**Criteria 6 modifications and justification**

**Form changes and removals**

**frmMain changes**

In the main window a new button was added that was not present in the original design as a means to refresh the list box on the main screen was not considered. A button was the easiest way of performing said function as well as being easy to understand for the user, and thus a new button under the name of btnRefresh was added to re-add the questions stored to the list box, accounting for changes the user may have made during their usage of the program.

A new menu strip item was added to delete the files storing the questions. This was not present in the original design as it was thought that a function to delete individual questions would be sufficient. However, after coming across an error that could not be fixed within the economic constraints of the project (an error which seems to be unrelated to the code) another menu strip item was added to simply delete all of the questions through the deletion of the files that house the questions added. This was to ensure that the deletion functional requirement was met whilst also providing the user a convenient way to completely clear their added questions.

Two menu strip items related to the auto boot function of the program was added to the solution. A way for the user to enable such a feature was not considered in the final design, and thus two menu strip items were added to compensate, one to turn it on and one to turn it off.

**frmSearch changes**The layout of the form was altered to fit in a group box housing radio buttons that allow the user to change what they wish to search by, either by tag or name. This was done due to economic constraints and developer experience preventing the ability of being able to search using both the tags and name simultaneously, and thus to preserve the functionality in some form it was separated.

**frmStudy** **changes**

New controls were added to allow for the display of multiple-choice questions, which the design did not have. This was achieved through the addition of a group box with radio buttons which have their text match with the corresponding answers stored for the question. Furthermore, the design was slightly altered in that studying short answer and multiple-choice questions is done separately due to economic constraints. Additionally, the score functionality was unable to be tested due to an error external to the code preventing evaluation of the function, thus its effectiveness cannot be measured nor guaranteed.

**frmAddModifyQuestion changes**

Within this form more textboxes and a label was added to account for the ability to make multiple choice questions. This also included a textbox and label for adding the correct answer to the multiple-choice question so the user can assign the true answer to the question. Additionally, the decision to store the multiple-choice and short answer questions in separate files was made to make functions that make use of both easier with the two separated as it is easier to merge the two by using arrays and list within the code.

**The other forms not stated here stuck to the original design.**

**Pseudocode**

As a whole, the code within the solution had to be far more detailed than the pseudocode included within the final design, added validation as well as more code to fit the syntax and to properly fulfill the planned function. Furthermore, pseudocode was not created for some functions such as editing or opening up the program on boot.  
  
The random nature of the study questions being displayed was scrapped due to time constraints and the functionality being deemed low priority due to it not being one of the functional requirements.

Within the pseudocode a binary search was planned to be used, however throughout the course of development it was discovered that a linear search would be more appropriate as it would be compatible with searching by inputted text and matching it with the stored names and tags. Similarly, the quick sort within the pseudocode went unused instead opting to use the inbuilt sort function due to time constraints.

**Data dictionary and object descriptors**

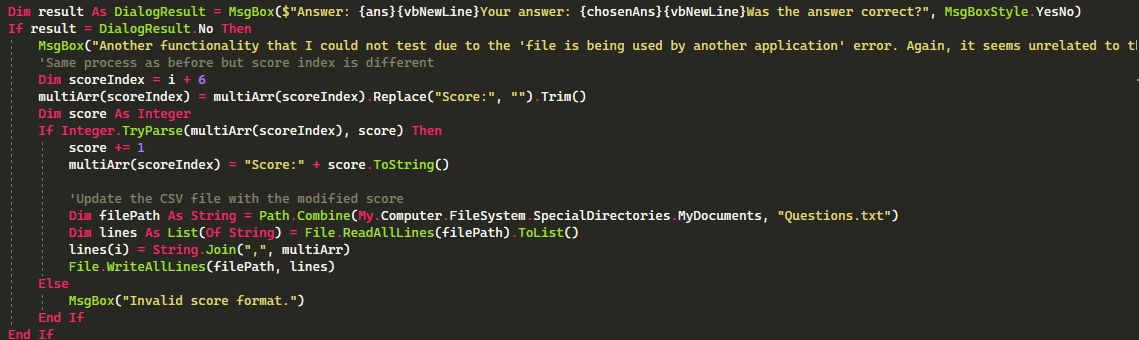
Although most of the variables listed in the final design’s data dictionary were used within the program, it was not considered that a great number more would be essential for the functionality of the solution. This includes having variables store the values that will auto-fill the text boxes when editing a selected question.  
  
In a similar vein most of the object descriptor items were used in the final solution but more had to be added as some of the designs were modified. Names and other properties were left largely unchanged.

**Evaluation criteria not met**

**EC04 - (FR04) Being able to sort the questions by tags**

Although sorting by alphabetical order and searching by tags was added, there was insufficient time to add the ability to sort the questions by the tag. Given that the base functionality of sorting was complete, the remaining time was prioritised for other requirements.  
  
**EC05 - (FR05) A ‘score’ system to see which questions were struggled with the most**

The code for incrementing the score by one is present within frmStudy, however due to an unprecedented error with updating the file storing the questions, the effectiveness of this functionality cannot be guaranteed. Code for updating the label which displays how many times the user got a particular question incorrect was also created.



A screen shot of a computer program

Description automatically generated with low confidence  
  
**EC07 - (FR07) Customisation options for the feature of opening up on boot**

An attempt was made to create the functionality of the program automatically opening up on device boot, however despite the code seemingly being without errors, a registry key that is required for the functionality to occur fails to be created for reasons unknown. Thus, further customisation options for this feature was not completed as the base functionality could not be guaranteed.

**Possible contingencies**

* Unforeseen test cases causing fatal errors
* Device specific issues (e.g., file saving permissions and inability to use the software due to the hardware lacking requirements)
* Complications arising from extended usage beyond the development period
* Use on older Windows operating systems
* Security and volatility of the storage files (as they are stored in the documents folder as plain text, unencrypted)
* Internet functionalities with further development
* Cross-device synching with further development
* Problems arising from the usage of the solution by a wider range of users (e.g., lower technological literacy increasing difficulty using the solution)
* Potential problems arising as the files the solution uses to store data increases exponentially
* Client proposing new features/functions that may require re-working large parts of code
* User’s installed programs may conflict with the solution and cause unprecedented errors

**The following websites were referred to during development to assist in coding:**

* <https://stackoverflow.com/questions/26791067/how-to-make-a-function-sub-available-to-all-forms-in-a-project>
* <https://www.codeproject.com/questions/253297/want-to-make-vb-net-application-run-at-startup-whe>
* <https://stackoverflow.com/questions/47190131/simple-dialog-like-msgbox-with-custom-buttons-vb>
* <https://learn.microsoft.com/en-us/dotnet/visual-basic/developing-apps/programming/drives-directories-files/how-to-append-to-text-files>