**Evaluation**

**Fitness for purpose**

My initial idea for this project was to create a software application that stores account information in the database. In my program I have created this by using SQL queries executed by my program when needed. The user can register for an account, play the game, and then log back into that account to have all their progress. The user can also use the leader board system backed by a bubble sort algorithm to filter for the highest scoring users in the database. This gives incentive for the user to keep the same account and use the login sign out save account system.

I believe my solution is fit for purpose as it meets all the requirements set in the requirements specification. Adding extra functionality to the project in terms of using emails to verify logins (otherwise known as 2FA) was always something I strived for as this is how literally everything works these days. As a result, from the extreme delays from the database connection and not having the support to fix this I then emailed a tester from a company, and he helped me. Working with Phil the tester took a while and was not as efficient as working with an advanced higher teacher also explaining further delays. I was stuck between either trying to quickly get an authentication system in and potentially rush testing and evaluation or to drop it. I removed this from the project.

I think that my end user test evidently shows that this project is fit for solution. The fact that a user can log in and have their level saved and the program output that shows that the login system works.

**Results of testing**

Through testing, my program does store the information in the database is retrieved upon request by use of SQL queries after connecting. Users can register an account complete a level or two and then log back into that account and play the same level that they logged out on. A user can also only play the game when logged in. User inputs are validated as shown for the login system, the register system the level system and you could also not sign out or save an account when not logged in. The username and password a user enter in is always 8 characters otherwise an error message is sent out. The queries for inserting and deleting and reading all work as they should. When user clicks save account, it deletes their account and then inserts, this was something that seemed tricky as I was executing 2 queries one after another and it worked. I checked that the leader board works by entering several different instances of databases and it outputted the right answer every time.

The major issue I found in testing was if a user doesn’t know the answer to a question then how can they play the game, In future development this will need to be considered.

**Maintainability of the code**

The use of SQL queries in my code makes it extremely maintainable. If a developer wanted the query to do something else like sort the table then they could do this. The use of an array of records also proves maintainability as if a developer wished to add another function such as a lives bar then they could do this by adding another variable to the structure.

**Robustness of code**

In terms of robustness my program is not at all robust as it could be, there isn’t a user input that is not validated and if the program has an error with a process such as reading data or a level on a database somehow being higher than 3 then processes are put in place to stop the program and say error. I wanted to add validation so that users could use different languages or characters but because of the time situation I wasn’t able to.