C22063583 - YOUTH GROUP SCENARI

**Reflection**

One of the ways I could have altered the project was changing the dbms so that I used non-relational no-sql software to create my database. It can often run more efficient than traditional sql databases. I do not believe that would have helped as non-relational are best used for unstructured data whereas sql is best for relational data. My Database is based on the 5 main objects students, volunteers, parents, sessions and badges and how they interact so I think using no-sql software would have subtracted from my final product

I believe my succeeds in a lot of areas but I also have noticed in retrospect there are a few things I would have changed about the final product. One of which is my hashing algorithm. I used md5 as it was easy and built into mysql the problem is it isn’t really that secure. It was found the algorithm could easily produce collisions where different inputs would lead to the same output and was eventually declared “cryptographically broken and unsuitable for further use” (cmu software engineering institute 2010) this could easily be solved by using a different algorithm instead.

One of the issues I have with my software is the add to group function. I believe I used more logic than I should have done. The function should work out how many people are actively volunteering multiplying by a value depending on their age group then subtracting how many students are attending. if there are any spaces it would select the latest students from the waitlist and using a “limit” with the variable of the result as a subquery in an update query. However you cannot use a limit in a subquery (in this version at least). You also cannot store more than one piece of data in a variable so I settled on using a for loop adding one student for each time it loops back. I believe there should be a better way to accomplish this maybe with updated software.

I believe that I gained a lot of knowledge on stored procedures. Before this project, I knew the basic syntax so I started out using basic queries with inputs such as the “addvolunteer” and “addstudent” functions and added functions on the data such as md5 to alter the inputs. Over time I gained confidence to add error handling for when a unique

Another Issue was with what data I hashed. A students ID is made out of their full name and their password. It is feasible to expect 2 students with the same name to exist and it could not handle the same name and password. I would change this by hashing the studentCode instead as its fully unique meaning no collisions.

A minor set of errors would be mishandling datatypes for example phone numbers being “int(10)” when they should be varchar instead so if it starts with 0 it isn’t removed and should also have the length of 15 to store all possible numbers. Similarly I should have assigned to the “subscribed” row on the parent table the data type int instead of Boolean. A parent can have multiple children so it would be better to store how many times they have to pay rather than if they need to pay.

References

Cmu software engineering institute. 2010. Accessed on 22/4/24 at https://www.kb.cert.org/vuls/id/836068