# Advanced Health & Care Technical Test

## My Approach - Jeremy Pridmore

Whilst listening to the verbal brief, I made notes of the key features, the inputs and outputs and some of the soft requirements such as how it is to be tested, memory, speed etc. I noted the technical platform, i.e. to be delivered as a console application and noted that the goal is to focus on writing beautifully structured code and not necessarily how good the output looks. David said that the full details of the test would be delivered via email to me. We agreed a deadline for delivery.

I brain stormed the best approach to kick start this project and the process that would best demonstrate my skills.

## Brain Storm

* Treat as client. New client in myhours and track time.
* Present time spent on project and task break down.
* TFS vs GIT - TFS - choose Scrum process template vs David saying that moving over to GIT - so create a new GitHub repo.
* Need to look at Kaban process if using GIT?
* Use NUnit testing - create new test project. Which runner?
* Impact on memory key requirement - use ANTs Memory profiler? Most significant impact will be on the data structure and search algos.
* Also speed of execution is key requirement, so timing framework to compare performance of different methods?
* Need to come up with my own solution rather than looking at rehashes of other peoples solutions.
* Need to consider software design patterns - good OO design - perhaps consider outlining solution in UML behavioural/structural diagrams.
* Consider BDD - scenarios to describe behaviour of software. Although, David suggested they are more TDD.
* Need to make sure I evaluate performance and describe how I approached the development - using this document!
* Consider using Waffle.io to track project so that David has access to my tasks.
* Need to focus on just the requirements that are listed and not add any other superfluous features.
* No time for diagramming - stick to verbal descriptions of requirements and tests.
* NOTE: I am purposefully resisting the temptation to go and look up the algorithm online, instead attempting to demonstrate my analytical skills by breaking down the problem using my own knowledge of algorithm design and data structures.

## Questions

* How does want solution delivered/packaged?
* Any performance metrics need to meet?
* Share repo now so that we can discuss if required?

## Process

* Research/Investigation.
* Adhering to agile tdd process
  1. Stakeholder participation
  2. Initial requirements envisioning and research.
  3. Writing Acceptance Tests to fulfil requirements.
  4. Just in time modelling for each test.
  5. Planning iterations and releases.
  6. Mapping requirements and tests onto a test matrix.
  7. Building the backlog.
  8. Implementing tests and code/mocks/stubs that is just barely enough.

## Participation

* Delivered links to github and pivotaltracker.

## Requirements Envisioning

* Mapping out high level requirement(s)
  1. **Feature** - Search for Shortest List of Words
     + **In order to** obtain a file containing the shortest possible list of words starting at Word A and ending at Word B where each word is found in a given dictionary and each subsequent word has only one letter which is different from the set of letters in the proceeding word.
     + **As a** User
     + **I want to** execute a Program with a Simple UI, type the name of a dictionary file, type Word A, type Word B, type the name of the results file, start the search and be informed that the results are ready to be viewed in the results file.
  2. **Scenario** -
     + **Given** a text file containing a dictionary of words with each word listed on a separate line

**and** a start Word A

**and** an end Word B

**and** assuming both words are present in the dictionary

**and** the name of a results file

* + - **When** I execute the program
    - **Then** I want to be prompted to enter the name of the Dictionary file

**and** I want to be prompted to enter Word A

**and** I want to be prompted to enter Word B

**and** I want to be prompted to enter the name of the Results file

**Then** I want the program to generate the results

**and** I want to be informed that the results are ready

* Breaking down high level scenario into lower level features/scenarios.
  1. **Feature** Read name of Dictionary file
  2. **Feature** Read Word A
  3. **Feature** Read Word B
  4. **Feature** read name of Results file
  5. **Feature** Load the list of words from the dictionary file
  6. **Feature** Find words that only have one letter which is different
  7. **Feature** Link words together that are only one letter different to form paths
  8. **Feature** Find the first Word A
  9. **Feature** maintain a list of found words
  10. **Feature** Check if Word B has been found
  11. **Feature** record the length of a found path
  12. **Feature** backtrack to search alternative paths
  13. **Feature** Record when a path has been searched
  14. **Feature** Stop searching if the length of the path is greater than the shortest found path
  15. **Feature** Stop searching if all paths have been searched
  16. **Feature** Write a list of Words that form the shortest path to the results file.

## Assumptions

* **Situation:** The SoR ask for a program to be written that calls a procedure that takes four parameters. It does not state the source of those parameters.

**Inference:** Should the program prompt the user to type them, should they be supplied on the command line, should they be read from a configuration file?

**Assumption:** The program will read them from the command line as this offers the best interoperability.

* **Situation:** The SoR states that it is safe to assume that both StartWord and EndWord can be found in the dictionary.

**Inference:** The dictionary may not be read completely or corrupt.

**Assumption:** The program should still check that both StartWord and EndWord exist.

* **Situation:** When searching for paths, more than one path of the shortest length may exist.

**Inference:** Should the program output both paths or the first one if encountered?

**Assumption:** Just output the first one.

* **Situation:** The dictionary file may not exist, yet the SoR does not state what to do in this case.

**Inference:** The program will need a dictionary in order to continue.

**Assumption:** The program should output a simple message saying that the dictionary could not be found.

* **Situation:** The SoR does not state the format of the contents of the dictionary file, nor what to do if the file is empty.

**Inference:** Having inspected the contents of the sample dictionary file, each word is listed on a separate line.

**Assumption:** The program will be written to read each line and treat the contents of that line as a single word and discard and empty lines.