation, I believe that the range of validation that I had in my program was large and I am extremely happy with its overall result.

In conclusion, the post-prototype was a vital part of my program and it helped me much better understand what most people felt the prototype was like in addition to seeing any necessary information I should add in the future and any fixes or features I should focus on in the final product.

9.7: Software development

After the post-prototype I moved onto the final area of developing the program, the software development. This was the largest and most difficult stage in the project. I did however find this part of the project to be most enjoyable. Due to the base of my prototype however, this was significantly easier than I expected as I already had the backbone to my program.

The most difficult part of the program for me was tkinter, I found it to be a quite confusing language and the format of it to be messy and strange, however when I got the base of most of the code done the remaining tkinter was relatively simple. I am most proud of my choice of using a treeview with tkinter, a treeview is a way to display data in it and its format is very much like one you would find in a table or a database which is very helpful for me. In addition to being aesthetically pleasing, it is easy to use, allows the user to highlight specific records and allows the user to change the size of columns. This was vital in making the program especially strong.

I then connected the treeview with all the buttons, such as adding records, when the records were added it would refresh the table along with any other updates to data within SQL. I also created a wide range of outputs, with 12 in total. 5 being in treeview, 6 being in text and the last one being a popup for a user's BMI. I feel like all the inputs, processes and outputs that I used in the program was highly effective. I am also very proud of my inputs, especially deleting a client where you could simply highlight a row and press the delete button. I am very proud of how the navigation through the program works and how the majority of it is in one frame, except for adding and editing records which I made as a popup for design reasons. Each table is separated by tab, and the tab frame is on the same window as the login one. I am also happy about how simply tabs are created in the program.

I create a very large array of validation, it included all typical types in addition to some that were quite complex, such as a format check would check all possible types of postcodes that can be entered in the UK. I used regedit and a lot of online resources to create these format checks. I also created checks for length, range, type, presence and attempting to break the program. Views were implemented effectively within the program as well, I had one login page which would find the position you are in the company and then show you suitable tables and controls of each in addition to not allowing you in if you get the username or password wrong. I found these to be a highly effective method of data security and it suitably keeps the system access limited to the right users.

Overall, I am very proud with my software development. I feel that I have exceed the expectations I had when I originally started developing it and I am happy it meets all of the features that I was originally required to do.

9.8: Developmental testing

This was one of the smaller sections in my documentation. I did however find this stage very useful in finding out where the problems with code was and what suitable methods I could take to work on them and improve them. It forced me to think logically about where the error was and how to fix it. It also helped me notice any recurring errors that I had during the remaining development process of the program. Although this section was short it took a significant amount of time to complete as errors are often difficult to find and solve.

Overall, this was highly effective in any debugging throughout my program and allowed me to approach any errors that I came across in my code from another angle. As a result, it made the software development significantly easier.

9.9: Testing

The testing for my program took a significant amount of time. I did a large of variety of tests throughout my program. I first tested my program for valid, invalid and extreme data in addition to some GUI tests. And then finished off with acceptance and end user testing. I am proud with my testing and for the significant majority of the tests I found them to be successful, this shows that although my code doesn't contain zero bugs, my entire program is very limited with them and is stable. It also shows the variety of validation that I created during software development.

I started my testing by doing white box testing, I looked at the purpose, description, data entered, expected result, result and the test type when testing my data and did tests using valid data, invalid data, extreme data and GUI. After creating a table for all of the tests I attached evidence at the end of it with an explanation for each and how it functioned. I am proud with the tests that I have completed, and I believe I did a good variety of tests for my project, I also believe that I provided a suitable amount of explanation about everything I was testing and that the program has a very limited number of bugs and errors throughout it and can spot the majority of invalid data within it. Afterwards, I moved onto end-user testing. I believe that I received a good amount of feedback and that the general thoughts on my program were good. I managed to find the main criticisms about my program which is quite helpful for anything that I plan on developing in the future and what the main weakness in my program are.

I then gave my program to the owner of the company through acceptance testing, the comments were very good, and she spoke highly of the program. She told me that the program is a 'massive upgrade over our previous system' which I am very pleased about. I was told what she found to be the best parts of the program in addition to any criticism that she had when using the program. At the end of testing I created a list of all possible improvements that I could use in my program. I feel that this is a list, that if in hindsight I made the program again would be much smaller. I am very happy to have learnt from this experience.

All in all, I think that my testing was very strong, and I am very happy with the amount of things that I tested in addition to the feedback that I received both from the staff and the owner. I also feel that this section has provided me with suitable feedback that would help me approach the solution differently in the future.

9.10 Effectiveness of programming languages used

I chose Python as my programming language for my coursework. I decided to use Python because of its simplicity and its range of modules and I don't regret choosing it as a language. Python is a very simple language to understand and program with. It significantly helped my program to be full of self-documenting identifiers and I also found that Python was very effective in debugging and telling you where your error is and what the error is. Python also has access to a very large array of modules. Most notably tkinter and SQLite3. Without these modules the program wouldn't have reached anywhere near the level of sophistication it would have and it was a very vital part in programming my coursework. I also chose Python due to my experience with it, having used it as part of my A453 OCR Computer Science coursework I have become quite strong in the language and it helped me know what I was doing much more quickly than I would with other languages that I am less experienced with. I am very proud to have chosen Python as the language for my database and if given the opportunity I would do it again.

Candidate Number:
3097

Being able to use SQL was a major strength of Python, SQL is a module in Python that you can use to create databases and is also internationally used. I choose SQL over other methods of storing data for a variety of reasons, the first reason was the easy access to it, SQL is in built in python so no modules would have to be externally installed, I also had a large amount of experience using SQL as I used it during my OCR coursework and thus I am literate in the language, it is also a very simple language to understand and read and is very similar to pseudo code. This means that people who are not familiar with Python will likely still be able to understand what is going on in my program. SQL allows me to easily set data types, create relationships between tables. It also allows me to very easily create, edit, search for and update records. As these were the functions I needed I believed that SQL was the ideal choice for a database and I stand by that.

After I chose my database, I also chose tkinter. The biggest problem with Python was its lack of good GUI modules. Although I believe tkinter is the best of the available GUIs for Python, I think that the GUIs available in other languages are significantly better and much easier to use. However, if I wanted to take this approach it would require me to learn an entire new language and thus wasn't worth it. Tkinter allowed me to visually show users the program instead of a CLI. The biggest difficult I had with tkinter was moving items around the frame, not the actual functionality behind it which I managed to figure out fairly quickly. I used treeview to display data within tkinter which I feel is the best way to display data to the user based on the possibilities within tkinter. I found this to be the most significant issue and downside with Python.

Another really good reason that I chose to use Python was its popularity around the world, being fluent in Python is extremely helpful and is one of the major programming languages, the language is one required by lots of major companies throughout the world which is extremely beneficial. Learning more python will significantly advance possible future employability which is a massive plus.

Overall, I am proud to have chosen Python at its modules SQL and tkinter for my project. I still believe that this was the best choice given my circumstances and the resources that I had available.

Candidate Number:
3097

9.11: Successes, shortcomings & potential areas of improvements

Successes

I believe that my overall project was very successful, and I am very proud with the overall result, I am very pleased with everything that I created especially given the timeframe of the project. I am happy with all the features I created and the overall look of the program.

Firstly, I am very happy that nearly the entire program is within the same window. I believe it was a very good design choice to this. As it took a considerable amount of effort to make it, so tabs are created after the frame has already been created and putting a frame on top and deleting it to change to a new one. But overall, I am very happy with what I managed to create with overall navigation and I'm happy that all the tables can be accessed by simply switching between tabs. I believe the validation techniques that I used throughout my program were highly effective. I am especially happy with the format checks I used, the postcode format check being the one I am most proud of as it doesn't just check for the types of postcodes in my area, but all possible postcodes in the UK. I also created strong format checks for date and time that check not only the format, but also make it so, for example days can't be 31 for months that don't have 31 days. I am also proud with the wide range of type checks, lengths checks and presence checks through the program. I have also created validation that prevent breaking the program, for example attempting to editing records when no records have been created. I am also proud with all the outputs available throughout the software. I have a total of twelve outputs in it. And these are different types of outputs, from text, to treeview and alerts. Some of the outputs also calculate things such as total profit or BMI instead of just displaying the data searched for. I am pleased with how I displayed the data within Python. I think using a treeview was a very good choice for displaying data and I feel that the data looks like it is held very clean and orderly which I am very happy about. In addition, I am happy that my database has been normalized to third normal form. The structure of my database is very strong and clean and the data redundancy throughout is very minimal. I am proud that I managed to look at my data structures prior to creating the database and normalize it to third normal form. Finally, I am pleased with the lack of bugs throughout the program, my extensive testing while creating it managed to minimize bugs significantly and took a lot of work but in the end my program works very effectively, very quickly and contains a lack of problems throughout.

Shortcomings

There were however a few shortcoming throughout my program.

Firstly, the database isn't automatically backed up. This could result in some security issues as if the database file is damaged it could result in loss of data. You can however, manually backup the data by copying and pasting and this is why I didn't attempt to use the time I had allocated to this project on backing it up. The programs GUI is also quite bland, and I feel I could add some colours to it to make it more aesthetically pleasing and more attractive to the user. I believe that if I had more time to focus on the looks this would have been done but I aimed to sort out the functionality of the program before the GUI. Finally, the data isn't encrypted, and it is stored in plain text. As a result, if someone got access to the database they would be able to access all the data as it isn't encrypted. Given more time, this would be the first thing that I would focus on.

Potential areas of improvements

There are a number of areas in my program that have potential areas for improvement, some of which have been briefly mentioned in the shortcomings.

Firstly, I think there is lots of room for improvement for the GUI. I could add some colour to the program to make it look less bland, I could add a scrollbar to the table and I could also get rid of some of the issues that occurred with the GUI, for example being able to stretch the records out of the view. In addition to the previous, I could vastly improve the outputs that go out of the program and put the outputs into a HTML file or into a PDF in an aesthetically pleasing format. However, as far as the program goes the GUI shouldn't have much priority and the functionality and security of the program should be much more important during the development process. I could also take a number of measures to improve data security. I could first encrypt all the data in the program. This would be done to prevent any hacker who has access to the database file to access any data in the table easily. I could also make it, so the database is backed up automatically and stored on a cloud service like Google Drive with a date for each backup for the file name. This would significantly reduce the risk of any data loss for the clinic. In addition to what I have already stated, I could normalize my database further, to 6^{th} normalized form. Although my program is already normalized to 3^{rd} normal form, 6^{th} normalized form would further reduce data redundancy in the program which would make the entire program much quicker. I could also add more tables for new types of data. In addition to normalizing the table, I could link the table to a server to the data can be updated automatically on another computer when data is manipulated on one.

Overall, I believe that there are a number of steps that I could take to improve the program, however I do believe these improvements would be unrealistic given the timeframe for the development.

9.12: My own strengths and weaknesses

Strengths

I think the biggest strength of mine during the development of this project was my code, I used a wide range of techniques and methods to bring it to a high standard. I am very happy with what I have created with my code. The code strongly improved the other parts of my project and I feel it is the strongest part. I am especially happy with how fluid and intuitive the program is and how similar it is to other commercially available systems today. Another strength I had was my ability to understand what to expect before doing the code, I believe it is evident that I properly assessed the program beforehand and understood what I would be able to program under the time constraints without over or under aiming and as a result have a piece of code that meets all aspects of the success criteria and broad aims. I think that my scope of the project that I did was the best and most realistic that it could be. The final notable strength I had was how I laid out the design stage and how it later transferred over to the software development section. If you compare both the design and software development section it is clear a lot of the choices I made throughout it was inspired by the Design and very similar to the overall project.

Weaknesses

I feel like one of my main weaknesses was my writeup, I feel like a lot of my work was quite jumbled together and disordered which could make it harder for other readers to understand what I was saying. I think in the future I should lay everything out in a significantly clearer format. Another weakness that I had was the limits on the type of testing, during the testing stage I omitted doing black-box testing, alpha testing and beta testing. In retrospect I feel that I should have done these during the creation of the program, so I could more significantly understand what the strengths and weakness of the program that I had created was and what I should have done to improve them.

9.13: Comparison to the success criteria

Clients

The program must allow the user to add a client to the clients table if all information added is valid – the program allows user to add a client by clicking the add client button on the client table and then filling out the client form with appropriate data

The program must validate all data inputted into the clients table – a number of validation checks are used throughout the program to validate that all data entered in the add or edit client sections is correct

The program must list all clients in a table – all clients are outputted in the table in the client tab

The program must allow the user to search for a clients of a specific keyword in a specific column and output it into the program and into a text file – after searching for a specific

keyword and pressing search the program will output all the search results in a file called ‘clients.txt’, in addition to this it updates the result in the treeview

The program must allow a user to add a client and create a medical record connected to that client’s primary key at the same time – when adding a client, the program creates another record in SQL that creates a connected medical record for that client

The program must allow a user to edit a client - if you press edit client and input suitable data in the form the program allows you to edit a client

The program must allow a user to delete a client – if you highlight a record and press delete selected client the client will be deleted from both the treeview and the SQL file

The program must list all client information in a clear fashion – all clients in the program are listed in an ordered and clean fashion within the treeview with nothing crossing over into other columns

Medical records

The program must validate all data inputted into the medical records table – a number of validation checks are used throughout the program to validate that all data entered in the or edit medical record is correct

The program must list all medical records in a table – all medical records are outputted in the table in the medical record tab

The program must allow the user to search for a medical records of a specific keyword in a specific column and output it into the program and into a text file – after searching for a specific keyword and pressing search the program will output all the search results in a file called ‘medicalrecords.txt’, in addition to this it updates the result in the treeview

The program must automatically create a medical record when a respective client is added and connect to that client – when a client is added to the program a medical record is automatically created that connects to this

The program must allow a user to edit a medical record - if you press edit medical record and input suitable data in the form the program allows you to edit a medical record

The program must allow a user to delete a medical record – if you highlight a record and press delete selected medical record the medical record will be deleted from both the treeview and the SQL file

The program must list all medical record information in a clear fashion – all medical records in the program are listed in an ordered and clean fashion within the treeview with nothing crossing over into other columns

The program must allow a user to calculate the BMI of a selected medical record – when you highlight a record and press calculate BMI an alert will pop up stating that medical records BMI

Candidate Number:
3097

Appointments

The program must allow the user to add an appointment to the appointments table if all information added is valid – the program allows user to add appointment by clicking the add appointment button on the appointment table and then filling out the appointment form with appropriate data

The program must validate all data inputted into the appointments table – a number of validation checks are used throughout the program to validate that all data entered in the add appointments section is correct

The program must list all appointments in a table – all appointments are outputted in the table in the medical record tab

The program must allow the user to search for an appointment of a specific keyword in a specific column and output it into the program and into a text file – after searching for a specific keyword and pressing search the program will output all the search results in a file called 'appointments.txt', in addition to this it updates the result in the treeview

The program must allow a user to add an appointment and create a transaction connected to that appointment's primary key at the same time in addition to saving the date and time of that transaction – when adding an appointment, the program creates another record in SQL that creates a connected transaction for that appointment, it also saves the date and time that the transaction record is created

The program must allow a user to cancel an appointment – if you highlight a record and press cancel selected appointment the appointment will be set to cancelled in both the treeview and the SQL table

The program must list all appointment information in a clear fashion – all appointments in the program are listed in an ordered and clean fashion within the treeview with nothing crossing over into other columns

Transactions

The program must allow the user to add a transaction to the transactions table if all information added is valid – the program allows user to add a transaction by clicking the add transaction button on the client table and then filling out the transaction form with appropriate data

The program must validate all data inputted into the transactions table – a number of validation checks are used throughout the program to validate that all data entered in the add or edit transactions sections is correct

The program must list all transactions in a table – all transactions are outputted in the table in the transactions tab

Candidate Number:
3097

The program must allow the user to search for a transactions of a specific keyword in a specific column and output it into the program and into a text file– after searching for a specific keyword and pressing search the program will output all the search results in a file called 'transactions.txt', in addition to this it updates the result in the treeview

The program must allow a user to add a transaction - if you press add transaction and input suitable data in the form the program allows you to add a transaction

The program must allow a user to edit a transaction - if you press edit transaction and input suitable data in the form the program allows you to edit a transaction

The program must allow a user to delete a transaction – if you highlight a record and press delete selected transaction the transaction will be deleted from both the treeview and the SQL file

The program must allow a user to output all transactions of a particular date and output the total sum of that date – if you write a date in the lower entry box in the transaction tab and press save a text file will be created with a list of all the transactions of that date and at the bottom the sum of those transactions will be listed

The program must list all transaction information in a clear fashion – all transactions in the program are listed in an ordered and clean fashion within the treeview with nothing crossing over into other columns

Staff

The program must allow the user to add a staff to the staff table if all information added is valid – the program allows user to add a staff member by clicking the add staff button on the client table and then filling out the staff form with appropriate data

The program must validate all data inputted into the staff table – a number of validation checks are used throughout the program to validate that all data entered in the add or edit staff sections is correct

The program must list all staffs in a table – all clients are outputted in the table in the staff tab

The program must allow the user to search for a staff of a specific keyword in a specific column and output it into the program and into a text file – after searching for a specific keyword and pressing search the program will output all the search results in a file called 'appointments.txt', in addition to this it updates the result in the treeview

The program must allow a user to add a staff member- if you press add staff and input suitable data in the form the program allows you to add a staff member

The program must allow a user to edit a staff member - if you press edit staff and input suitable data in the form the program allows you to edit a staff member

Candidate Number:
3097

The program must allow a user to delete a staff member – if you highlight a record and press delete selected staff the staff member will be deleted from both the treeview and the SQL file

The program must list all staff information in a clear fashion – all staff in the program are listed in an ordered and clean fashion within the treeview with nothing crossing over into other columns

Usernames and passwords should be stored in this table however passwords will not be able to be seen within the program – passwords are hidden out of view from this section due to data security reasons and usernames and passwords are both stored within this file

Overall

The program must be in third normalised form – the table is in third normalised form, you can see the design stage for evidence of this as I used the same data structure in the final program

The database must be stored in SQL – the database is stored in SQL, you can see this as all the code written for databases uses SQL commands

The program must implement views – views in addition to a login system is used in this program, this is done to prevent certain tabs being accesses in addition to certain buttons not showing if you're not a certain position

The program must contain a GUI - if you use the program you can see that it contains a GUI

The program must be fast – logging in is a quick and easy process and pressing tabs in instant, in addition to that the tables are automatically refreshed when a record is added

The program must contain a readme.txt to guide any new users on how to use the program and how to troubleshoot – a readme.txt file is in the same folder as the code, you can see that it contains the main usernames and password for each position in addition to a guide on how to use the program

The program must be intuitive and easy to understand, and it shouldn't be difficult to use any of the functions – based on the feedback from staff the program was intuitive and easy to use, and all controls were simple

9.14: Comparison to commercially available systems

What I'm comparing

Given that clinics are used worldwide throughout the globe I decided that I should compare my final system to ones that currently exist. I would be comparing these on a variety of aspects, these would be:

- The GUI

- Data security

- Data structures

- Validation

- Additional features

The GUI

The GUI of currently existing systems is very similar to others, however based on what I have looked at is mostly web based. I did find that it typically took more clicks to preform an action compared to the system, this was due to opening out the menu on the side of the screen and the overall program being more complicated. I found that a lot of company's priorities looks over use which I disagree with. I believe that a simpler system would be better for the clinic as it allows the less technology literate to understand how to use it, and it also isn't necessary as the clinic doesn't require a very complicated program.

Data security

The data security was different depending on the system I checked, some systems had no views within their system and just a simple login system, whereas some had no login system at all. Others used a google login button to enter their system. I believe that my views and login system are more secure than theirs, however some of the programs had data that was encrypted and backed up which may be more important given the scenario.

Data structures

The data structures varied a lot depending on the system I checked, I found that in a number of cases the company was just given a template for the program and thus had to create their own tables and connections which resulted in a number of databases having severe issues with data redundancy. Others however were normalized to higher forms than mine. I believe that I my data structure was good enough given the size of the company I was working, as the ones that had a high normalized form usually were used by much larger companies

Validation

Surprisingly, a number of companies had no validation at all, and many of those that did had very little. As a result, I found that my program had significantly more validation than a large number of currently existing systems. For example, many of them only had postcode checks for those that worked for one type of format, instead of all the possible ones you may have in the UK and date checks would allow you to input a 31st day in February or a 13th month, instead just checking if that value contained a digit.

Additional features

This is where other commercially available systems were significantly better than mine, a number of them allowed you to look at scans and were connected onto a server. Some allowed you to look at 3D scans and move around in them had an AI help desk that you could talk to. However, I do believe that this would have been much too complicated for the clinic and that the time spent on developing such features could be better focused elsewhere.

Overall

Overall, I believe I match up well against other systems. I stand by the believe that my system would be better for the clinic in comparison to the ones I looked at. This is for a number of reasons, the first reason would be the price of the complicate and details systems, these were much higher than the company would be able to afford, other systems were off-the-shelf meaning that they weren't custom made for the company, this could result in specific requests not being met. I also found that in many GUIs they often prioritized the aesthetics of the database over the user functionality, in contrast mine has a strong emphasis on the functionality to the user. My system contains all the features requested by the clinic while many other examples of software don't have all of them and instead you would have to purchase multiple systems to get the same features requested.

9.15: Changes that could be made to avoid problems experienced

There were three main issues with my system that I feel could have been done to further improve the system and avoid possible problems

The first of these would be the programs data security, the main issue with my program was its lack of data security, the data within the table wasn't encrypted which would mean that if an unauthorized user had access to the table they would be able to see all the information it contained without having to spend a significant amount of time trying to reverse engineer the data. In addition to this the database is not automatically backed up, this could mean that if the database is ever lost or damaged the data could potentially be gone for life. As a result of this I suggested to the staff members of the clinic that they manually save the database every day on cloud storage.

The second change I would make would be the structure of my documentation, I feel like the structure of it can be messy and hard to understand and I should have planned out better what I was going to write. This can result in issues with any future person trying to look through my program and its documentation as they may not easily find what they are looking for. I have attempted to somewhat restructure the documentation that I already have however this takes a significant amount of time and is still not as good as it would have been if I based the documentation on structure in the first place.

The final change that I would make would have made would be to connect the database to a sever, this would allow any other members of staff to access the data and see it changed live. It not being connected to a server can mean that you would have to request the new data from another member of staff and it would make trying to put data that has been updated on different computers complicated and time consuming. To get rid of this problem I could have initially connected the program to a sever to allow all of the data to be easily accessed wherever you are. In addition to this, it would also allow staff members to possibly access data from home which could be useful if, for example, a call comes in after hours about wanting to book an appointment at the clinic and all of the staff members are at home.

9.16: Overall

Overall, I believe the that entire project was successful. I believe that the software that was created was strong and that the documentation throughout was detailed. I believe that the software that was created for this program was strong and that user interface was very intuitive and easy to use in addition to containing a large array of features, inputs and outputs. I am very pleased with the software that I have created, and the clinic are too. I am also very happy with all the validation and other data security methods that I have used throughout the program which is a strong improvement over their previous paper-based system. I also believe I have done a comprehensive range of documentation and have had a suitable scope in my project. I also believe that the design of the project is strong and strongly lays out the structure of my program and how exactly it was laid out, I do believe however, that if given more time I could improve data security, fix some of the issues with the program and add more features and if I attempted the project again that the documentation would be significantly less muddled up. Overall however, I am very proud with the project that I have created.