Creating an Original solution

Tuesday, 16 January 2018 11:55 AM

After considering existing solutions, and discovering that they do not provide the client with the specified requirements for a dedicated patient database solution, the decision has been made to create an original solution. A number of decisions about creating an original solution will be addressed in this document.

Web-based or Windows application?

While the previous solutions displayed to an extent, the superiority of web-based application for such a project, with a small team of people, low budget, and limited experience, there are new factors to be considered. As an example, I have no experience working with databases, a constantly changing UI, or HTML. The first two new programming concepts will be a new challenge to overcome when creating this project, separate from the rest of the smaller challenges. Attempting to re-learn every single concept, already known to me in HTML, CSS, and JavaScript, makes this project very difficult to create. Furthermore, while a web-based application allows for access from a variety of platforms, it also allows people outside of Dr Zolfaghari's practice to attempt to access the database. While the program will include a password protection function, this will nevertheless increase the risk of unauthorized access. Finally, a web-based application with a professional URL will require a yearly payment of $100. For these reasons, and with consideration of the nature of the development team, the decision has been made to make the final product a WPF Windows application, written in C# and XAML, as these are the areas in which the team has experience.

Where would the database be kept?

The database must be stored either on-site, or in my computer at home. There are various factors to take into account here. Firstly, as the client does not own the institution in which they will be using the solution, my access to their network is highly limited. Secondly, it is unclear how large this database can become, and whether the client's work computer has enough storage for such a database. These two factors begin to justify the more complicated, yet more effective option of hosting a server at home, linked to the database. The program can then access the database via a TCP connection. Furthermore, hosting the database will allow me to easily access the database and server, giving me a connection with the program without me having to go there. For this reason, it is decided that the database will be kept on my computer, rather than on-site.

What database application will be used?

Here is a list of database applications with their positives and negatives:

MS Access:

Positives	*Negatives*
+ All data is inside a single file + Already acquired by the team + Very easy to use	- Multimedia data is difficult to incorporate - Poor Time Critical transactions

SQL Server:

Positives	*Negatives*
+ High Performance	- Extremely expensive, up to $5000

+ Manageability	

Oracle:

Positives	*Negatives*
+ Very Stable + Helps with data mining	- Also costs thousands of dollars - Steep learning curve

After considering some of the most well-known databases to use for the solution, the decision has been made to use MS Access. This is due, mainly to the fact that the database for the solution will not be very large in size. Furthermore, this development team does not have the funds to afford an expensive database such as SQL Server or Oracle. Due to this, we must simply work around the issues of MS Access when creating the solution. For example, rather than store multimedia files in the actual database, we may simply store the location of the multimedia file and the server may retrieve the file using the location and send it to the client, this way, we can avoid actually storing multimedia files within the database, and just store them as normal files on the hard drive of the server. The database can be used to direct the server in this case. Furthermore, due to the small size of the database, time critical transaction should not be an issue. For these reasons, we have decided that MS Access will be the most appropriate choice for a database to use in this solution.

How will the database be encrypted?

There are a variety of pre-made algorithms to choose from, for example the Rijndael algorithm, or the Triple DES algorithm. We may also use the original encryption algorithm written in the last project. While the more well-known encryption algorithms are more tested and reliable, the use of the encryption algorithm of the last project certainly has its advantages. Firstly, it must be addressed that the renown of the pre-made algorithms does nothing to discredit the security of our custom algorithm.

Our custom algorithm fundamentally works with the same idea as DES (the standard for file encryption for a long time) encryption, however, it has a variety of extra elements that make it far safer than DES encryption. For example, the custom algorithm breaks the file into parts, encrypts those parts, puts them back together, and then encrypts the entire thing. This adds a new layer of protection to the encryption algorithm. Furthermore, the algorithm uses character values to encrypt bytes, the character values of characters change depending on the position of the byte and the position of the character in the password. This means that file sizes change the way the algorithm encrypts. Furthermore, the "anti-brute force" aspect of the algorithm uses a second key to create stud bytes on either end of the file, changing the file size until the correct key is entered in. This means that the sizes of the parts the algorithm breaks the file into will be incorrect if the correct key is not entered, meaning the entire decryption process will be off. This means that if one was to brute force the database, they would have to repeat the entire brute force process multiple times until the correct key is found.

Eg: on a 1 MB file, the algorithm splits it into 250 KB parts and encrypts each part (remember that the algorithm relies on file sizes to determine how it encrypts), the parts are put back together and encrypted again. Finally, if the brute force key is '500500', 500 bytes (.5 KB) of fake bytes are added to either end of the file, the total file size is increased by 1 KB. This means that the file sizes are all out of place and each part, as well as the whole file is decrypted incorrectly, even with the correct password. This means that after trying an endless number of passwords, to no avail, a hacker would have to take off one byte from the front of the file, and start the entire process again. The hacker must keep doing this until the right number of bytes are taken from each end of the file (note that the same number of bytes are not always added to each end, eg in the key 111, 11 are added to the beginning and 1 is added to the end, as the fake and encrypted bytes are indistinguishable, the hacker may begin deleting parts of the real file, further undermining their effort to decrypt the file.

The development team has an extensive knowledge of the operation of the encryption algorithm, and the hacker community has no knowledge of this algorithm, as it is not publicly available in any program. This adds a new layer of security to the algorithm, as hackers would have no idea as to how they could go about decrypting this database.

Due to the extensive knowledge of this algorithm possessed by the development team, implementing this algorithm would be easy. This, in combination with the fact that many pre-made algorithms would be less secure, and the fact that the development team has no experience using external algorithms for encryption, has led to the decision to use the already developed encryption algorithm of the development team.

Sorting

A decision must be made regarding the sorting of patient information in the application. Due to the simple nature of the sorting feature, the decision has been used to utilise the already existing and established sorting algorithm known as 'selection sort'.