QuWordFinder – Developer Challenge

Document purpose

This document has the intention to describe the work done for the developer challenge.

Code Explanation, Analysis, and Evaluation

Overview

In this project, I have developed a WordFinder Console Application that allows the user to input a matrix of characters and then search for specified words either horizontally or vertically. One of the requirements is that the Max Size should be 64 but it didn't specify the Min Size – I decided to set 3 as I think words of 2 characters are not representative. I've also decided to have a square matrix as usually is how exists.

The program includes features such as input validation, the ability to handle various edge cases, and the use of efficient algorithms. The main goal of this implementation is to provide a clear and maintainable solution for searching words in a matrix.

The program was built in VS2022 with C# and .NET Framework 4.7.2 and best programming practise with SOLID principles.

Key Functionalities

Matrix Input: The program prompts the user to input the size of the matrix and then enter each row one by one. The rows are validated to ensure they contain the correct number of characters, and the matrix size is checked to ensure it falls within the acceptable range.

Word Search Logic: The program checks if a word exists in the matrix either horizontally or vertically. This is done by iterating through the rows for horizontal checks and iterating through the columns for vertical checks.

Validation: Each row is validated to have exactly the same length as the matrix size. If the user enters an invalid row, they are prompted again. This was a requirement.

Handling Configuration Values: The matrix size limits and other parameters are managed through configurable constants or settings that can be defined in the program or through an external configuration file.

Looping and User Interaction: The program uses a do-while loop to ask the user if they want to repeat the process. This approach ensures that the user can repeatedly search for words in different matrices without restarting the application.

Results: Display the results indicating the words that were found and if any word in the word stream is found more than once within the stream, the search results should count it only once.

Conclusion

Strengths:

- **Clear Structure**: The code follows a clear, modular structure, making it easy to read, maintain, and extend.
- **User-Centric**: Input is validated continuously, ensuring that only valid data is processed.
- **Efficient Algorithms**: The use of LINQ expressions for word search is efficient and readable.
- **Layered architecture**: This allows us to maintain, escalate and have our flexibility for Future Changes e.g. adding a Data/Core Layer to manipulate DB access.
- **Use StringBuilder**: This improves performance when working with larger matrices. Concatenating strings directly creates new string objects on each operation, as strings are immutable. In contrast, StringBuilder uses a mutable internal buffer, which allows for more efficient string concatenation by avoiding the creation of new objects during each append operation.
- **Use try-catch blocks**: This will guarantee us to have a solid error handling. I've also added a commented line for a future "Log" to track details of the exceptions for better tracking. Specially if we decide to start working with DB