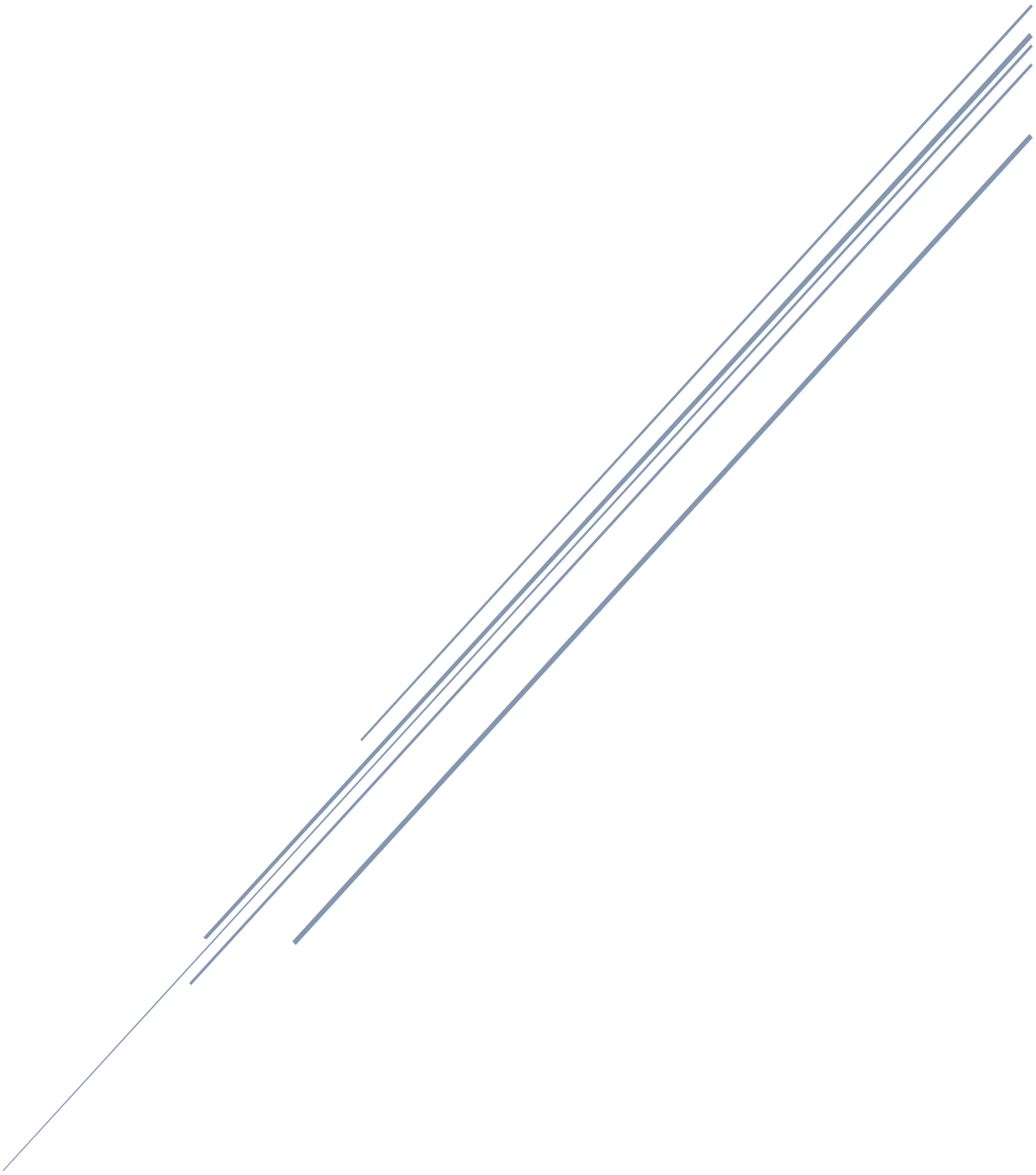
SUDOKU SOLVER USING A BACKTRACKING ALGORITHM

MARC TEMPLETON | B00687477 | APRIL 2019



Ulster University, Jordanstown
BSc Hons Computing Science

Abstract

This project developed a Sudoku solver using a backtracking algorithm.

The project originated with the intention to develop a solver for the popular puzzle, ‘Sudoku’.
This product was to incorporate a solver using the functional programming language, q in which recursive function calls would be utilised.
This evolved to allow the user to play Sudoku.

The solution was presented through a website to the end users, which would be interacted with through the use of the mouse and keyboard.

An external API was developed to allow for puzzles to be fetched for the application.

This report outlines and describes in detail the background, planning and analysis, implementation and critical evaluation of the product.

Table of Contents

1.0 Introduction

1.1. Problem Elucidation and Statement

Solving Sudoku is challenging.

Sudoku is a combinational puzzle in which Wilson (2006, p.ix) describes it as, “given a 9x9 grid divided into 3x3 mini-grids (boxes) and with some numbers already placed in it, fill in the rest … that no number appears twice in the same row, column or box.”

Despite its usage of numbers, Sudoku requires no mathematical computation. “The only thing you need to solve a Sudoku number place puzzle is logic.” (Hinkler, instructions)

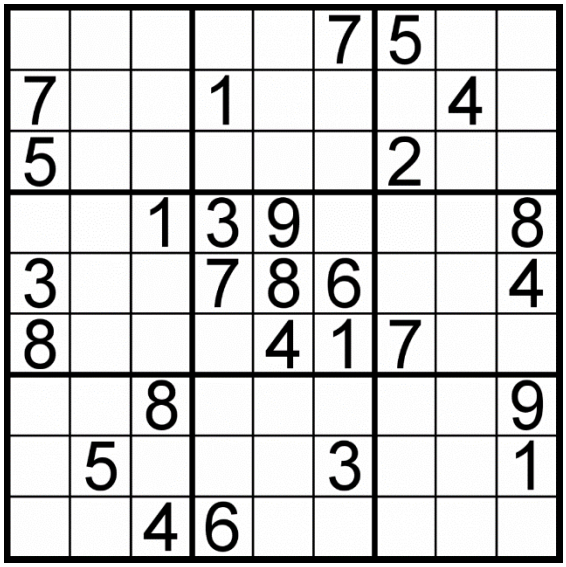


Figure 1. A typical Sudoku board (Sudoku of the Day, 2015)

Sudoku was originally published by Dell Magazines (a subside company of the puzzle giant Penny Publications) in the late 1970s in New York under the name ‘Number Place’. (Sieger, 2013, 00:01:13)

However, it was not until it reached the shores of Japan, a few years later, where its iconic name was coined.

Nikoli is a Japanese publisher of logic puzzles who first introduced Sudoku to its audience in 1984, or rather it published 数独, comprising of the Japanese characters Su and Doku roughly translating to ‘number’ and ‘single’ respectively. (Smith 2005)
Since then it has reached most of the globe and since 2006, boosts an annual Sudoku Championship which takes place in countries around the world. (World Puzzle Federation, 2018)

Since the early 2000s, Sudoku has been played on desktop computers through Wayne Gould’s downloadable application (Gould, 2018) and countless online sites, one of the largest to this day being Web Sudoku. (Web Sudoku, 2018)

Sudoku puzzles need to have a clear layout to allow users to work through them.

This must also be the case when ‘notes’ are used – that is, small numbers placed in cells to denote possible solutions for it. It is possible for these to clutter the Sudoku puzzle making it difficult to use logical reasoning to eliminate noted numbers.

Sudoku solvers have been around since its western introduction in the mid-2000s.

Gould spent 6 years developing a computer algorithm for creating Sudoku puzzles called Pappocom which incorporated a solver. (Smith, 2005)

Solving Sudoku is difficult due to its unique solution and partially given clues.

Therefore, creating an algorithm to solve such puzzles is both complex and challenging. The positioning of given clues heavily influences the difficulty of Sudoku puzzles and there is no direct correlation between the number of clues given and the difficulty of a given puzzle; it is possible for puzzles with a greater number of starting clues to be more challenging to solve than those with fewer.

In this day and age, the mobility and global availability of internet access and the advancements in search engines efficiency means that most people have libraries worth of data at their fingertips; fuelling our obsession for understanding just about everything.

This has made us inpatient. Solving Sudoku takes time, however the desire to know or solve things quickly, means that the pleasure of taking an evening to look at a problem and solve it, is equally enjoyed by the speed in which that problem can be solved.

In addition to the complexity of solving Sudoku puzzles, verifying completed ones can also be an issue.

Looking through a completed Sudoku board is both tedious and monotonous and whilst dedicated Sudoku and other puzzle books contain the answer, other form of mediums (among the most popular for daily Sudoku plays, e.g. newspapers and magazines) do not. Therefore, there is a need for quick, reliable and customisable solutions.

Traditional solving engines use pre-programmed Sudoku rules to find the unique solution, e.g. naked pair, x-wing, swordfish, etc.

However, as Sudoku has progressed, there have been many adaptions (still confined to a typical 9x9 grid) to challenge enthusiasts including: Addoku, Nonomino Sudoku etc. These alterations mean that traditional solvers must be reprogrammed to deal with adjustments to the rules.

Despite its lack of mathematical computation, Sudoku still has a great interest in the mathematical community. Primarily is application in computational complexity – the study of complex systems.

1.2. Project Aim

The project will allow users to complete Sudoku puzzles.
These puzzles can be custom created by the user or randomly sourced with varying difficulty.

The project will also have a solving mechanism incorporated that will discover the unique solution to viable Sudoku grids using a backtracking algorithm.

1.3. Project Objectives

OBJ 1) Display blank Sudoku boards
OBJ 2) Read-in Sudoku boards via an API
OBJ 3) Display read-in Sudoku boards
OBJ 4) Allow numbers to be entered/altered for cells
OBJ 5) Get new Sudoku boards
OBJ 6) Create an algorithm that can solve the puzzle given the pre-existing clues
OBJ 7) Host a web application on a local server
OBJ 8) Create APIs to run the algorithm on the web application
OBJ 9) Create a user-friendly interface on the web front-end
OBJ 10) Allow unit testing throughout
OBJ 11) Allow acceptance testing once a product is established
OBJ 12) Ensure code is self-describing through structure and naming conventions

2.0 Literature Review

2.1. Part I

2.1.1. Popularity and Relevance

Sudoku's epidemic origin can be traced back to a retired Hong Kong judge known as Wayne Gould in 1997.

"As soon as I saw the grid with the empty squares, I felt very tempted to fill them in." The opening thoughts to what would bring Britain into a puzzle frenzy. (Smith, 2005)

Sudoku is a puzzle enjoyed by participants of all ages Gould sells puzzles via his website, www.sudoku.com, and has customers ranging from seven-year-olds to octogenarians." (Smith, 2005)

Puzzles like these promote logical reasoning and problem solving; skills highly desired amongst the UK workforce. However, there is a deficit in them, a recent report highlighted that four in ten hiring managers said they see a shortage in problem solving skills with 26% of those managers expecting to see this decrease throughout 2018. (Walters, 2018) To halt this decline, the UK government has recognised that skills in STEM are an important asset to the economy.

The World Puzzle Federation (2018) recognises this push to encourage logical thinking for young people and as of 2014 it has included an under 18-year old category in its annual Sudoku World Championships.

Sudoku is also popular amongst other age groups, including the elderly.

"Many seniors are interested in keeping their minds sharp and in continuing to strengthen their cognitive abilities as they get older." (Easybrain, 2018) With its simple concept and aesthetic it is an ideal pastime and Jeff Slater (2015) who has worked with elderly people for the last 24 years believes that Sudoku is a great cost-free application to enhance their quality of life.

There have also been suggestions that completing logic or memory puzzles, like Sudoku, can help delay mental health issues.

Recreational activities for the elderly are in the highest demand ever and will only continue to grow as the global population continues to age.

Appendix A showcases Japan's population pyramids where the number of over 50-year-olds has more than doubled over 40-year to over 57 million with just under 10 million individuals over 80 years of age. There are also trace amounts of those over 100-years and so, not only are recreational activities going to be in high demand, but suitable ones to suit limited mobility and those with mental health illnesses. Sudoku with its simplicity is suitable for many elderly people.

The World Puzzle Federation (2018) acknowledges this and as of 2014 it has included an over 50-year old category in its annual Sudoku World Championships.

Sudoku is also available across a number of different mediums.

Sudoku initially grasped the western world by storm when Gould introduced it to the Times newspaper. Papers began to attempt to get a one-up on their competitors during this time, such that some claimed to have it first just under a different name (Kemmochi and Mikami, 2007) or by making more drastic measures like The Guardian (2005) did who published a Sudoku on every page of its subsidiary G2 paper, including its cover with the company stating that is was to, "demonstrate our superiority over our so-called rivals."

This Sudoku appeal is still evident within the newspapers of Britain to this day – however not to this extent.

Most papers host a small dedicated puzzle section to provide some brainteasers to consumers during breaks and alike. There can be upward of 3 boards of varying difficulty to challenge enthusiasts and novice alike. The Sun (2018) newspaper in 2017 recognised as the UK's most popular paper boasted with over 30 million readers of their content.

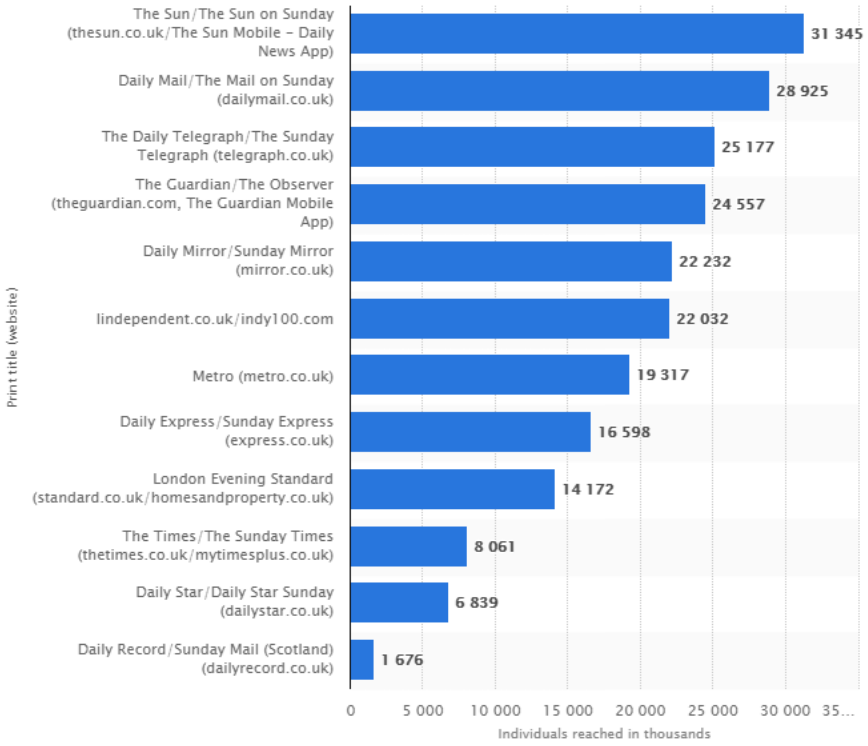


Figure 2. Newspaper reach for the UK per month per thousand (Statista, 2018)

Newspaper corporations have moved into the digital age and most host an online edition of their content. Some of these online servers contain a puzzle page, where users can tackle Puzzles.

The Telegraph (2018) online Sudoku puzzle section for March 2018 boosted on average just under 500 plays (489) daily plays across its 2 daily puzzles (see Appendix B).

Prior to Sudoku's appearance within newspapers the puzzle's main paper-based distribution came in the form of monthly magazines.

These magazines were essential in first: introducing Sudoku through Dell Magazines' publications and then: popularising it globally. Japanese puzzle magazine distributors such as Nikoli still have a large audience today with Wilson (2006) stating that there are "five [Japanese] publishers currently producing monthly Sudoku magazines for over half a million readers."
In addition to this there are a variety of strategy guides and books outlining the best ways to solve Sudoku aimed at novices and those wanting to fine-tune their skills.

Sudoku is primarily completed during commutes due to the availability and portability of dedicated books found in station stores.

Randstad (2013) found that 14.5% of people read a book, newspaper or magazine on their commutes; 20.8% of which happen by train or bus. This coupled with the commute time which the BBC (2016) reported to be over two hours or longer each day for 3.7 million travellers give them plenty of time to relax.

There are also dedicated websites which see millions do Sudoku monthly such as WebSudoku and recreational mobile applications across Android, Windows and iOS.
With the advancement in technology including AR/VR and Smart TVs there has also been adaptations in these of Sudoku application such as a 360° AR solver developed and demoed by geaxgx1 (2017).
There have also been TV shows, board games, card games and various video games with Carol Vorderman at the forefront.

On mediums which do not provide solutions to the grids such as newspapers and magazines, the company may impose a small charge for clues or solutions, e.g. The Sun newspaper's daily, 'Sundoku' provide clues or solutions at a charge of 80 pence per clue/minute.
Clearly a solver would save costs when struggling with a puzzle.

2.1.2. Mathematical Properties of Sudoku

Why is Sudoku so difficult, or easy, to solve?

Below the author delves into the mathematical properties of Sudoku and their impact on the difficulty to solve these.

Whilst modern day Sudoku is known to introduced in the late-70s, its concept was thought of well before by the Swiss mathematician Leonard Euler who devised 'Latin Squares'; self-described as, 'a new type of magic square'. (Smith, 2005)

8	1	6
3	5	7
4	9	2

Figure 3. 3x3 magic square (England, 2015)

Magic squares are said to be a, "mathematical stunt" by England (2015) and comparable to a magician's act!

These are special grids in which every row, column and diagonal add up a set value. The numbers cannot repeat and can only be used once.

Within a 3x3 grid, Talwalker (2015) produces 8 possibilities but later reveals that there is only one pattern and that the only 7 are simply reflections or rotations of the original.

Aiden (2006) describes Latin squares as, "grids filled with numbers, letters or symbols, in such a way that no number appears twice in the same row or column."

It is possible to map one Latin square to another by performing operations on them such as rotation, reflection or row permutation. Gao (2005) identifies these in their report as 'isotopic' to the original grid in which it was derived from. Therefore, they go onto state that all Latin squares are, "divided into subsets, called 3 isotopy classes" which are distinct from one class to another.

There exists normalised forms of Latin squares (see figure 4) that Aiden (2006) describes as one in which the first row and column have the symbols entered in ascending order.

Latin squares of size 3x3 have 12 variants, however only one normalised form exists.

A	B	C	D
B	C	D	A
C	D	A	B
D	A	B	C

Figure 4. Normalised 4x4 Latin square using letters (York University, 2013)

Sudoku is a form of Latin squares with the added combinational constraint of sub-grids.

The overall board is an example of a Latin square; however, the individual grids are special forms of these.

The standard 3x3 sub-grids in a typical board take the form of a magic square, with the exception that their rows, columns and diagonals do not add up to a set value.

This additional constraint means that there are fewer valid 9x9 Sudoku grids than 9x9 Latin squares.

Felgenhauer and Jarvis used a computer program to uncover that there are 6.671×10^{21} possible grids in a typical 9x9 setup. (Cornell University, 2009)

Similar to normalised Latin squares, Sudoku grids can also be classified in subsets known as Sudoku families with 5.5×10^{9} families existing. (Exeter Mathematics School, 2016, 00:03:50) These operations must preserve the, "underlying structure of that grid including the relationship between the values." (Jones et al, 2012) Taking this into consideration Russell and Jarvis used another program to reduce this known number of Sudoku grids down to 5.473×10^{9} unique Sudoku grids. (Cornell University, 2009)

Sudoku is different from magic or Latin squares in that, they do not start from an empty board but rather, some starting 'clues' are provided. The least amount of clues needed to be provided in order for a valid solution to be logically worked out is 17, which was discovered by the mathematician Gary McGuire by proving that no puzzles with 16 starting clues led to a unique solution. (Numberphile, 2012, 00:01:54)

Sudoku puzzles have difficulties assigned to them.

Tom Davis (2008) reports that this is little to do with the number of clues given initially but rather how difficult it is for a human to logically solve. He discusses that in fact it is to do with the advancement of pre-defined rules which are needing to be applied in order to find the solution. The most difficult rules like x-wings and swordfish would result in a greater difficult puzzle than one which required finding hidden pairs which results in a higher difficulty than simply using naked singles.

2.1.3. Solver

A recursive backtracking algorithm attempts to find a combinational solution by using aspects of graph theory.

Nodes can be connected to any number of other nodes by a pre-determined relationship (usually following a grid or map system), forming 'edges' between them. These connections are mapped out and form a tree structure. This structure expands outwards from an initial starting node known as the 'root'. Traversing down from the root, nodes which have connections beneath them are said to be 'parent' nodes to 'children' nodes. This structure allows for backtracking to be logically laid out and executed by traversing it way back up the tree until it reaches the root node.

The difference in breadth-first compared to depth-first is regards how the algorithm traverses the tree structure and therefore the order of which nodes are assessed.

In breadth-first nodes at equal levels are looked at, before the program advances down the most optimal branch at this level and repeats.

In contrast, depth-first algorithms look at the nodes directly beneath the node under consideration. If a fork in the graph is discovered, the algorithm will pick one path to follow until its end. Once it reaches here, it traverses back up the path and investigates the other fork path for a solution.

During both of these methods a queue is kept and updated providing the path taken to reach this node and the value assigned to it (whether time taken, number of nodes used, etc.) Once the algorithm has finished this queue is revisited and checked for the most optimal path from root to finish and that is outputted.

Solving Sudoku is entirely feasible using a backtracking algorithm.

The typical Sudoku setup allows for a simple graph to be made in a grid formation, in which: nodes are identified as the individual cells and edges between nodes are adjacent cells.

Checking for validity for a given node is done by simply checking the cell's row, column and grid for the value temporarily entered into it.

If the check fails then all assignments to the children of that node is avoided and identified as invalid, reducing computational executions significantly.

Using this approach will always provide a solution because of its brute-force technique with the exception that a board with no solution is entered, in which case this should be identified.

Boards will multiple solutions will be solved with the first solution the algorithm comes across. However, the algorithm can solve illegal Sudoku boards with one solution that cannot be derived using logic.

There are boards which instinctively work against backtracking by making the starting row or column entirely blank, with the solution to the board being the numbers in descending order. These boards are again solvable with backtracking; however, they require the program to execute maximum cycles and

therefore, conform to its worst-case scenario. Using this simple algorithm on larger boards requires greater computational power and time to solve.

Due to Sudoku's typical square grid shape, choosing whether to use the breadth-first or depth-first algorithm variation is largely redundant.

The simplicity of the underlying graph for this problem means that these two methods simply map the relationships between nodes to either checking the row or column exhaustively. Therefore, there is little, to no, computational implication of using either.

Lop-sided boards which have unequal number of rows and columns can add to this computational overhead on very large boards. However, the computational overhead from the size of the board will greatly overcome that of the boards' lop-sidedness and thus, this overhead can made redundant.

There are alternatives to a backtracking algorithm which come with their own pros and cons.

Creating an application that logically completes the puzzle using the pre-defined Sudoku rules ensures that the unique solution to a puzzle is found. Using the same Sudoku rules which the user uses can allow for a step-by-step instruction output to be produced. This approach also allows for invalid Sudoku boards to be identified. Computational issues may still arise with larger boards; however, the application of rules will remain largely the same. Having the rules pre-programmed into the application means that it is not diverse enough to solve Sudoku spin-off applications such as Addoku.

Creating, training and verifying a machine learning (ML) model to solve Sudoku is extremely complex and requires an extensive dataset to do. However, this model can provide almost instantaneous solutions to boards. The ability of incorporating a Sudoku board reader could be developed using similar libraries such as TensorFlow to develop Convolutional-Neural Networks (CNNs). However, with most ML applications, the program is treated as a black-box and therefore understanding the working components is challenging which can make it difficult to develop with it.

The choice of programming paradigm to solve this problem can aid or hinder the developer.

This problem, with its excessive cycles per solve, is suited for a recursive approach.

Recursion is functional programming's (FP) main asset and is extremely well written to work within it.

The simplicity of functions as a method of organising a program makes maintaining and debugging easier. Using pure function which only operate on their input result in no side-effects during execution aiding the debugging overhead.

FP's immutability towards data ensures that the desired result is always outputted and can be said to be deterministic.

This concept allows for concurrent processes to be carried out in parallel because they each work with only the data provided to it and they do not change the state of data any of them are operating on.

This allows for functional systems to be extensively scaled which will allow a solution to solving a board to be applied to any sized board.

The Sudoku problem is a typical example of the P vs NP problem; a Millennium prize problem that CMI (2018) states as, "If it is easy to check that a solution to a problem is correct, is it also easy to solve the problem?"

A core concept that the P vs NP problem provides is the idea of scaling problems into larger forms of themselves. With problems contained within P, the challenge of a computer to solve these large versions of problems doesn't change much. However, with problems contained in NP, like Sudoku, the challenge of solving scaled up versions of them is great due to increased relationships, nodes and checks (hackerdashery, 2014, 00:04:21).

FP is extremely well-suited to up-scaling functions due to its deterministic concept, allowing for parallel processes to take place at once.

2.2. Part II

In addition, with research into the problem area outlined, the author also investigated into existing solutions which are available.

The following applications are freely commercially available across a multitude of platforms and are comparable to the outlined aims.

2.2.1. Sudoku Solver

Sudoku Solver is a free-to-download, ad-free, Android application developed by kevinhhl (2013) and available on the Google Play store with over one thousand downloads.

Once downloaded and opened (inside a minute with reliable internet access) the user is presented with a simplistic, neutral in colour and empty Sudoku board to which they can add numbers to, appeasing to the aim of allowing users to create custom Sudoku puzzles and complete them freely.

The input into the board is done by first selecting a cell and then selecting a digit from the pop-up keypad. The keypad is positioned in the middle of the screen, neutral in colour and non-intrusive. It also allows the user to close the keypad or delete the number present in the cell, if there is one, otherwise it just closes the keypad. If the user selects a number for a cell it will overwrite any value present in it.
The keypad also greys out and prohibits the selection of conflicting values – that is values that appear in the same grid, column or row and therefore are in-eligible for that cell.

It also has additional features that make it an enjoyable and a user-friendly Sudoku app.

The ability for users to save and load puzzles to come back to solve at a later date is a positive addition. Flexibility for the user when completing puzzles is granted which the features of clearing the board or undoing your latest move (unlimited times).

The solver for this application and the output to the user is its main attraction and maps to the aim of incorporating a solver into the system.

Sudoku rules, such as hidden single, x-wing, etc., are used to provide users with solutions to the entered board. The solver is advanced enough to provide all viable solutions given the pre-determined board and displays these neatly in the form of additional Sudoku boards which can be switched to using a simple back and forward arrow position under the board, relating to the aim of outputting solver-specific data.
The filled in numbers from the solver are differentiated from the given clues by a change of colour.

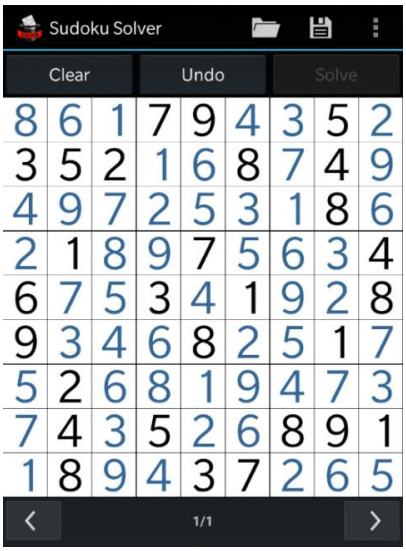


Figure 5-1. Solver's solution to a valid Sudoku (kevinhhl 2013)

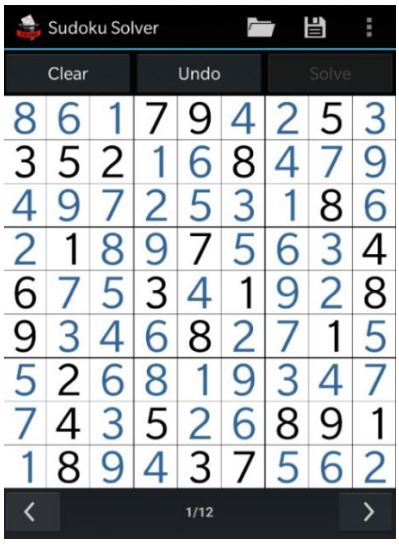


Figure 5-2. Solver's solutions to an in-valid Sudoku (kevinhhl 2013)

However, the app does not generate nor fetch valid Sudoku boards and therefore does not allow varying difficulty boards to be generated which is one of the outlined aims.

However, with its allowable custom inputting and fast solutions it is a hard-core solver application to boards found in medias such as newspapers and magazines where the solutions are not given and may incur a charge for clues or full solutions.

A comment from H. S. on Google Play gave the app 5 stars are stated, “Hi just needed a few clues to complete a hard puzzle, I used this app and saved £1.50 in phone calls.” Highlighting this app’s worth as a solver but a comment from R. L indicates that it is not viewed heavily as a Sudoku playing application, “Sorry but this so missed out on having pencil marks. A very importance feature!” whilst awarding it just 3 stars, under its 4.6 average rating.

2.2.2. Sudoku Solutions

Sudoku Solutions (2018) is an online Sudoku application.

On loading the webpage, the user is presented with 2 distinctive and equally-sized horizontal areas.

Contained within a box at the top is the application: a reasonably sized and presented empty Sudoku board and to the right of it a number of options. Within this box there is a menu option which a number of options.
Beneath it (and an advert) a large text box informs the user about the application and the features of it.

This layout seems wasteful as it takes away from the main feature (Sudoku board) of the webpage by sharing it with text about the application.

In conjunction with the fact that the app has a menu which could easily contain this information for any user in need of it.

This application allows the user to enter values in the grid using keyboard inputs.

It also allows pencil marks to be inputted into cells based on the location of the cell in which the user clicked where top-left is 1 and bottom-right is 9 and ‘snaking’ its way down. These pencil marks can be automatically generated by the program. Conflicting values are highlighted once entered.

The user can start on a blank board, allowing them to custom create a Sudoku puzzle and thus, appeasing one of the aims.

It is also possible for the application to fetch Sudoku stored boards across 4 difficulties: simple, easy, medium and hard and later reference these boards by their unique board number.

This application has an abundance of additional features that make it a real enjoyment for users to work-out Sudoku puzzles, including saving and loading, undo and the ability to determine the difficulty of the board based on the remaining cells to be filled in, etc.

This application can solve or partially solve Sudoku boards which corresponds with the aim of having a solver built into the product.

The application uses Sudoku rules and brute-force when necessary to provide solutions.

The software allows users to get hints for solving the next viable cell, the solution to any given cell or the solutions to all cells which appease selected Sudoku rules.

Solver output is provided by outputting a well-presented, textual step-by-step guide on how it was solved.

This web application can also provide multiple solutions to boards which have more than solution, however the communication with the user that this is the case is not clear and easily missed.

However, in order to get solutions for any given board it must adhere to some constraints such as the board to be solved cannot be empty and must contain at least 16 starting clues.
The application distinguishes starting clues from solver values with varying colour.

Comparing this to the solver of Sudoku Solver highlights the poor layout of multiple solutions and its refusal to provide solutions to boards with minimal clues is underwhelming; giving that the server which this runs off has (presumably) better specs all round than the less than 3MB-sized Android application.

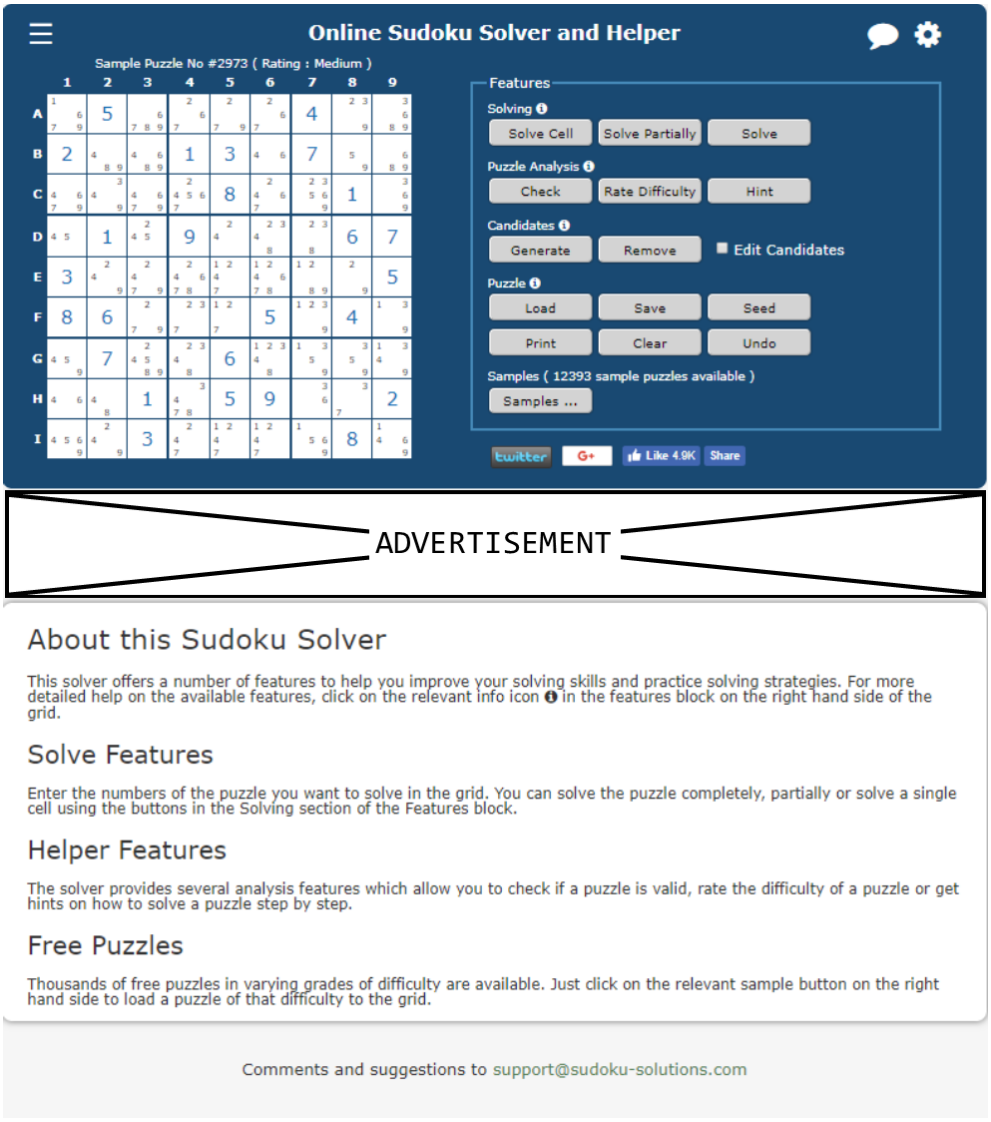


Figure 6. Sudoku Solutions user interface (Sudoku Solutions, 2018)

2.2.3. Pappocom

Prior to his pitch to The Times, Wayne Gould spent 6 years developing a Sudoku application called Pappocom (Smith, 2005) which is a Windows desktop application.

The application use to be a top-of-the-range Sudoku playing application and available commercially for an undisclosed fee, since, however Gould has made it available for free by providing an unlimited key.

After the application is installed and opened, the user is presented with a window pane in which the program runs. This can be enlarged.

This application, in-contrast to the previous one, generates its own Sudoku boards across 6 difficulties, allowing it to remain a stand-alone offline Windows application.

It does allow the user to enter custom Sudoku boards for completing through its 'dubbing' feature, however these must be valid puzzles.

This application allows the user to enter values in the grid using either the mouse or keyboard, allowing this setting to be changed mid-game.

Conflicting values can be highlighted if the option is selected in the game settings.

This application prohibits the alternation of pre-determined clues.

The application also provides the user with the ability to undo the last move they made or selectively remove previously entered values. The board can be returned to the original Sudoku clues by resetting it.

The application provides a number of features related to the solving aim outline for the project.

The application is capable of solving any of its generated boards and provides a nice contrast in colour between: starting clues and user entries.
The application solves its boards using Sudoku rules and as such is able to provide hints to the user for solvable cells.

This application has features which correspond to providing user output; the time taken for a user to solve any particular puzzle is presented via a timer. This feature, like many can be disabled at the user's discretion.

The application provides ample features for users to help complete Sudoku, including: customisation of "35 visual elements" with 3 user pre-sets supported, save and load, print puzzles and recover the previous puzzle.

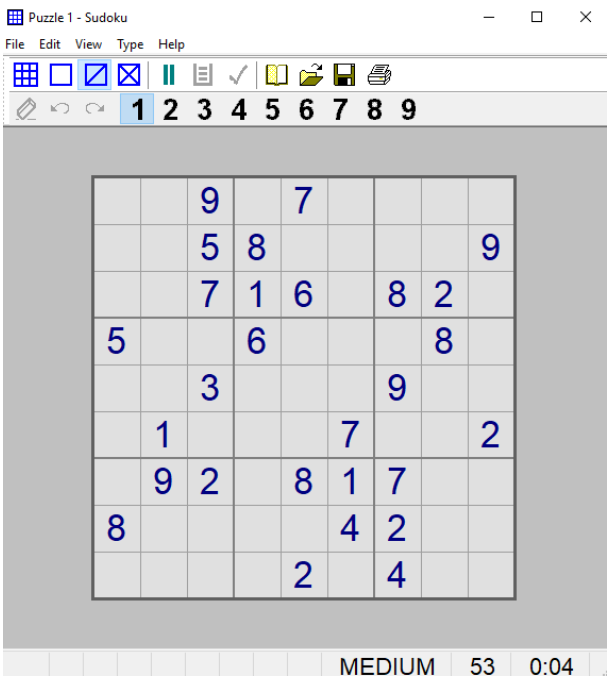


Figure 7. Pappcom (Gould, 2004)

2.2.4. Comparison

Below is a comparison of characteristics that the 3 analysed products and the proposed product have.

Product	*Requires internet connection?*	*Allows invalid boards?*	*Solver Algorithm*	*Solutions displayed* *(invalid boards)*	*16x16 supported?*
Sudoku Solver	✗	✓	Sudoku rules	All	✗
Sudoku Solutions	✓	✓	Sudoku rules, brute-force	All**	✓***
Pappocom	✗	✗	Sudoku rules	N/A	✗
This project	✓	✓	Brute-force*	1	✗

* = backtracking variant
** = not on main UI
*** = with limited features compared to base 9x9 board

Table 1. A summary comparison between similar existing products and the proposed solution against a number of characteristics

3.0 Project Plan

3.1. Stakeholder Identification

There are a number of stakeholders associated with this project.

A stakeholder of this project is the project developer who will be responsible for producing all code and documentation over the course of the project's development and delivering this to the appropriate personnel (if required).

The mentor of the project is also an internal stakeholder, acting as an overseer who provides guidance and support where necessary during bi-weekly 1-on-1 meetings.

An allocated Peer Support Group (PSG) – a group of likeminded developers – is also a stakeholder within the project, offering suggestions, guidance and a beta test group.

Another stakeholder is the mathematician Dr. David Glass. Dr. Glass' interest within the project is the theoretical mathematical aspect regarding the algorithm: its capability and the complexity involved with up-scaling to solve large boards.

The final stakeholder is a group of Sudoku enthusiasts. These individuals will provide the system with its requirements and provide feedback on the incremental product during sprint reviews.

3.2. Requirement Gathering

To derive a product that will be meet the needs of its stakeholders, they must be involved directly in the gathering of the product's requirements.

These candidate requirements are then analysed for ambiguity, technical feasibility and validity amongst other criteria, before being established as requirements for the proposed solution. These established requirements are then assigned as either: functional or non-functional.

Prior to this project being accepted, a proposal had to be drawn-up outlining the goals of the product.

The idea for this project was based off the developer's interest and so they had ideas for system features. The ideas generated from the aims and objectives of the project as well as the developer's past experiences where in-need of extreme validation for feasibility.

Researching into the project area involved looking into existing solutions.

This provided more concrete features because of their implementation in a working product. The project developer began brainstorming features that they had seen and wanted to implement into the project.

These ideal features needed validated for technical feasibility given the project's limited time-scale.

On a bi-weekly basis, the occurrence of a: PSG, 1-on-1 and group mentor meeting ensured that the developer received frequent ideas regarding the product.

However, suggestions from PSG meetings where given by other project developers were often related to their own projects.

In contrast, the suggestions from the project mentor during group meetings were often general to appease most developers' projects.

These suggestions were desperately in-need of being analysed.

In the same timeframe as the meetings above the developer and project mentor had 1-to-1 meetings.

These meetings were an opportunity to bring-up issues or concerns regarding the project. The mentor, with their experience, was able to provide guidance.

In early meetings the developer received ideas for features which were usually based on past projects of a similar nature.

These ideas were often previously not thought of by the developer and in-need of minimal validating because of the clarity and coherent nature in-which they were suggested.

To get the Sudoku focus group involved, interviews were conducted over the course of a week and a half to gather ideas.

This user-group's age ranged from 22-70 and their technology usage ranged as well from inexperienced to those with moderate usage.

During the course of the interviews, notes were taken on their responses to the open and closed questions (see Appendix H). These questions were in relation to a number of key areas that were in-need a greater understanding. However, the interviewer was not confined to these question areas and allowed the interviewees to speak of other topics.

Throughout the interviews the clarity was sought after in vague responses. Their inexperience of the interviewee being a stakeholder in a technical project may have led to vague responses. Obtaining this clarification was, at times, exhausting and led to difficult to read transcripts.

During the interviews the interviewee was presented with a moderately difficult Sudoku puzzle to complete (see Appendix J).

During this they were asked to talk through their actions and the interviewer would ask additional questions based on these. These small interjections allowed them to alter and add to the interview transcript where appropriate.
This unearth previously unknown requirements that were not obtain through the interview alone, e.g. players wrote out the number 1-9 at the bottom of the board and scored them off once they were exhausted from the puzzle. This action was then able to be moulded into a viable suggestion that became of interest to the user.

Finally, a meeting was held with Dr. David Glass to discuss the potential limitations of the project's backtracking algorithm and how to overcome these.

This resulted in the addition of research into the field of complexity and computational complexity with specific regards to Sudoku puzzles.

It is possible for additional requirements to be gathered after these initial gathering activities.

To professionally manage these changes, a change form which must be filled in by the requestor has been created (see Appendix L). These changes undergo the same verification process as outlined prior to this and may be more critically analysed because of the shortened implementation time.

With the short implementation time, change freezes were instigated over the year in-which no changes to the baseline requirements could take place.

3.3. Requirement Prioritisation Strategy

To ensure that the project would be completed on-time the requirements were exposed to a prioritisation technique.

This ensures that the features being worked on at any given instance are worthwhile and contribute to the final product and the final user experience.

Two methods were assessed for suitability within the project: MoSCoW and Weiger's analysis, in which Weiger's analysis was opted for.

This technique provides a mathematically calculated priority value for each feature, allowing features to be ranked relative to one-another.

On the other hand, MoSCoW uses a ranked list that is solely determined by the stakeholder's wants and so is bias to their needs without considering the project's capacity output work. This can lead to rushed software which does not met their needs.

With Weiger's ranked list of features a threshold can then be applied against it to determine which features are included and which aren't.

This threshold can be raised and lowered over the course of the project's lifecycle to exclude/include additional features depending on the progress of the product, again adding to its flexibility for the project.

MoSCoW does not support such flexibility.

Weiger's analysis enables weights to be assigned to individual features, depending on the importance or difficulty meaning that it can be tailored more extensively to this project.

MoSCoW does not utilise weightings.

The author believes that the first two columns of Weiger's analysis are comparable to MoSCoW's entire solution.

However, Weiger goes on to compare the want of the stakeholder's requirement against constraints related to its development producing a much more coherent prioritised list.

3.4. System Requirement Specification

Listed below are the suggestions and ideas for the system as requested by the stakeholders.

These features form the full set of the system's requirements which will be used to derive subset of requirements to obtain a balance between functionality and time known as the requirement baseline.

Identifier	*Requirement Type*	*Requirement*
F1	*Functional*	Puzzles from an API can be fetched as either: easy, medium or hard difficulty
F2	*Functional*	Puzzles can be saved down
F3	*Functional*	Saved down puzzles can be loaded
F4	*Functional*	Clues cannot be altered
F5	*Functional*	Values can be added to cells
F6	*Functional*	Values can be removed from cells
F7	*Functional*	Values can be overwritten in populated cells
F8	*Functional*	Pencil marks can be added to cells
F9	*Functional*	Pencil marks can be removed to cells
F10	*Functional*	Invalid values for cells are recognised
F11	*Functional*	Board can be cleared and made empty
F12	*Functional*	Board can be reset and returned to the originally clues
F13	*Functional*	Timer for users' completion time
F14	*Functional*	80% of valid puzzles can be solved
F15	*Functional*	Hints can be given
F16	*Functional*	Alternative sized puzzles are available
F17	*Functional*	Invalid boards with no solution are recognised
F18	*Functional*	Keyboard shortcuts can be used
NF1	*Non-functional*	Clues are a different colour to values
NF2	*Non-functional*	Pencil marks are small and do not clutter cells
NF3	*Non-functional*	Invalid values are highlighted when attempted to be entered
NF4	*Non-functional*	Timer can be hidden
NF5	*Non-functional*	Values remaining to be filled in are displayed
NF6	*Non-functional*	Exhausted values are highlighted
NF7	*Non-functional*	Any difficulty puzzle is solved inside 1 minute

NF8	*Non-functional*	Web interface should be initiative to use
NF9	*Non-functional*	Web interface should be consistent across popular web browsers
NF10	*Non-functional*	Web interface should be consistent across webpages
NF11	*Non-functional*	Application should be able to be updated without being taken offline
NF12	*Non-functional*	Difficulties connecting to APIs should be handled
NF13	*Non-functional*	Application should be well-tested

Table 2. Requirement Specification

3.5. Software Life-Cycle Methodology

For the development of this project to be done in a professional, logical and structured manner, it must adhere to follow a software development life-cycle (SDLC) methodology.

The main choices considered where: waterfall, agile and prototyping.

A blend of plan-driven and agile methodologies were chosen for this project having been analysed for their suitability within this project.

Given the project's time-constraint agile is extremely well-suited for delivering results because of its use of sprints.

These sprints allow for continuous delivery of software to the stakeholders on a frequent basis.

The software lifecycle technique must also reflect the personnel resource allocated to any project.

Agile is well-suited for development small teams however, it is most suited than the other options for a one-person development project.

Providing the stakeholders with small functional pieces of software allows for feedback and the potential of changes to the original plans.

Agile is excellent at adapting initial requirements and implementing change.

This gives the project flexibility in altering the baseline of requirements gathered including any form of requirement prioritisation techniques which will be used throughout the project.

Agile is extremely suited for this project's 2-phase implementation as it allows for a functional prototype to be developed, demonstrating some project objectives.

To manage an agile development process there are two primary options: Scrum and Kanban.

Considering the project, it has been concluded that Kanban is best suited.

Kanban is suitable for a sole developer project as it allows for appropriate work in-progress (WIP) limits to be set to restrict the amount of work items that can be undertaken at any one time.

Somerville (2011, p. 64) states that a balance must be established when using in conjunction a plan-driven and agile development practise and that balance is decided by the needs of the stakeholders.

The project will initially undergo plan-driven development and will partake in the initial stages of it which includes extensive documentation including planning charts such as a Gantt chart. This methodology will then encompass an agile approach during implementation.

The project will have sprints of around 2 weeks which ensures continuous development and testing of system features and delivering them to the appropriate stakeholders.

3.6. Implementation Plan

3.6.1. Work Breakdown Structure

Figure 8 below shows the Work Breakdown Structure (WBS) for the proposed system which was developed using a top-down approach.

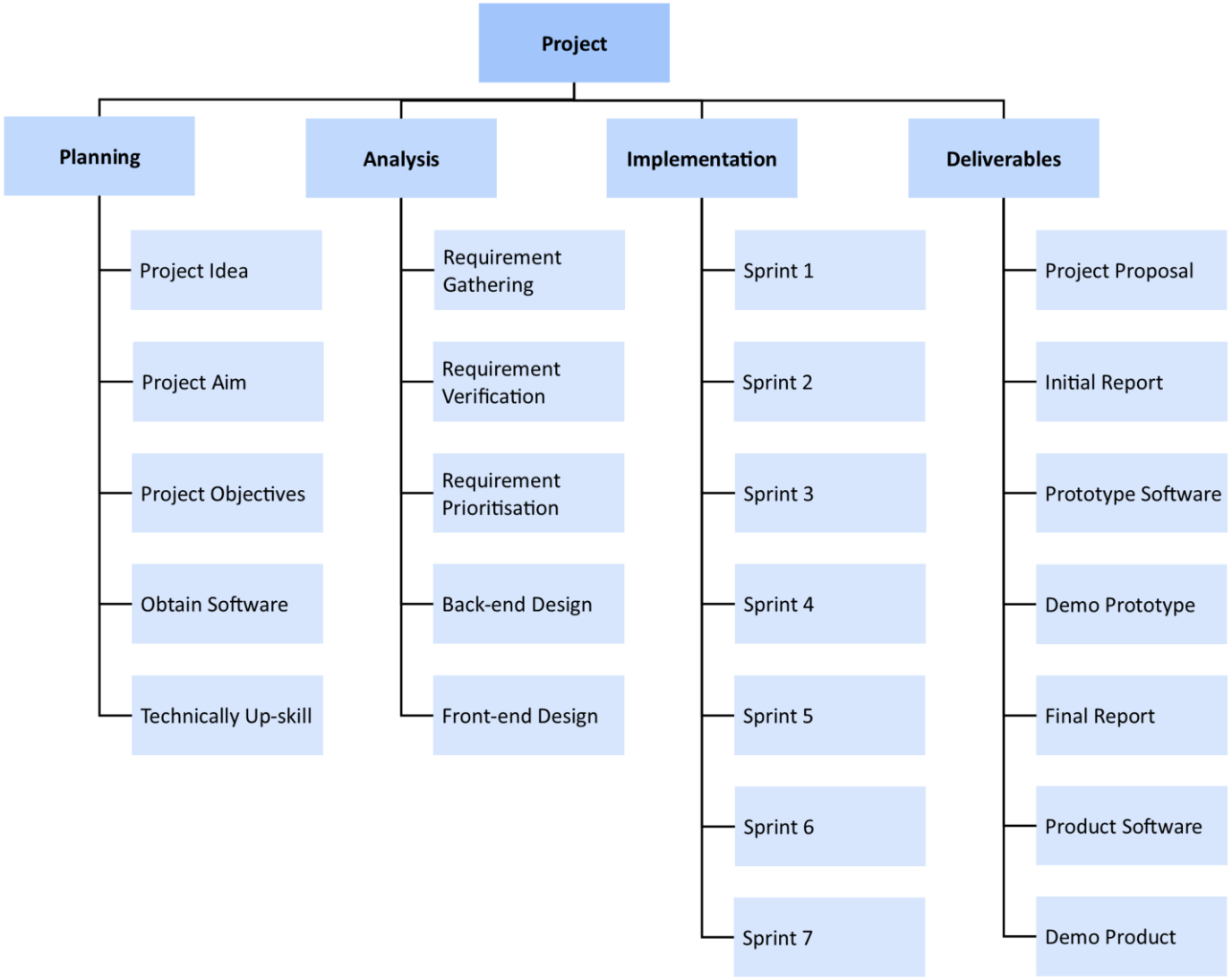


Figure 8. Work breakdown diagram for the project

Table 3 below highlights the delivery dates for each of the WBS' tasks.

Planning	*Wednesday*	*31st October 2018*
Project Idea	Friday	28th September 2018
Project Aim	Tuesday	9th October 2018
Project Objectives	Friday	12th October 2018
Obtain Software	Monday	29th October
Technically Up-skill	Wednesday	31st October 2018
Analysis	*Friday*	*9th November 2018*
Requirement Gathering	Friday	2nd November
Requirement Verification	Thursday	8th November 2018
Requirement Prioritisation	Friday	9th November 2018
Back-end Design	Friday	9th November 2018
Front-end Design	Friday	9th November 2018
Implementation	*Friday*	*12th April 2019*
Sprint 1	Friday	23rd November 2018
Sprint 2	Wednesday	5th December 2018
Sprint 3	Friday	1st February 2019
Sprint 4	Friday	15th February 2019
Sprint 5	Friday	1st March 2019
Sprint 6	Friday	29th March 2019
Sprint 7	Friday	12th April 2019
Deliverables	Friday	31st May 2019
Project Proposal	Friday	28th September 2018
Initial Report	Thursday	6th December 2018
Prototype Software	Thursday	6th December 2018
Demo Prototype	Friday	14th December 2018
Final Report	Friday	19th April 2019
Product Software	Friday	19th April 2019
Demo Product	Friday	31st May 2019

Table 3. Project phases delivery dates

3.6.2. Gantt Chart

Task Name	Duration	Start	Finish
Change Freeze - Winter	**25 days**	**Mon 12/11/18**	**Fri 14/12/18**
Change Freeze - Spring	**20 days**	**Mon 18/03/19**	**Fri 12/04/19**
Proposal Accepted	0 days	Fri 28/09/18	Fri 28/09/18
Planning	**33 days**	**Mon 17/09/18**	**Wed 31/10/18**
Project Ideas	8 days	Mon 17/09/18	Wed 26/09/18
Project Aim	7 days	Mon 01/10/18	Tue 09/10/18
Project Objective	3 days	Wed 10/10/18	Fri 12/10/18
Gather Software	19 days	Wed 03/10/18	Mon 29/10/18
Tech Upskilling	21 days	Wed 03/10/18	Wed 31/10/18
Requirements Defined	0 days	Fri 09/11/18	Fri 09/11/18
MVP Defined	0 days	Sat 24/11/18	Sat 24/11/18
Analysis	**20 days**	**Mon 15/10/18**	**Fri 09/11/18**
Requirement Gathering	13 days	Mon 15/10/18	Fri 02/11/18
Requirement Verification	4 days	Mon 05/11/18	Thu 08/11/18
Requirement Prioritisation	1 day	Fri 09/11/18	Fri 09/11/18
System Back-end Design	5 days	Mon 05/11/18	Fri 09/11/18
System Front-end Design	5 days	Mon 05/11/18	Fri 09/11/18
Back-end	0 days	Wed 05/12/18	Wed 05/12/18
Intermediate	0 days	Fri 01/02/19	Fri 01/02/19
API	0 days	Fri 01/03/19	Fri 01/03/19
Front-end	0 days	Fri 05/04/19	Fri 05/04/19
Implementation	**80 days**	**Mon 12/11/18**	**Fri 12/04/19**
Sprint 1	**10 days**	**Mon 12/11/18**	**Fri 23/11/18**
Sprint 2	**8 days**	**Mon 26/11/18**	**Wed 05/12/18**
Sprint 3	**10 days**	**Mon 21/01/19**	**Fri 01/02/19**
Sprint 4	**10 days**	**Mon 04/02/19**	**Fri 15/02/19**
Sprint 5	**10 days**	**Mon 18/02/19**	**Fri 01/03/19**
Sprint 6	**10 days**	**Mon 18/03/19**	**Fri 29/03/19**
Sprint 7	**10 days**	**Mon 01/04/19**	**Fri 12/04/19**
Deliverables	**155 days**	**Mon 17/09/18**	**Fri 31/05/19**
Prototype	0 days	Thu 06/12/18	Thu 06/12/18
Prototype	0 days	Fri 14/12/18	Fri 14/12/18
Winter	**65 days**	**Mon 17/09/18**	**Fri 14/12/18**
Initial Report	59 days	Mon 17/09/18	Thu 06/12/18
Prototype Software	19 days	Mon 12/11/18	Thu 06/12/18
Demo Prototype	5 days	Mon 10/12/18	Fri 14/12/18
Product	0 days	Fri 12/04/19	Fri 12/04/19
Product	0 days	Fri 31/05/19	Fri 31/05/19
Spring	**90 days**	**Mon 14/01/19**	**Fri 31/05/19**
Final Report	60 days	Mon 14/01/19	Fri 19/04/19
Product Software	55 days	Mon 21/01/19	Fri 19/04/19
Demo Product	5 days	Mon 27/05/19	Fri 31/05/19

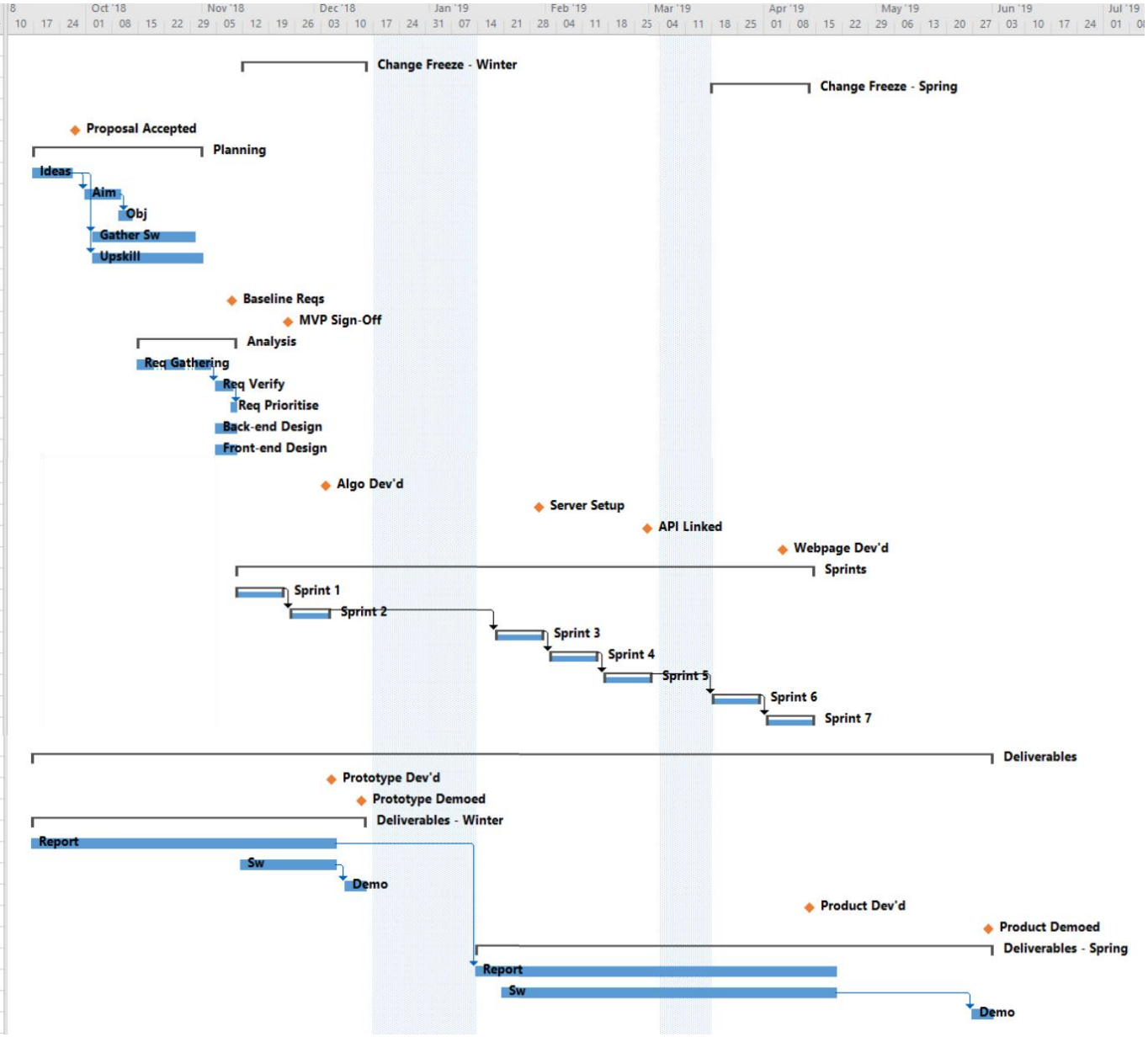


Figure 9. The project's Gantt Chart

The change freezes (identified as Winter/Spring) are time-frames in which no alterations can be made to the set of baseline requirements.

The holidays (identified as the blue bars) are time periods in-which the developer is not working on the project.

The deliverables (identified as Winter/Spring) are deadlines set for the project regarding its documentation, code, demonstrations and product.

Sprint setups are outlined in Appendix C.

3.6.3. Resources Identification

Table 4 below shows the planned hardware and software required to develop the system.

Hardware:

Server machine:

PC or laptop

- Standard specs*

*Solving for larger boards may require the improvement in CPU and RAM for the algorithm engine as it becomes more computationally intense

Software:

Operating Systems (OS):

Linux

- 32-bit v16.0.4
- This project will use a VirtualBox virtual machine (VM) to run Ubuntu 32-bit off a non-native Linux OS

Programming Languages:

Python

- 32-bit v 2.7
- This project will use Python for its back-end server

q

- 32-bit v3.6
- This project will use q for its back-end engine

Javascript (JS) / HTML / CSS

- This project will use these trio of languages to create a dynamic front-end

Frameworks/Plugins/ Libraries:

pyq

- This project will use the pyq plugin on Python to communicate between the algorithm engine and Flask server

rlwrap

- This wrapper library will be used to allow for the navigation of past executes in q

log4q

- This library will be used to enable the algorithm to log its standard output and error

Table 4. The hardware and software components required for this project

3.7. Verification Plan

To verify all of the product's deliverables they must be extensively analysed to ensure that they meet the stakeholders' baseline requirements.

This requires that the requirements attained are in-fact, the baseline requirements which will ensure that, if implemented, they will fully meet the needs of the stakeholders.

During the requirement gathering phase, the business analyst (BA) held interviews in which a series of open and closed questions were asked. This allowed for specific areas to be queried about as well as provided the interviewee to provide their thoughtful insight into more broad areas.
After the initial set of interview questions were asked in a professional manner, the BA provided the stakeholder with a Sudoku puzzle and asked for them to attempt to complete it.
Whilst they worked on the puzzle the BA was observing their actions and taking notes, further expanding on points asked during the initial interview.
These multiple gathering techniques were done on all identified stakeholders and as a result provided a range of needs for the proposed system. Carrying out these in a professional and systematic manner ensures that the suggestions provided, once extensively analysed for: redundancy, ambiguity and incompleteness, produced a complete requirement specification for the planned product.

Throughout the project's lifecycle (excluding change free periods) the stakeholders are free to submit a change form if the feel that the planned system is missing a feature.

This form undergoes an acceptance process in which the suggestion is analysed.
During this process the stakeholder may be contacted to explain any unclear areas within the request. This open communication with requirements is the same for determined requirements.

All deliverables from the system must met their initially set-out requirements.

Over the course of the project's agile approach, small pieces of functionality can be demoed to stakeholders to show progress of the system and how these changes meet the requirements chosen for that sprint.
These demos allow the stakeholders to make amendments to requirements executed which where perhaps misinterpreted during their gathering.

Acceptance cases for functionalities of the system were defined during interviews with the stakeholders and reinforced during bi-weekly sprint demos.

The performance of the solving algorithm is a key aspect of the product and so was outlined as its own separate requirement.
These cases allow for the system to be tested against its expectations from the user and so can determine whether the requirements are fully met.

Requirement NF12 states that the stakeholders are interested in the system being well tested and so test cases will be created and delivered to them.

Planned test cases are outline below but these are to be expanded upon and given in greater detail later in the project.

3.7.1. Test Cases

Table 5 below shows the planned test cases to verify each requirement defined in section 3.4 has been implemented to an acceptable standard.

Req#	*Test#*	*Action*	*Expected Result*	*Whitebox or Blackbox*	*Valid Test Case*	*Extreme Test Case*	*Erroneous Test Case*
F1	T1	Fetch easy puzzle from API	Easy board is found, returned and populated	Blackbox	Select to retrieve easy board with the on-screen Sudoku board being blank	Select to retrieve easy board with the on-screen Sudoku board not being blank	N/A
F2	T2	Save puzzle from GUI	User selects directory to save to and the board is saved	Blackbox	Select to save down when the on-screen Sudoku board is partially completed	Select to save down when the on-screen Sudoku board is blank or completed	N/A
	T3	Load puzzle from GUI	User selects a saved down board and the board is populated	Blackbox	Select to load a previously saved down board	Select to load a file that has been emulated to resemble a saved down board	Select to load an incorrect file that is not of the correct type

F3	T4	Fetch easy puzzle from API	All cells retrieved are identified as clues	Blackbox	Retrieve an easy board from the API	N/A	API connection error data sent back
	T5	Load previously saved board from API	Cells that should be clues are identified as so	Blackbox	Load a previously saved down board with only clues	Load a previously saved down board with clues and partially completed values	N/A
	T6	Load previously saved custom board	Cells that should be clues are identified as so	Blackbox	Load a previously saved down board with only clues	Load a previously saved down board with clues and partially completed values	N/A
F4	T7	Attempt to delete clue	Clue remains	Blackbox	Use backspace or delete key	N/A	Paste whitespace
	T8	Attempt to overwrite clue	Clue remains	Blackbox	Use different number	Use same number	Paste number
F5	T9	Add a number to a cell	Number is added to cell	Blackbox	Use valid Sudoku number, i.e. 3	Use largest valid Sudoku number, i.e. 9	Use program specific null number, i.e. 0
	T10	Remove a number from a cell	Number is removed from cell	Blackbox	Use backspace or delete key on a populated cell	Use backspace or delete key on an empty cell	Paste whitespace
F6	T11	Overwrite number in a cell	Number is replaced with overwriting number	Blackbox	Use different number	Use same number	Paste number
F7	T12	Add pencil mark to cell	Pencil mark for number is added to cell	Blackbox	Use valid Sudoku number, i.e. 3	Use largest valid Sudoku number, i.e. 9	Use program specific null number, i.e. 0
	T13	Remove pencil mark to cell	Pencil mark for number is removed from cell	Blackbox	Remove pencil number present	N/A	Remove pencil mark not present

F8	T14	Enter value in a cell which is already in grid, row or column	Number is identified as being invalid	Whitebox	Enter value already in grid, row or column	Enter value only present in one of grid, row or column	N/A
F9	T15	Select to clear board	Board is made empty	Blackbox	Clear partially completed board	Clear empty / completed board	N/A
F10	T16	Select to reset API sourced board	Board is returned to its original clues	Whitebox	Reset partially completed API sourced board	Reset only sourced API board	N/A
	T17	Select to reset custom board	Board is returned to its locked-in clues	Whitebox	Reset partially completed clue locked-in board	Reset clue locked-in only board	N/A
F11	T18	Enable application timer	Timer is displayed and begins counting	Blackbox	Start timer	Start timer after mapped key is pressed	N/A
F12	T19	Select to solve valid board	Board is solved with a valid solution	Whitebox	API sourced board	Custom board with clues locked-in	Invalid or completed board
	T20	Select to solve varying difficulty boards	Board is solved with a valid solution	Whitebox	API easy difficulty	API hard difficulty	N/A
F13	T21	Select to provide hint	Cell's value is revealed	Whitebox	Cell with no value is selected	N/A	Cell with value is selected
F14	T22	Select square sized puzzle	On-screen board is sized accordingly	Blackbox	9x9	16x16	N/A
	T23	Select lob-sized puzzle	On-screen board is sized accordingly	Blackbox	6x2	5x3	N/A
F15	T24	Select solve on invalid board	Solver recognizes that it is unsolvable	Blackbox	Custom board	API sourced board with user values	N/A

F16	T25	Mapped key is pressed	Action is carried out	Blackbox	Key mapped is pressed	Key mapped is pressed along with another key	Key not mapped is pressed
NF1	T26	Timeout connection to API occurs	Handled and error message appears	Blackbox	API is down	API is busy	N/A
NF2	T27	Clues are given on a board	Numbers appear in a different colour from user values	Blackbox	Board sourced from API	Custom board with locked-in clues	Custom board with no locked-in clues
NF3	T28	Pencil marks are entered into a cell	Numbers are small and slightly transparent	Blackbox	One pencil mark is entered into cell	All pencil marks are entered into cell	N/A
NF4	T29	Invalid number is attempted to be entered into the cell	Number is highlighted	Blackbox	Invalid number is entered into cell	Invalid number is entered twice	N/A
NF5	T30	Select to hide timer	Timer is no longer visible on the UI	Blackbox	Timer is hidden from default appearance	Timer is hidden after being shown	N/A
NF6	T31	Board size is selected	Numbers used in the puzzle are presented	Blackbox	Empty 9x9 board	Empty other sized board	N/A
	T32	Board is fetched / entered	Numbers that are exhaustive are highlighted from those presented	Blackbox	API fetched board	Custom entered board	N/A
	T33	Last occurrence of a number is entered into last grid, row and column	Number is indicated as being exhaustive	Blackbox	Number with only one occurrence left	Last number of the puzzle	N/A
	T34	Number which was previously exhaustive is removed from a cell	Number is no longer identified as being exhaustive	Blackbox	Number which is identified as exhaustive	N/A	N/A

NF7	T35	Select to solve valid puzzle	Puzzle is solved inside 1 minute	Blackbox	Partially filled in 9x9 board	Hard 9x9 board	N/A
NF8	T36	Use application	Interface is logically laid out	Blackbox	Components' positioning is questioned	Missing components are speculated	N/A
NF9	T37	Use application in common web browsers	Interface should be identical across all	Blackbox	Application is used across multiple browsers	Application is different browsers are side-by-side compared	N/A
NF10	T38	All application webpages are viewed	Webpages should follow a similar layout and theme	Blackbox	Webpages are viewed logically	Each webpage is compared to each other	N/A
NF11	T39	Update is posted to the application server	Algorithm has up-to-date changes	Whitebox	Check updates are updated after refresh	Check updates are updates with no refresh	N/A
NF12	T40	Application is used vicariously during both alpha and beta testing	No unexpected side effects are discovered	Whitebox	Standard web users are exposed to the application	Experienced web users are exposed to the application	N/A

Table 5. Initial test cases

3.8. Validation Plan

To validate the project, it must function correctly within its expected working environment whilst being operated on by its anticipated end users.

The users with regards to the project are typical Sudoku users.

These users will vary mainly in age and technological familiarity.

The fluctuation in age can result in differences in health conditions which can directly affect the user's experience when using computers, e.g. eye strain. This must be taken into consideration when designing the UI when choosing fonts and colours.

Users of this product will have differences in technology usage. This means that the final product should be intuitive to use and require minimal additional downloads to run. The layout of the delivered product must follow a logical structure.

The usage of the product will differ from user-to-user and will range from a daily to one-off basis.

The expected environment that this product will run in, with regards to the identified end users, is in a building with the application running on a typical PC.

The building in question will most likely be a home or office surrounding which share many characteristics when using a computer like ventilation and lightening from either natural sources, e.g. airflow and sunlight or artificial, e.g. air-conditioning and lighting fixtures. Characteristics such as these can extend continuous usage of computers and reduce strains such as screen glare.

The machine used to develop the product is a typical middle-of-the-line spec laptop; 3.3GHz processor and 8GB of RAM. The computational power required to the product will be significantly reduced as most of the heavy lifting (server hosting and algorithm running) is down on the host machine. Conor from CNET (2017) states that the minimum specification for a PC translates to a 2.1GHz processor and 4GB of RAM which is suitable for users of the product.

The product is also flexible with input devices.

International keyboards – QWERTY, QWERTZ and AZERTY, with or without numpads, will work fine with this product as the product's core functionality is related to its number keys.

The pointer device used is also flexible as the product supports typical trackpads, mice and specialist trackerballs and other ergonomic devices.

The product will be outputted on a screen.

The screen can be a standard LCD monitor or CRT with the brightness and contrast adjusted to the user's personal use.

The product will require a reliable internet connection either through a wireless router or eternal LAN configuration.

Users in-use of a computer in a working environment or home-life will have this connection setup.

To validate the product against these considerations a prototype will be produced with core functionality and presented to an internal panel of developers and the expected end users for functional analysis and suitability to these constraints.

The prototype will be demoed to the end users directly and will receive feedback. This allows the developer to get insight into the suitability of the rest of the proposed system. The users can express whether the system is intuitive in its concept and whether the proposed UI is user friendly in design, layout and theme.

This demo (along with the one to the internal panel) will take place in a similar and comparable environment to that which has been outlined. The room will be well-ventilated with an abundance of natural light.

The demo will run off the developer's laptop and does not require access to the internet. The prototype software will have its IDEs, framework, plugins pre-installed.

Validation will also take place during the acceptance testing at the end of each sprint (see Appendix C).

These meetings will consist of showcasing the piece of software developed over the sprint and whether it is up-to-standard. All comments received from the stakeholders during these will be taken on-board and acted upon.

4.0 Risk Assessment

4.1. Risk and Mitigation Strategy

Table 6 below highlights the identified project risk identified and how to mitigate them.

Risk	*Likelihood of Risk Occurring*	*Impact of Risk on Project*	*Mitigation Strategy*
Algorithm cannot solve Sudoku puzzles expected of it			• Conduct research • Seek assistance from senior developers • Setup extensive testing
The API used to source puzzles is not available or changed during implementation			• Have a backup API
Unable to connect the system components together			• Seek online assistance
Not understanding the technologies being used			• Have a backup technology language or framework for each system component • Make use of online tutorials and help • Seek assistance from senior developers
Balancing other workloads			• Have deadlines marked and manageable work items alerted via mobile calendar reminders

Table 6. Risk omitting strategies

The risk of not having an algorithm that cannot solve these kinds of puzzles is made redundant due to the extensive research conducted prior to the project's initial implementation.

In addition, the support and guidance from Dr. Glass on the complexity of the problem and the limitations of the algorithm in solving that problem has also rendered this risk mute.
The developer will also have access to senior developers within the project's implementation environment who have overseen projects of a similar nature and so can provide assistance if required.
Creating detailed test cases for all types of puzzle will ensure the algorithm's validity in solving them.

If the API used is unavailable or changed significantly during the project's implementation stage, this can result in the project unable to source Sudoku puzzles, reducing the user's experience of the application.

To reduce this an alternative API for the provision of valid Sudoku puzzles with a range of difficulty will be sourced.

Currently, the developer is undergoing a relatable full-stack development course which will provide assistance and guidance when connecting all of the system components together.

The Python Flask server framework which is being used is well tested and renowned for acting as an intermediate between varying technologies and such should support the technologies being used in this project.

Having a backup language or framework for each system component, in which the developer is more experienced with will reduce the risk of incompetency in any of the technologies used during this project.

In addition, the use of online support groups and extensive documentation can up-skill the developer in weaker areas. This ensures that the end project is functional and understandable to the developer allowing for quick bug fixes or future features or scalability to be implemented easily.

Knowing upcoming deadlines and splitting up work into manageable workloads allow the developer to plan their time efficiently and make informed decisions such as: the need to work late, attend help classes, etc.

This can then reflect in a daily or weekly to-do list which can help keep them on-track. Doing this prevents the developer missing deadlines and delivering work late.

The creation and demonstration of a prototype which encompasses some core functionality for the project will omit its defined risk identified above in table 6 .

5.0 Prototype

5.1. Risk

Throughout the planning phase of this project it was identified that the method of risk mitigation was done by applying Weiger's prioritisation against the requirement specification to uncover the necessity of each.

A prototype of a small, functional part of the system was scheduled to be delivered on Thursday 6th December to mitigate against a known risk of the project.

This is a piece of functionality that is critical to the success of the product and was identified by using Kano's interpretation of 'basic expectations' with regards to the product.

Therefore, the risk being omitted by being demonstrated during the prototype session is the ability for the product to solve varying Sudoku puzzles.

This requirement did not receive the highest priority using Weiger's analysis, however its beneficial and detrimental impact on the product was identified greatly. Selecting this functionality as the risk to be omit ensures that the product will meet its basic expectations, beneficially impacting its validity to its end users.

This risk directly impacts on several requirements (F12, F13, F14, F15 and NF6) and is regarded as the most technologically difficult aspect of the project from the developer's point of view.

Therefore, addressing this early in the project's lifecycle will ensure its delivery as the developer is fresh and eager to work through challenges it throws up.

Developing an early version of the algorithm will allow time for its efficiency to be improved which effects the end product's validity in its end environment.

The prototype is to be demoed to an internal panel which perfectly fits in with the project's agile life-cycle approach.

During this demoing, the product will be validated in a similar end environment and therefore issues relating to its validity can be brought up and addressed early in the product's development phase.

5.2. Design Artefacts

The design artefact presented below is of the backtracking algorithm developed in the functional programming language q.

This algorithm will be embedded in a Python Flask server using the plugin pyq.

The pseudo code for the algorithm (see figure 10) is the following:

```
Take the inputted board

Check for null cell
   If there are none, return solved
Assign the cell with a value
greater than it is

Check for conflict
   If there is, check that the
   board is invalid
      If it is, terminate
Assign recursively
```

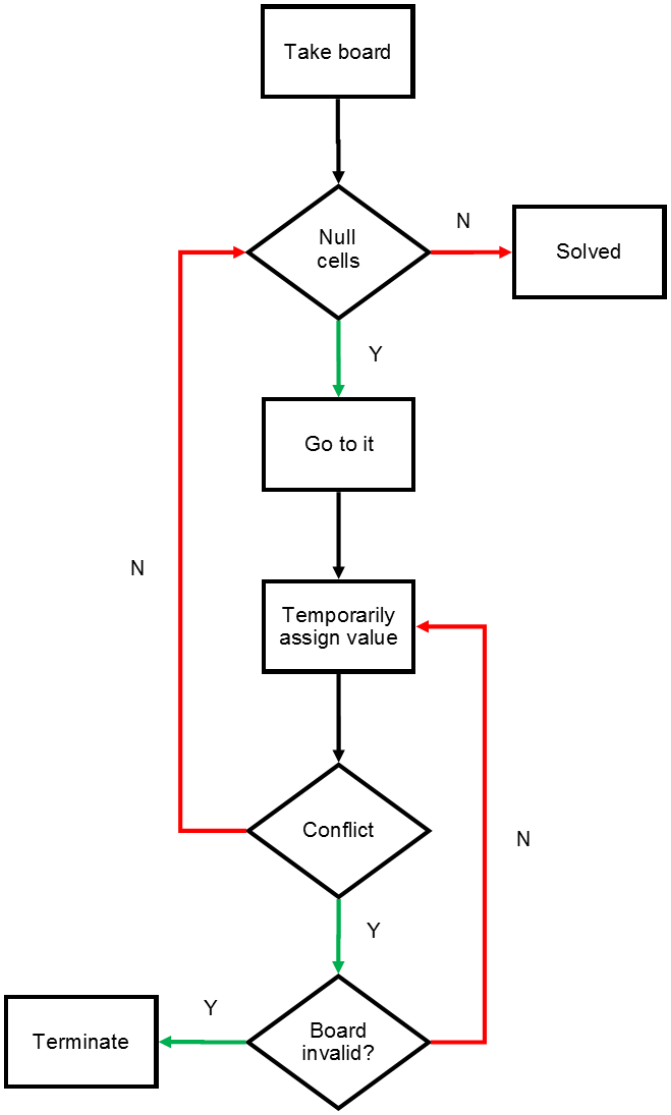


Figure 10. Recursive algorithm design

6.0 Project Management

6.1. Requirements Management

6.1.1. Requirements Evolution

Throughout the processes of attaining, validating and managing requirements there have been some alterations.

These changes are a result, in-part, of adhering to professional standards which the developer has been formally taught on over the course of the project's early stages. These include ensuring that requirements are atomic and stand by themselves as unique features or characteristics, non-ambiguous and demonstrate a clear and concise need for the stakeholder which helps ensure that they are testable, either directly or indirectly and with the distribution of a change form (see Appendix L) to each stakeholder, requirements can be created, altered and deleted with adequate traceability.

Initial requirements were refined over the course of the project's infancy during brainstorming sessions with the developer and their experienced project mentor.

These sessions looked at removing ambiguity and atomicity from the initial requirements by questioning the wording, structure and underlying aim of each requirement. This helped to make them testable too.

Appendix M shows a change form which shows a feature that was wanted by the stakeholder for the final product.

The stakeholder has identified that when using existing online Sudoku solvers to validate their completed board, seeing the solution board isn't clear. One reason for this is the other elements on the webpage distract them. In addition, using the web browser's zoom feature poses a problem because it is relatively unknown to many older users who have no known way of returning the screen to its original zoom level after, making the experience on the website confusing and frustrating.

This feature was accepted into the final product because it exceeded the priority threshold of 0.336 at the time when it was returned to the developer.

Appendix N shows a change form with a feature that was rejected because it did not meet the threshold when it was returned.

The choosing of an agile project methodology allowed for such flexibility because it is one of the key principles in its manifesto, "welcome changing requirements, even late in development" (Agile Alliance, 2019)

6.1.2. Baseline Requirements

The baseline requirements for this project are outline below in table 7.

These consist of the needs and desires of the stakeholders which exceeded a given threshold. This threshold was derived using Weiger's analysis on the complete set of requirements and then taking the median value.
The two Weiger's priorities and risks in table 7 indicate these values over time.

The baseline requirements were changed part-way through the project.

The initial baseline requirements were derived from first set of requirements whose threshold came out as 0.697. These set of 15 requirements were agreed between the stakeholder and the developer to be included in the final product.
However, the developer was handed 2 change forms with the hope of implementing a further 2 features (see Appendixes M and N) in January/February.

The calculation to work out the priority of each of these was modified slightly, mainly that the weighting for risk was increased 5-fold because of the limited remaining time.

This change in weightings affected the existing priority scores for requirements and as such the new threshold was 0.337.
The change in weightings and subsequently the threshold meant that requirement R8 was now below the adjusted priority threshold, however because it was agreed in November the feature still has to be implemented.

Requirements of a well-tested application and exception handling were not included in the prioritisation criteria because the project will undergo professional coding practises when being developed, including extensive: error handling and testing.

Identifier	*Requirement Type*	*Requirement*	*Weiger's Priority*	*Weiger's Risk*	*Risk Assessment*
R1	*Functional*	Puzzles from an API can be fetched as either: easy, medium or hard difficulty	**0.857** (10) **0.342** (15)	**6** (3) **6** (3)	If this requirement isn't met then the user will be unable to fetch puzzles to play, hindering the program's usage of a Sudoku application
R2	*Functional*	Clues cannot be altered	**0.697** (15) **0.360** (12)	**3** (10) **3** (10)	If this requirement isn't met then verified clues can be altered which will may lead to previously legit puzzles being either: unable to solve – if the clue was overwritten by an erroneous value or, have multiple solutions – which is not a characteristic of authentic Sudoku puzzles
R3	*Functional*	Values can be added to cells	**0.998** **(2)** **0.634** (1)	**2** **(13)** **2** (13)	If this requirement isn't met, then the user will not be able to add values to cells and so they could not enter boards to be solved or play a game of Sudoku
R4	*Functional*	Values can be removed from cells	**0.998** (2) **0.634** (1)	**2** (13) **2** (13)	If this requirement isn't met, then the user would be unable to remove erroneous values and therefore unable to enter a correct board to be solved or correct values when playing
R5	*Functional*	Values can be overwritten in populated cells	**0.877** (9) **0.544** (2)	**2** (13) **2** (13)	If this requirement isn't met, then the user would have to firstly remove the current value in the cell before entering their desired one, making the application slightly cumbersome
R6	*Functional*	Pencil marks can be added to cells	**0.853** (11) **0.398** (8)	**4** (6) **4** (6)	If this requirement isn't met, then users who use pencil marks as a means of working out harder puzzles will be unable to and left disappointed
R7	*Functional*	Pencil marks can be removed from cells	**0.853** (11) **0.398** (8)	**4** (6) **4** (6)	If this requirement isn't met, then the user would be unable to remove erroneous values which will make working out more challenging boards cumbersome

R8	*Functional*	80% of valid puzzles can be solved	**0.815** (14) **0.280** (16)	**9** (1) **9** (1)	If this requirement isn't met then it is more likely that the solver will be unable to solve boards from time-to-time, reducing its effectiveness as a Sudoku solver
R9	*Non-functional*	Clues are a different colour to values	**0.948** (5) **0.480** (4)	**3** (10) **3** (10)	If this requirement isn't met, then clues and user entered values will be impossible to differentiate which will make adjusting erroneous values difficult
R10	*Non-functional*	Pencil marks are small and do not clutter cells	**0.930** (6) **0.438** (5)	**4** (6) **4** (6)	If this requirement isn't met, then having multiple pencil marks in one cell will become untidy and difficult to read cells' values or pencil marks
R11	*Non-functional*	Values remaining to be filled in are displayed	**0.879** (8) **0.398** (8)	**4** (6) **4** (6)	If this requirement isn't met, then the explicit values to be entered into the board are not clearly identified which may make it less intuitive especially for new players
R12	*Non-functional*	Exhausted values are highlighted	**1.031** (1) **0.528** (3)	**3** (10) **3** (10)	If this requirement isn't met, then users may spend time looking to insert an exhausted value into the board leading to loss of time, mistakes and frustration
R13	*Non-functional*	Solvable puzzles are solved inside 1 minute, regardless of difficulty	**0.912** (7) **0.343** (14)	**7** (2) **7** (2)	If this requirement isn't met, then users will become agitated waiting for a puzzle's solution and may search for an alternative application elsewher
R14	*Non-functional*	Clicking a solved board displays it in a new tab	**N/A** **0.407** (7)	**N/A** **1** (16)	If this requirement isn't met, then users will struggle to validate their attempt against the solution
R15	*Non-functional*	Web interface should be initiative to use	**0.964** (4) **0.408** (6)	**5** (4) **5** (4)	If this requirement isn't met, then the application will not be friendly to new players or users with less technologic usage

R16	*Non-functional*	Web interface should be consistent across webpages	**0.843** (13) **0.357** (13)	**5** (4) **5** (4)	If this requirement isn't met, then users will become confused and frustrated when using the application
R17	*Non-functional*	Difficulties connecting to APIs should be handled	**N/A**		If this requirement isn't met then the application may freeze, crash or present erroneous data as a board
R18	*Non-functional*	Application should have 80% of its functionality tested and passing	**N/A**		If this requirement isn't met, then the application may not function as expected leading to the user becoming frustrated and leave to search for an alternative

Table 7. Baseline requirements

6.2. Plan Management

It was uncovered during the prototype's demonstration that testing the performance of the algorithm could prove to be inefficient by merely running through a series of puzzles, as had been done during its development.

This was unveiled by the internal panel of developers who suggested that a web scraper could be developed to retrieve a substantial number of puzzles before running them through the algorithm and comparing their solutions with the algorithm's output for equality. This would provide extensive validation testing as the puzzles are randomly sourced across a variety of difficulties (identical to the difficulties the product will offer) and are likely to be similar to those inputted by users.

A scraper was therefore decided to be implemented.

This scraper was scheduled to be developed in the adjacent sprint after the demo (sprint 3). This was chosen to allow any algorithm performance issues to be identified early and rectified as soon as possible, given that the algorithm is the primary feature of the project.

During this testing phase a memory limitation issue within the programming language was uncovered.

The limitation revolved around stack memory which became full of recursive functions during certain Sudoku puzzles. This problem was apparent with 388 of the 1,000 sourced puzzles which resulted in a 61.2% success rate, short of the desired 80%.

There were several ways in which to address this shortcoming. These possible solutions are outlined and compared in table 8 below.

SHORT-TERM

rotate the board and solve for each side

Advantages:

- quick to implement
- viable, due to Sudoku's symmetry and fundamental principals

Disadvantages:

- 'hack' work around which reduces professionalism of the algorithm
- adds more recursive calls to solve from each side
- doesn't prevent stack errors

LONG-TERM

change some aspects from recursive calls to loops

Advantages:

- lowers recursive calls
- more suitable for upscaling to larger boards
- more professional, less of a 'hack'

Disadvantages:

- slow to implement
- loops are not friendly in functional languages – may require additional language integration
- may not prevent stack errors

Table 8. Comparison of short and long term solutions to the stack memory issue

Considering the options to address the stack memory issue within the algorithm, the short-term solution of rotating the board and solving for each side was selected. This was selected mainly due to the limited time scale in which it had to be implemented.

These planned changes were included in an alteration to the project's Gantt chart provided below.

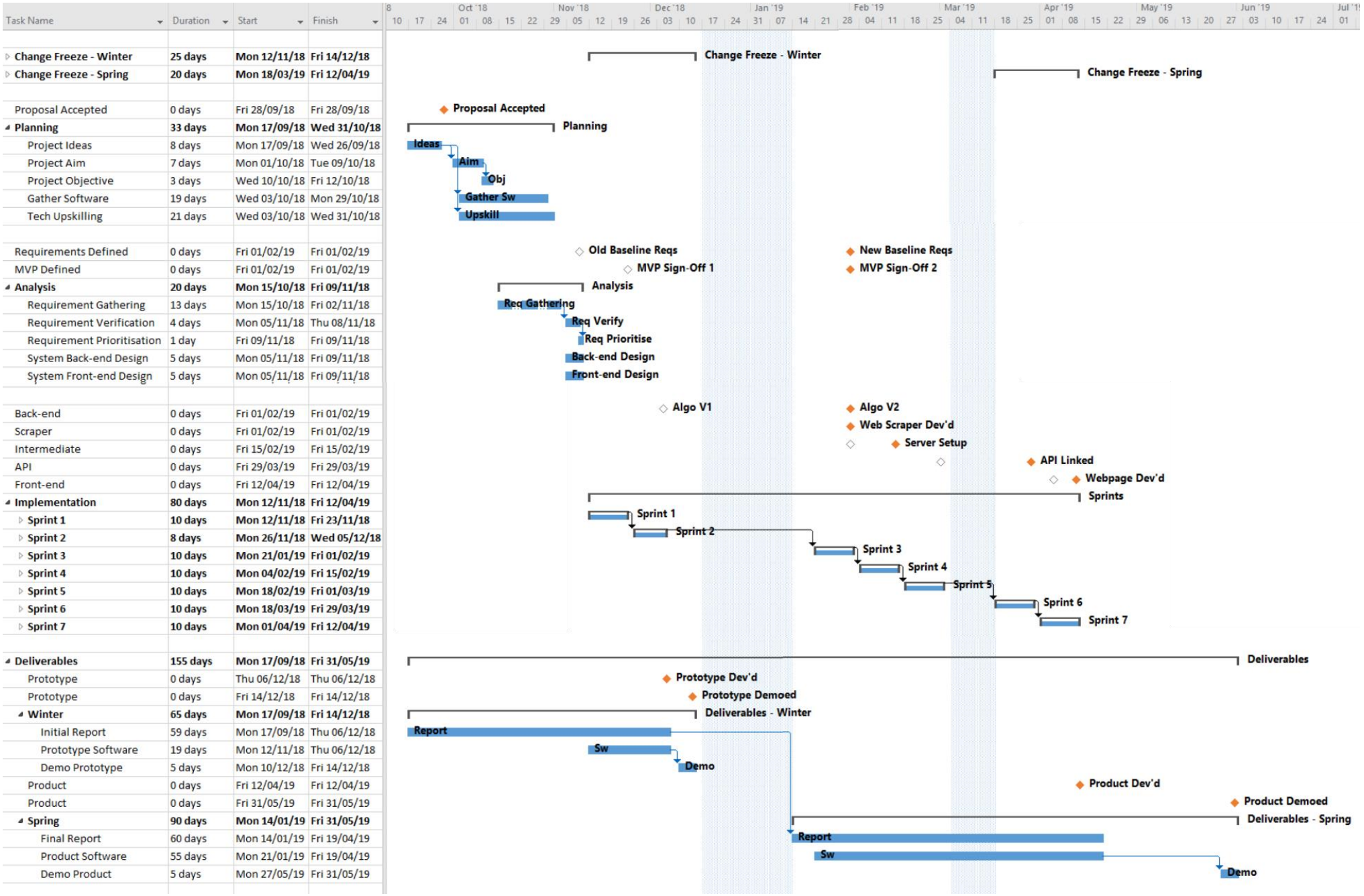


Figure 11. Revised Gantt chart

7.0 System Design

The approach to designing this system revolved around the idea of integrated system modules.
These modules took an extremely high-level view of the system.

The developer began by creating modules for the front-end and back-end.
From this setup it was clear that an intermediate was needed to communicate and transfer data between the two.

This approach was repeated until the high-level concept of the system, depicted in section 7.1 was created.

7.1. System Architecture

Figure 11 below shows the application's architecture.
The diagram shows the structure of its internal and external processes and the interactions between them across the three layers: presentation – providing a perceptive display from the underlying code, logic – the computational services and data – the underlying storage and management of process data.

The diagram took into consideration design principals outlined by Balosin (2017).

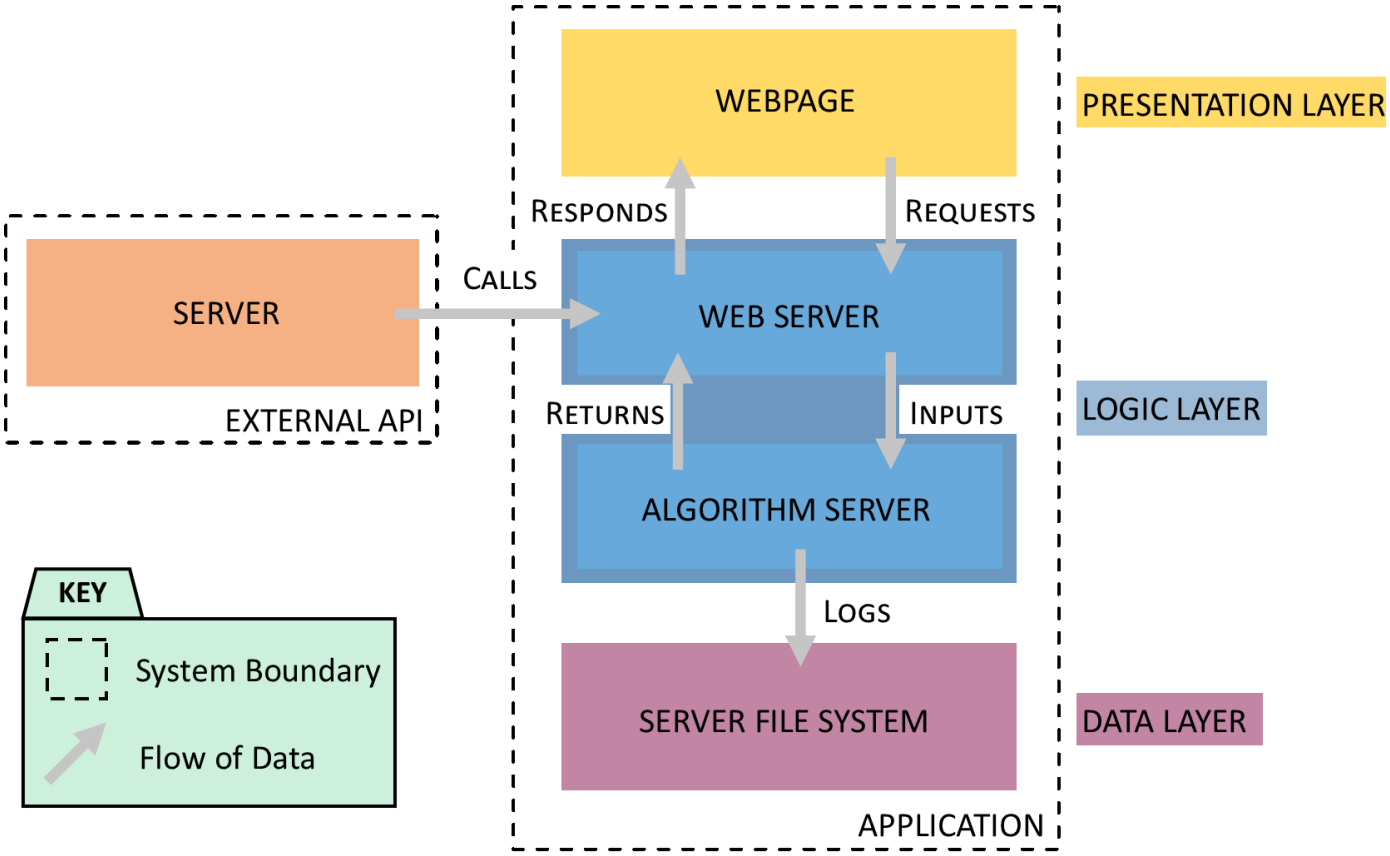


Figure 12. System's architecture

The presentation of this system involves interactions with the end users through the use of a general personal computer screen.
An interaction is made to the web server when the user makes menu selections or a request to solve the on-screen board through a button click. These interactions are handled by the web server. Interactions such as mouse and keyboard entry or validation are handled on the client-side.

The logic layer has two main components: the web server and the algorithm server.

The web server receives additional interactions, including those from the presentation layer defined above, from the algorithm server and the external server.

The internal algorithm server transfers a list of numbers representing a Sudoku board to the web server via an internal API call. If a solution from the algorithm was found, then these numbers represent the solved board, otherwise they represent the original board which cannot be solved.

This is the result of the interaction started by clicking the, 'solve' button.

The web server also requests a uni-directional call to an external server for a generated Sudoku board. This interaction is started once the user selects to generate a Sudoku board of a chosen difficulty. The external application provides a list of numbers to the server indicating a pre-generated Sudoku board of the difficulty specified by the user.

The data layer consists of the server which the algorithm server runs off.

The algorithm engine, housed in the algorithm server, provides a saved down log file of the algorithm's working out for solving a board.

This is initiated when the algorithm server calls for the engine to solve a board.

7.2. Interface Design

Wireframe prototype interface screens (low fidelity) were developed using Microsoft Publisher.

This application was chosen because of its simplicity and familiarity to the developer which allowed for wireframe diagrams to be drawn up quickly. This was an advantage over using such applications such as Balsamiq or Cacoo which the developer had no experience using and as such, time would be wasted signing up accounts and learning how to operate the software. These low fidelity images were basic and as such Publisher's built-in shapes feature would suffice adequately in displaying neat prototype dashboards as opposed to using bespoke software with a small to moderate learning curve.

The wireframes depict the planned creation of 2 webpages: the home page and Sudoku page and an options dialogue box to facilitate choice to the user.

7.2.1. Home Page

Figure 13-1 below shows the home page for the application, including the basic layout of the page's elements as well as some brief explanations on each.

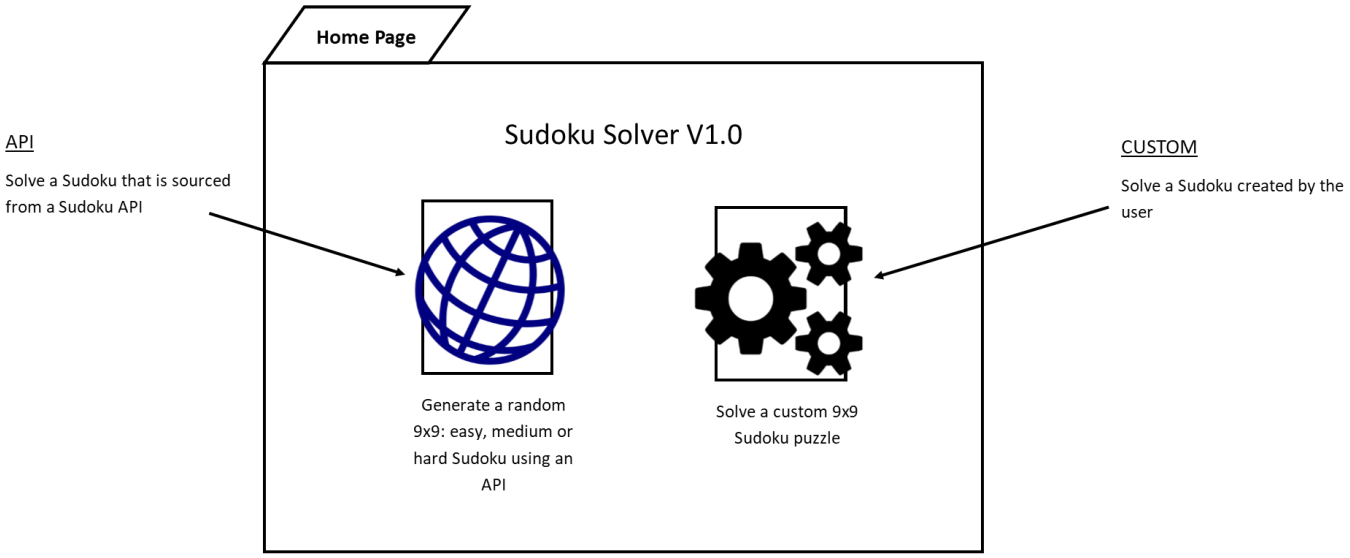


Figure 13-1. Wireframe 'home' page design

7.2.2. Options Dialogue – API

This is the options dialogue box that will be displayed if the user selects an API sourced puzzle from the home page before.

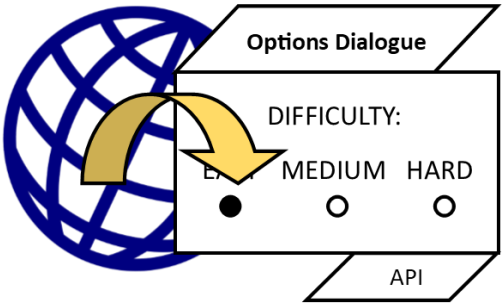


Figure 13-2. Wireframe API 'options' dialogue design

7.2.3. Sudoku Page

The webpage in which the user can play Sudoku and solve puzzles is shown below in figure 13-3.

The page has a linear and centrally aligned setup which was deliberately chosen as the application, in future renditions, would look to be ported to mobile.

This has been suggested during the development of this desktop version by stakeholders and the project mentor.

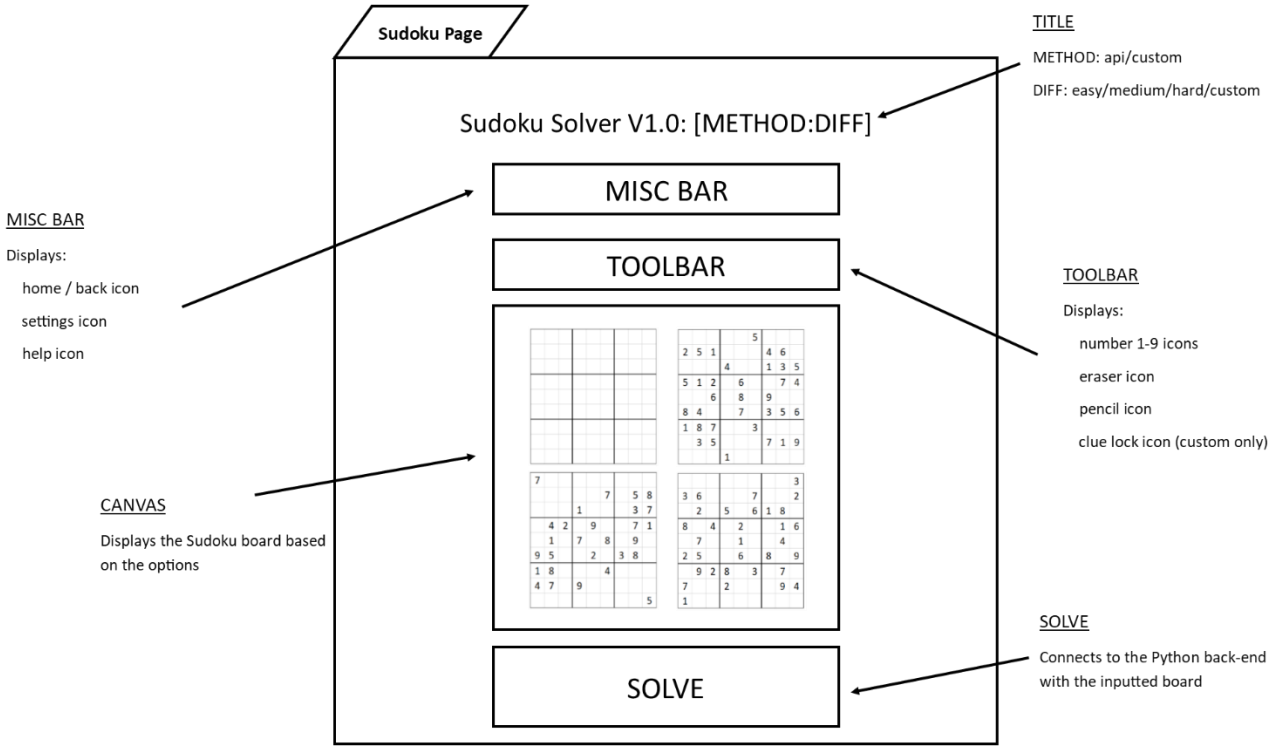


Figure 13-3. Sudoku page design

7.2.4. Human-Computer Interaction (HCI) Consideration

The use of colour and its shades can indicate actions or events as well as prompting subconscious emotional responses in humans. For example, the colour red often depicts scenes of anger and danger.

This phenomenon must be considered when deciding the colour-scheme of the product as not to cause unease.

The contrast of colours must also be considered as prolonged exposure can cause eye strain and headaches which will hinder the experience of the product.

Approximately 3 million British people can be identified as being colour-blinded to some degree (Colour Blind Aware, 2019).

There are different types of colour-blindness which exhibit different visual impairment. In order to make the product suitable for everyone, its front-end designs will be ran loaded into a Colbis (2019) colour-blindness simulator for distinguishability.

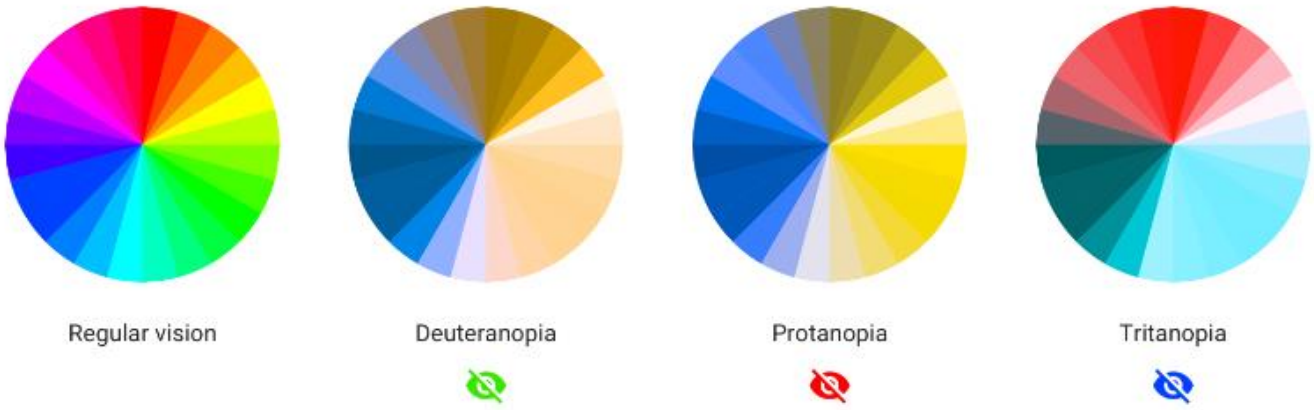


Figure 14. Different types of colour-blindness (Tuchkov, 2018)

The use of symbols to represent actions or events is used globally from maps to road signs because of its ability to traverses language and culture differences.

Within computer programs these symbols are often represented as icons – a small and very often simple picture, composing merely of a few edges and filled blocks – which inform the user in a very friendly manner that something is happening, or something will happen upon pressing it.

These computational complexities are hidden to the user as a small symbol which are transferred from program to program, e.g. a floppy disk denotes a saving mechanism.

Symbols are also much quicker to interpret which aids in developing familiarity and self-conscious understanding.

This project will use symbols to denote actions.

7.3. Data Support Design

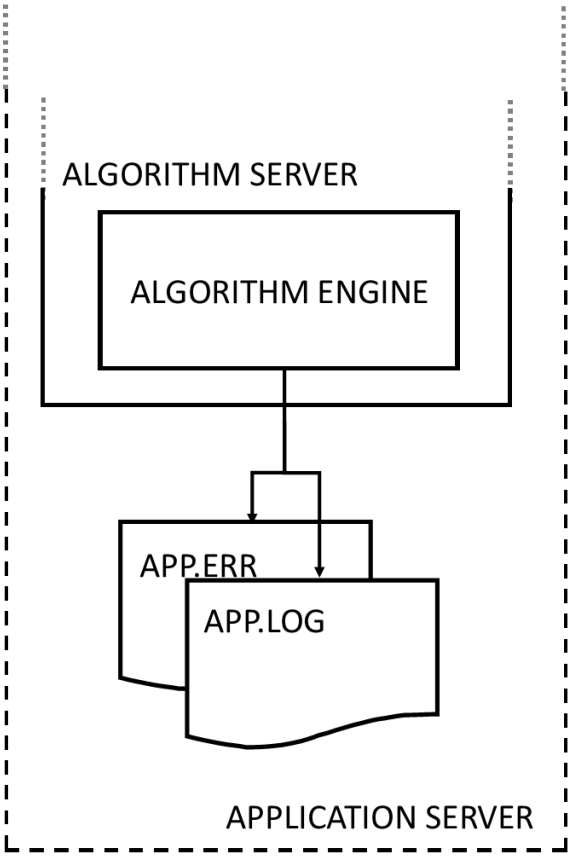


Figure 15. Application's algorithm engine's logging setup

Figure 15 shows the logging structure for the product.

The algorithm engine logs to the Linux server in which it is ran. It writes out to two separate files: app.err and app.log. The former of which writes standard error (2) from the q interpreter whilst the latter writes out standard output (1).

This is configured as such, through the following code:

```
\1 /path/to/log/directory/app.log
\2 /path/to/log/directory/app.err
```

However, not all standard output is written to this file.

The q logging library used (log4q) allows for messages of different severity to be distinguished. Debug, info and warn are examples of different severities within the library. Message of any severity are only written if the q server is configured to do so.

The below code shows the setup for this program's logging:

```
-log info
```

This specifies that only messages that are labelled as severity, 'log' or higher are recorded, i.e. 'log' and 'warn' are written to file whist 'debug' is not.

The system will make an API call to another server to extract a generated Sudoku board of a specified difficulty.

Figure 16 below, shows the transfer of the board as a series of numbers in which the web application will parse and validate before displaying it to the user.

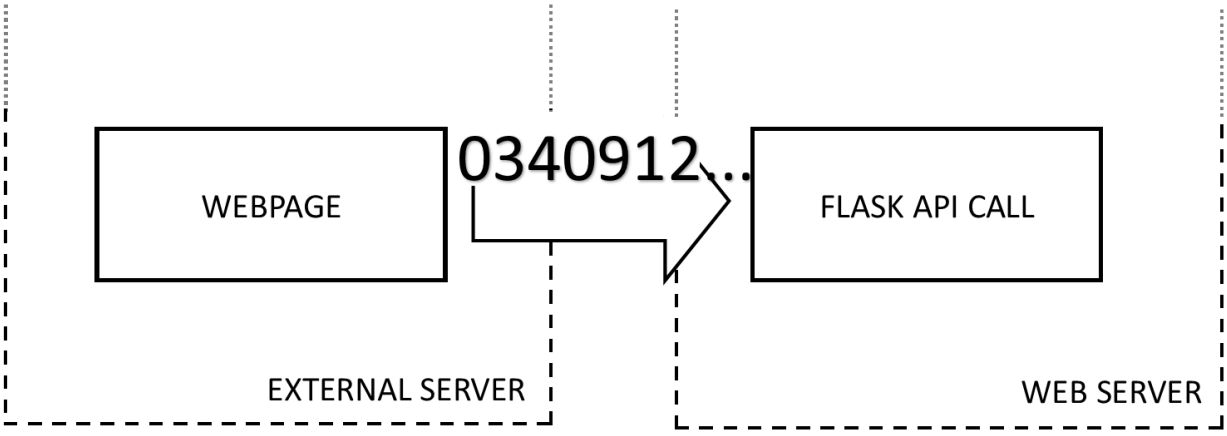


Figure 16. Transfer of a generated board from the external server to the web application

7.3.1. Data Validation and Security

The application will have limited sections in which the user makes selections or enters data.

The first data entry point regards the selection of Sudoku boards on the home page. The user's input here is limited to 3 by the use of radio buttons, which enforce that only one option can be chosen, removing any parsing errors.

Once a board is selected the user can now enter values into cells.

These inputs will be validated through the use of regex, allowing only the valid Sudoku values to be entered.

The boards which are passed to and from the web application must also be validated.

The front-end displays a board passed to it from an external API when the user selects for a generated board to be used.

The transferred board's values must be validated through parsing and errors must be handled in the case of difficulties connecting to the API. The board passed to the solver should be validated with these settings in-place and therefore should not require additional checking.

The clues fetched from a sourced board are regulated. These cells will be made disable to prevent any alteration to its contents.

7.4. User Interaction Model

7.4.1. Use Case Diagram

The use case depicted below in figure 17 highlights the process interactions within the scope of the system and the external entities starting it.

There are 2 entities acting upon the system in very different ways.
The user is a primary actor and initiates the internal processes of the program by operating with the user interface and making on-screen selections.
The external API interacting with the system is a secondary actor and provides data into the scope of the system when it is requested and, as such, is a reactive actor.

There are primarily, 2 sets of activities that can be initiated by the user.
The cluster of processes towards the top of the system scope revolve around the user generating a board from the onscreen interface.
This general process has 2 more specific child processes: generating a blank board (for custom boards) and fetching a board from the external API; the latter having an additional select difficulty process which is always triggered, hence the include relationship.
These subprocesses inherit the data structure of a puzzle board from the parent process.

The user, once a board is generated, can utilise the solving aspect of the program which is highlighted in the lower half of the program's scope.
Here the complex and internal mechanism can, on occasion, display one of two alert messages depending if the board supplied does not have a solution or a solution cannot be established.

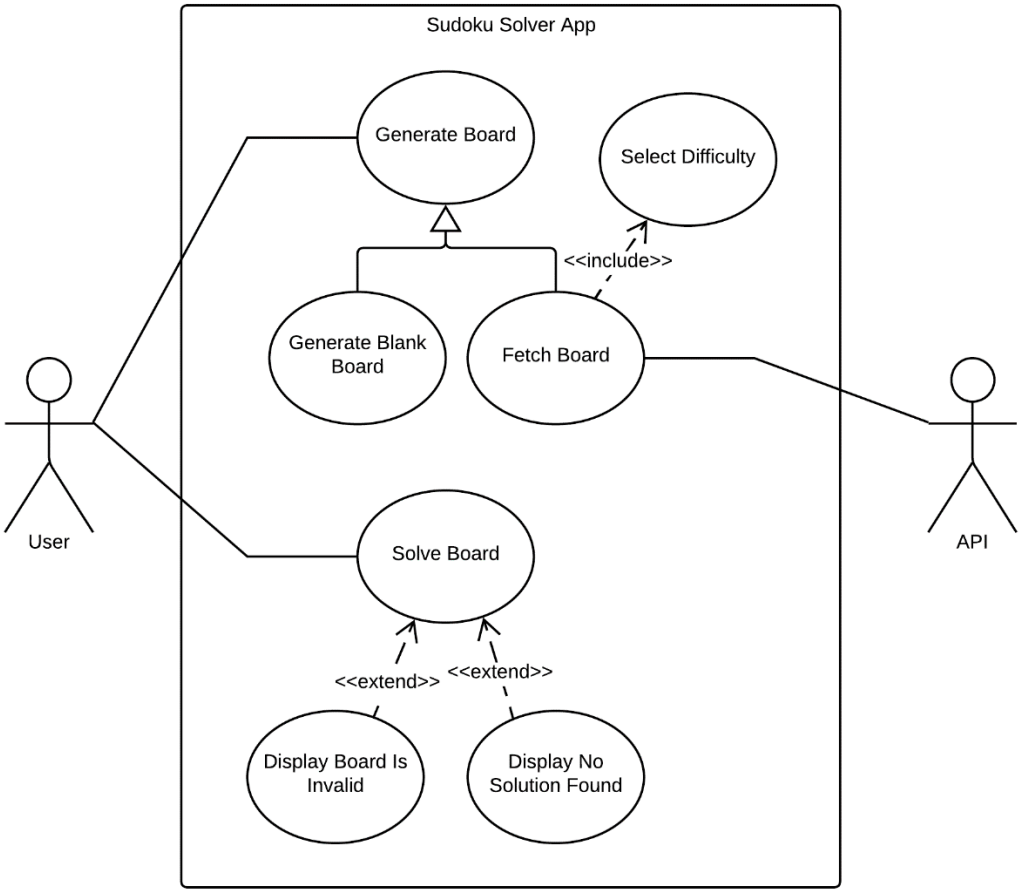


Figure 17. Use case diagram

7.4.2. Activity Flow Diagram

Figure 18 showcases the flow of data and order of activities for generating and displaying a Sudoku board.

This activity is initiated by the user upon selecting a board source; the structure of the board's values at this point is inherently empty as depicted by the pending board values.

Once the source of board has been selected the activity/activities performed immediately after differ, as described by the decision-branch node in the diagram.

If the user selected a board sourced from the API, then the diagram follows the left path from the node (as per the, 'api' guard condition).
Upon following this path, the process of choosing a difficulty is triggered and a prompt is provided to the user.
Once a difficulty is selected the flow continues downward and converges to a decision-merge node along with the path defined by the, 'custom' guard condition (which did not take any further action).

The board's values can now be added to the pre-existing storage structure: the API supplies its generated values whilst for custom boards the values are left blank.

Finally, the board is displayed and the activity ends.

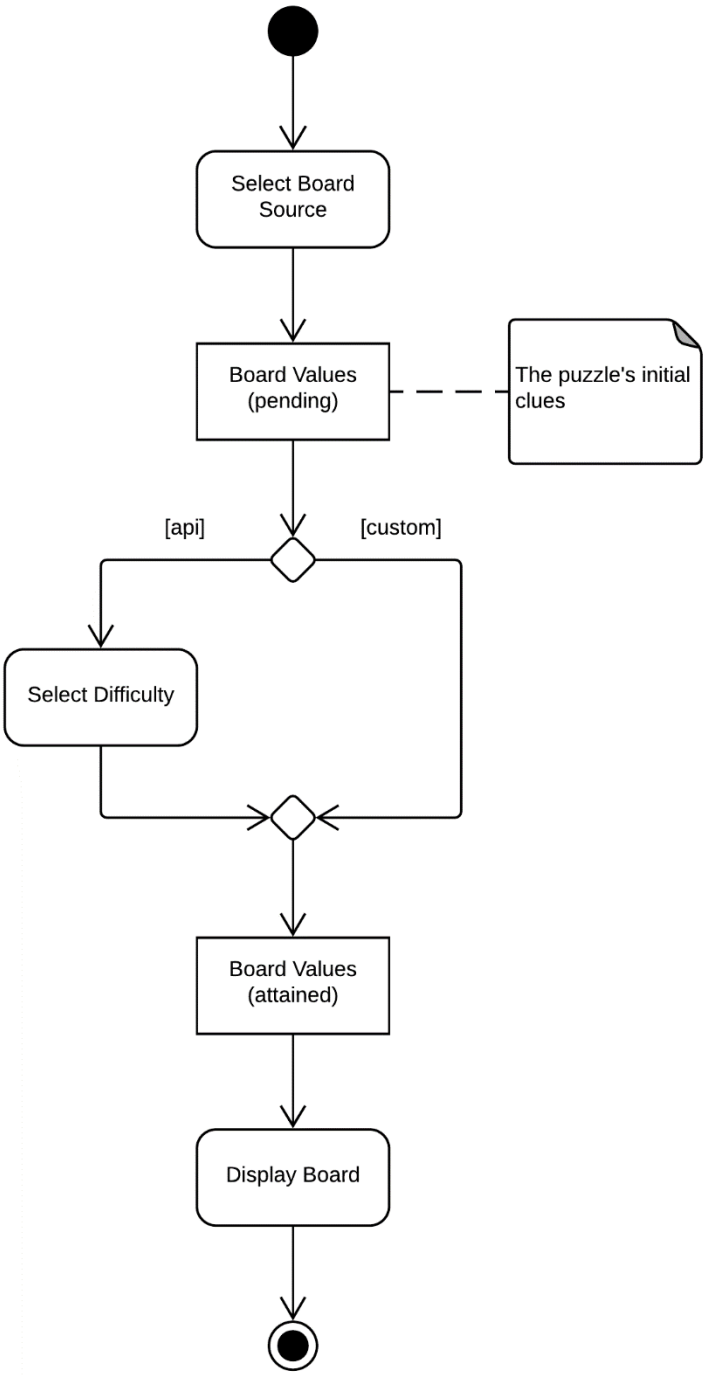


Figure 18. Generating and displaying a Sudoku board activity diagram

7.5. Additional Design Artefacts

The solving algorithm is predominately made up of 5 code blocks with some of these been triggered by meeting certain condition(s) within their parent function:

1. init
2. try_again
3. try_next
4. bt
5. bt_row

1. init

Figure 19-1. init function's code block

This is the init code block which is responsible for deciding whether the candidate value (the value in contention to be assigned to the cell) is to be assigned to the null cell in question.

The first decision within this block is whether the candidate value is invalid, i.e. not null.

If it is, this indicates that the cell is out of viable options for values and so, the algorithm backtracks (bt).

Next, the function temporarily assigns the first missing value (those missing from the row) which is greater than the current cell's value.

Once a temporary value is assigned to the cell, it must be checked for conflicts within the current state of the board (check row, column and grid).

If it conflicts, then the algorithm attempts to try the next value (try_again) or examines the next null value (try_next).

2. try_again

This is the try_again code block which is called from init when the candidate value conflicted with the current board setup and is responsible for deciding whether it was the last case of assigning that value to a cell in the row in question or not.

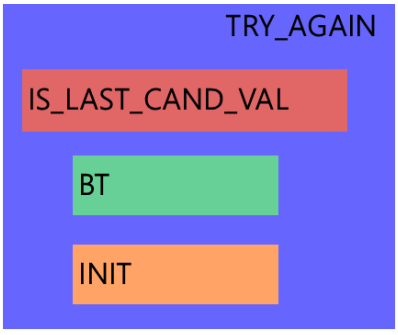


Figure 19-2. try_again function's code block

This function tests whether the value attempted to be assigned to the cell which has caused conflict is the last potential value to be entered, i.e. the highest which is missing from the row.

If this is true, then the cell is exhausted of potential values and deems that its true value is held by a cell before it in the row and so the algorithm calls backtrack (bt).
However, there are other values to try for that cell, then it simply calls init.

3. try_next

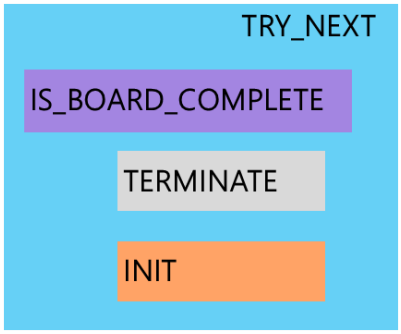


Figure 19-3. try_next function's code block

This is the try_next code block which is called in init when the candidate value did not conflict with the current board setup and simply checks whether that was the last null cell assignment and therefore, the puzzle is solved.

This function tests whether the board in its current state is complete, i.e. there are no more rows with null values.

If this is true, then the function terminates deeming the board solved and displays its solved state or it calls init.

4. bt

This is the bt code block and is responsible for deciding whether the algorithm needs to back-up to the previous row as well as checking whether the board is invalid.

The first consideration with this function is whether the board is invalid.

If it is, then this will result in the given board being deemed unsolvable and the program terminate, displaying the board.

Next, the algorithm considers whether the current row cannot be assigned any values because of conflicts, i.e. none of the missing values for the row can be entered into the first null cell of that row, suggesting an erroneous value in a previous row.

If this is the true, calls for backtracking to begin on the previous row (bt_row) otherwise, the init function is called.

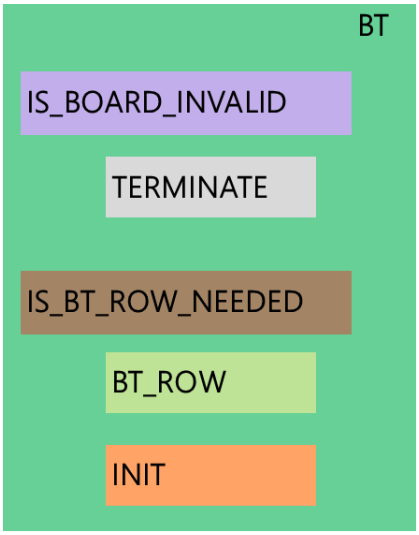


Figure19-4. bt function's code block

5. bt_row

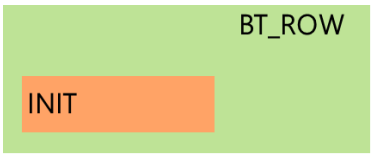


Figure 19-5. bt_row function's code block

This is the bt_row code block which is called by bt when the first null cell of a row cannot take any missing values without conflicts and is responsible for adjusting the values to be entered into the init function before calling it.

This function simply reassigns the row and cell to be examined to the previous row and calls init.

The recursive interactions between these code blocks are depicted below in figure 20.

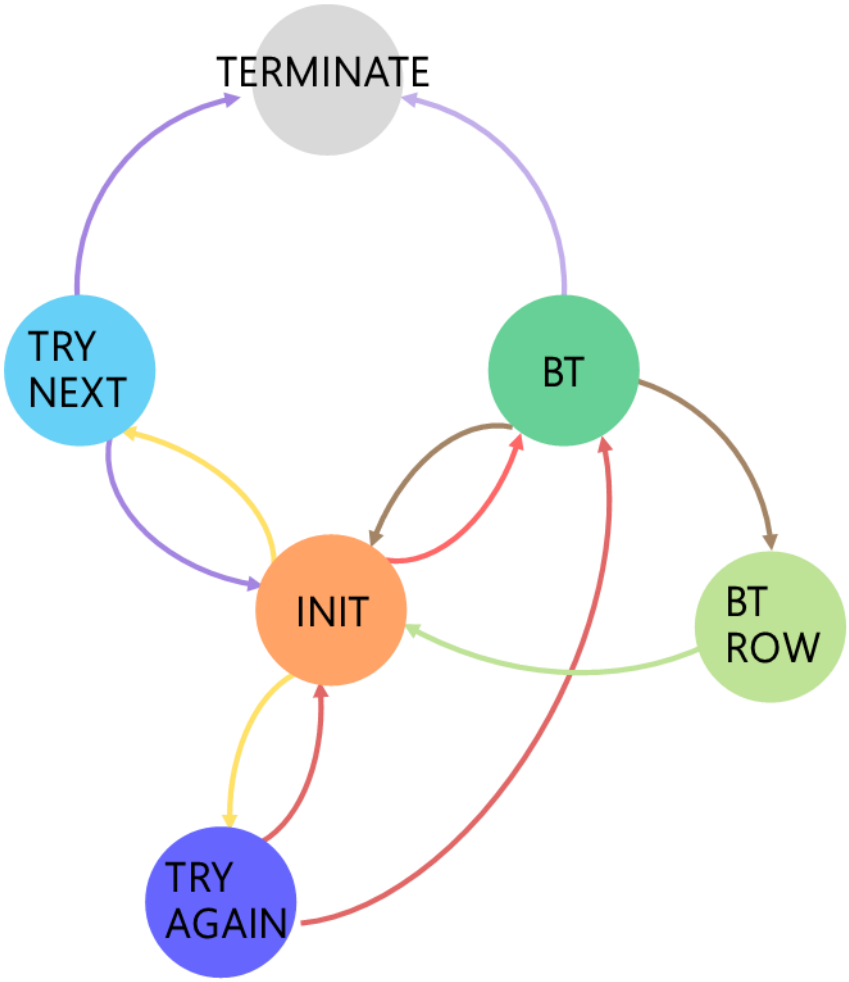


Figure 20. Recursive calls between code blocks

The solver utilises recursion and exhibits multiple recursive loops within its structure.

The solver's recursive nature can be categorised depending on the direction of 'movement' in which values are assigned.

The loops init-try_next-init and init-try_again-init can be said to move 'forward' on the rows (left-to-right) as the program assigns values to the current cell or ones to the right of it.

However, the direct loop init-bt-init and the more complex loops which are extended by the meeting of validation checks: init-bt-bt_row-init, init-try_again-bt-init and init-try_again-bt-bt_row-init move backwards, hence backtrack.

The alternative functions to init can be simplified to wrapper functions which adjust the values being inputted to init with the addition of error and validation checks.

8.0 System Implementation

8.1. Reflection against Implementation Plan

The plan of using the agile framework to manage and complete the implementation phase of this project was carried out satisfactorily in accordance with the agile manifesto.

The development and delivery of key software features as depicted in the original project plan was extremely accurate once incorporated with the revisited version.

The use of the agile methodology did result in the project being split up and developed in small and concentrated software areas of interest.

The sprints of, on average, 2 weeks did involve all of agile phases: task section, development and testing, however, the involve with the stakeholders throughout, was less than desirable; meeting up, after most, after 3 sprints due to other commitments held by the developer.

This mean that feedback was few and far between which could result in an underwhelming product.

A real-life Kanban board was used throughout to manage the process (see Appendix D).

The original project plan highlighted key features to be developed against each sprint, however there were delays and modifications to these.

During the Winter demonstration of the algorithm a stack memory issue in the algorithm and the uncertainty revolving around the evaluation of the solve, resulted in additional time and resources being devoted to the algorithm.

This meant that the adjacent sprint was now concerned with resolving these issues.

A web scraper was also to be developed in-order to evaluate the algorithm which was not originally planned.

This resulted in the Python library BeautifulSoup being drafted into the program.

These alterations to the original plan had a ripple effect throughout the rest of the project and its key features which was indicated by the revisited Gantt chart (see section 6.2).

8.2. Software Usage

8.2.1. Tools

To keep track of issues being worked on and future ones, the developer created a custom Excel worksheet (see Appendix F). The fact that this worksheet could be altered without internet connection meant that it was more flexible than bespoke issue tracking systems. Its simplicity to use and familiarity meant that there was minimal learning to be done.

Tasks were managed using a real Kanban board (see Appendix D). This board had presence which made viewing progress quick and easy. It also allowed for the developer to identify bottlenecks within sprints. The board was placed at the end of the developer's development space which allowed them to destress and, on occasion, undercover a more efficient way of working on the task.

8.2.2. Languages

The back-end algorithmic server, which housed the algorithmic engine was written in q; a functional programming language developed by KX. This language was selected for the algorithm because of its functional paradigm which makes extensive use of recursion which would be fully utilised and suitable for the advanced brute-force solving mechanism. Using a functional programming language and adhering to a fundamental principle of focused functions allowing code to be read easily and its specific functionality understood. The usage of higher-order functions (allowing functions to be passed into others and even returned) allows for extremely powerful functions to be developed in a few lines. Q's single threaded nature along with the idea of non-mutation with pure functions ensures that its functions are deterministic, guaranteeing repeatable and consistent results.

The front-end webpages are written in typical web-friendly languages: HTML, CSS and Javascript. These languages are often used in conjunction and complement each other excellently. Javascript will allow the page to be dynamic and allow for media queries, such as scaling, to be implemented.

The intermediate between the front-end and back-end will be handled by Python and the Flask framework within. Python was selected because of its abundance of modules and frameworks such as Flask, but it was also used in developing the web scraper. Python is also a well-engineered 4th generation programming language which is easy to read and understand.

8.2.3. Frameworks and Libraries

The Flask framework within Python was used as an intermediate server for the project. It was selected because of its robustness as a standalone server and fantastic community support and documentation, allowing for issues to be resolved quickly.

The plugin pyq, developed by KX, was used within the intermediate server setup to allow the Flask server to communicate with the q written algorithm engine.

The algorithm logs to disk and is formatted using the log4q module. Logging is exceptionally important within a q application as it allows developers who are maintaining the program to understand why and where errors are occurring in order to resolve them. Log4q provides a customisable log message to be outputted to different files depending on the severity. It also provides multiple levels of severity which can be customised. It also has a feature to email out messages.

The Python module BeautifulSoup was used to extract data from XML and HTML layouts. Within this project it was used to create the web-scraper for boards by filtering through HTML elements and extracting the embedded puzzle.

8.2.4. APIs

Q has built-in server hosting abilities which are used to pass solved boards to the Python intermediate through an internal API call.

The Flask server makes calls to the q server to solve a board and returns the solution.
The use of the pyq plugin allows for this connection to be established.

To fetch a puzzle of a chosen difficulty the Flask server makes calls to an external server.

This call scrapes a board from the external server and returns it to the Python server for analysing.

8.3. Version Control

Automated version control was used throughout this project in the form of Git and hosted on Github.

Git allows files to exist in 3 states: remote, staged and local.
Remote contains files that are pushed to a branch and recognised on Github whilst staged and local are contained on the local machine.
Staged files were those understood by Git as those to be committed to the branch whilst local were changes or files that were not known by Git.
This setup allowed the user to pull finished code onto a different machine and work off it.

This project made use of one repository which housed all the code.

Within the repo, the developer made use of 3 branches: master, feature and test.
The master branch contained code which was fully developed and committed to it through the use of pull requests.
However, feature and test branches were based off the issues added to the Excel worksheet (see appendix E) which perfectly resembled the sprint backlog tasks present on the Kanban board (see appendix D).
Feature branches were used to develop code to appease the issue's functionality whilst test branches were used to test the corresponding code fragments.
This setup allowed for extensive code traceability with its organisation of issues and code.

Pull request to the master branch were configured so that the request had to be reviewed and approved before being merged.

This forced the developer to look over their code once more to discover any minor bugs such as syntax issues or spelling mistakes to be rectified so they do not appear on the master branch.

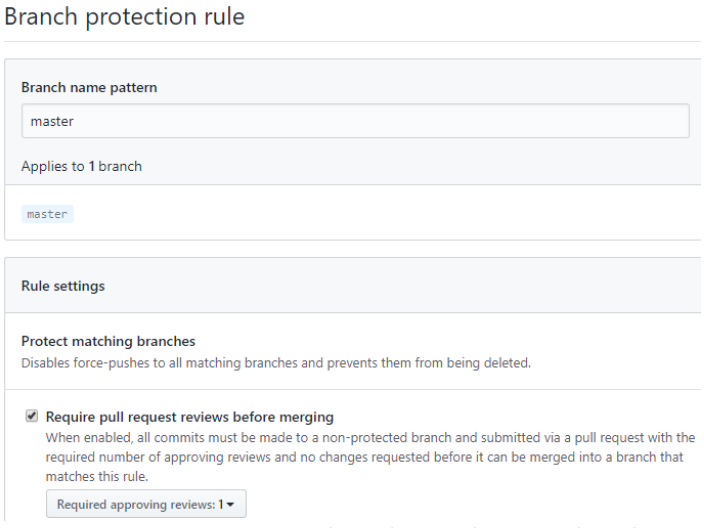


Figure 21. Merge request rule in-place on the master branch

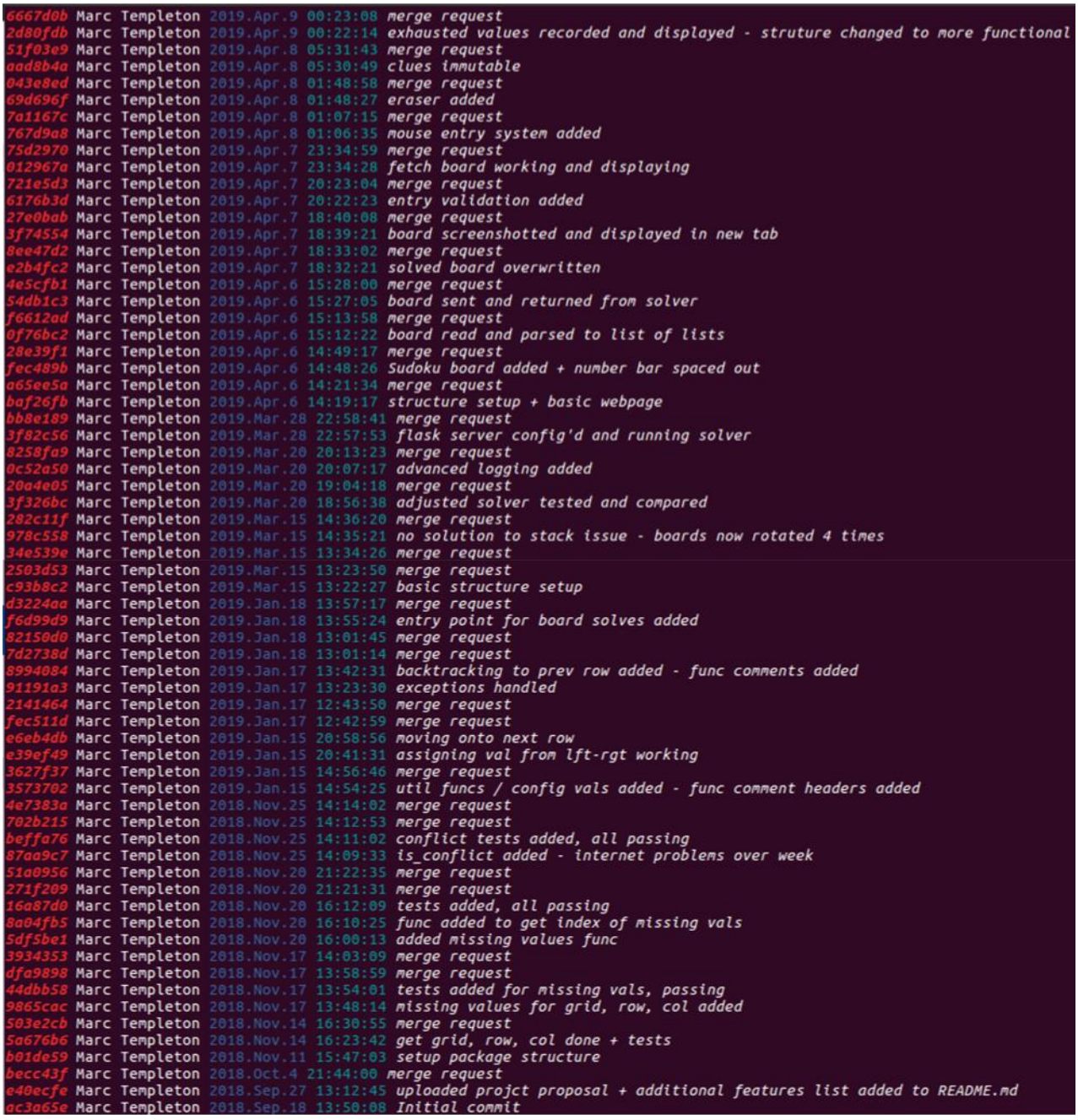


Figure 22. Sample screenshot of the git commits during the project

8.4. Code Volume

Table 9 below summaries the code developed for the project. The files were auto-formatted by Atom's 'Beautify' package and the lines were counted using Linux's 'wc' command.

	#Linux lines	#HTML lines	#CSS lines	#Funcs	Total Lines
WEBSITE					
index.html	-	54	2	-	56
sudoku.html	-	186	3	2	191
CLIENT-SIDE SCRIPTS					
style.css	-	-	189	-	189
sudoku.js	-	-	-	2	38
utils.js	-	-	-	30	290
FLASK SERVER					
src.py	-	-	-	4	28
ALGORITHMIC ENGINE					
init.q	-	-	-	-	4
src.p	-	-	-	1	11
src.q	-	-	-	30	595
test.q	-	-	-	65	237
PROCESS LAUNCH					
[server] launch.sh	1	-	-	-	1
[engine] launch.sh	9	-	-	-	9
scrape.sh	5	-	-	-	5
MISCELLANEOUS					
scrape.py	-	-	-	3	47
extract.py	-	-	-	1	20
TOTAL	15	240	194	138	1,721

Table 9. Code produced over the course of the project

8.5. System Walkthrough

When the user hits the application, they are presented with the 'home' page as figure 23 shows below. From here they can navigate to the rest of the application.

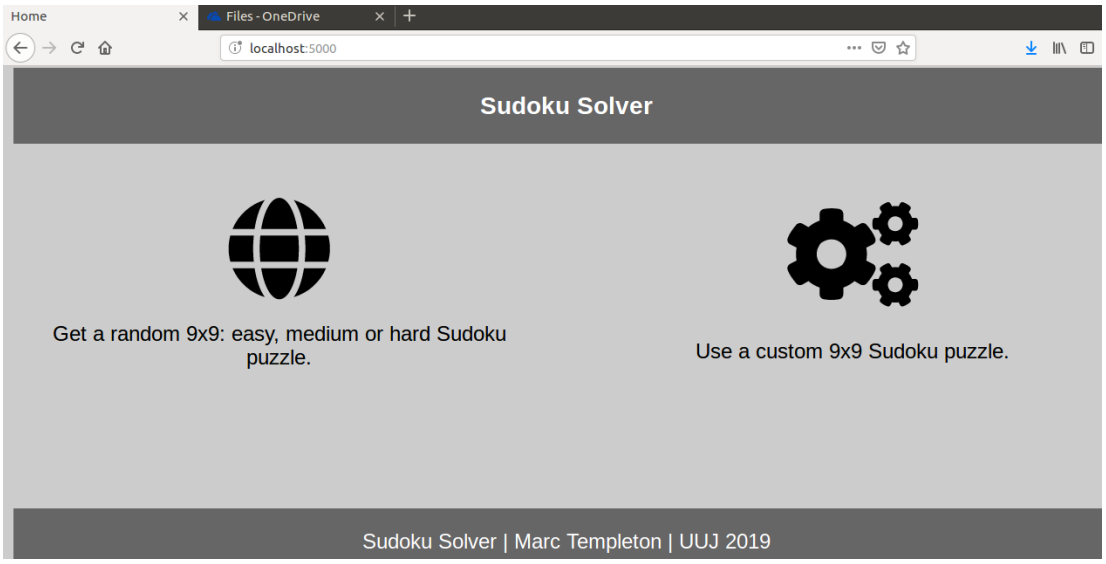


Figure 23. Home page

There are 2 clearly distinguishable board selections for them to choose: custom or generated.

On clicking generated a dialogue box is displayed (figure 24) to the user for them to select the difficulty of the puzzle to be fetched.

On selecting 'Easy' difficulty and clicking fetch, the application changes to its Sudoku page before fetching an 'Easy' board.

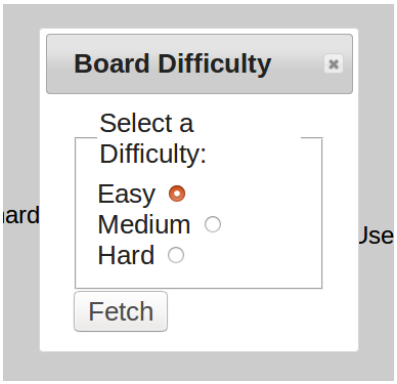
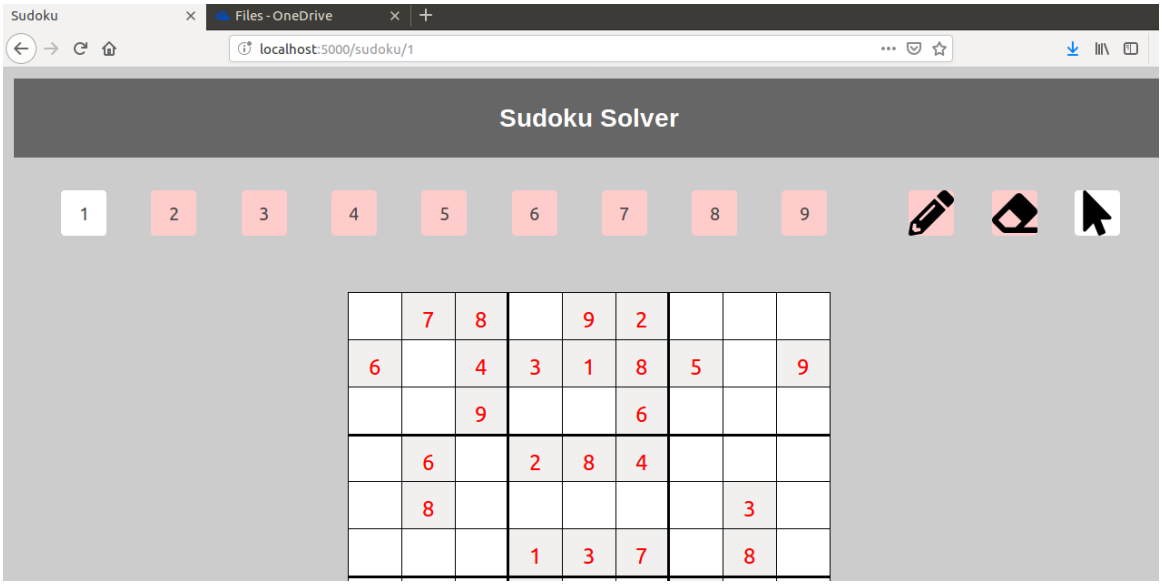


Figure 24. Fecthed board difficulty selection



	7	8		9	2			
6		4	3	1	8	5		9
		9			6			
	6		2	8	4			
	8						3	
			1	3	7		8	

Figure 25. Sudoku page with fetch 'Easy' board

The page's URL has changed to, "/sudoku/1".

For custom board this is, “/sudoku/0” and for medium and hard puzzles it is, “/sudoku/[2|3]” respectively.
The decision for the application to fetch a board is decided upon after it has been loaded.

Once all of the page’s HTML components have been loaded onto the page then the application checks to see if a board is needing to be fetched.
It does this by checking the URL of the page.

If the path of the URL after, “/sudoku/” is not 0,then it proceeds to fetch a board by calling the angular function fetchPuzzle.

It passes this, non-zero value, into this function as the fetched puzzle’s difficulty, encoded as 1=easy, 2=medium and 3=hard.

```
Once the page has loaded
   If the digit after “/sudoku/” is not 0, then fetch
```

To fetch a puzzle the appication makes an API call which is handed by the Flask server, as figure 26 below shows.

Figure 26. Application’s fetch call handled

Handling this request envokes a function call to, “scrape_board” and returns one of its elements; the scraped board.

This board is returned and parsed, drawn onto the board and made immutable as partly seen in figure 25.

Now the user is able to enter values into the board.

The mode of entry is highlighted by being illuminated.

The default mode of entry is mouse entry, denoted by the illuminated pointer icon.

When in mouse entry the user selects the value to be entered by selecting the digit from the number bar. This number is highlighted as being chosen.

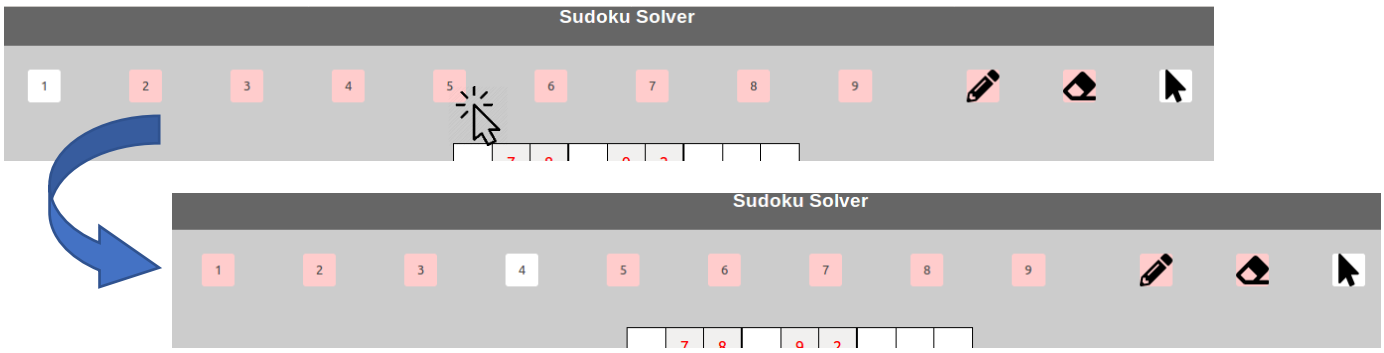


Figure 27. Number change for mouse entry mode

Now the user simply clicks on the cells in which they want that digit to appear in. It can also overwrite the value of a populated cell.

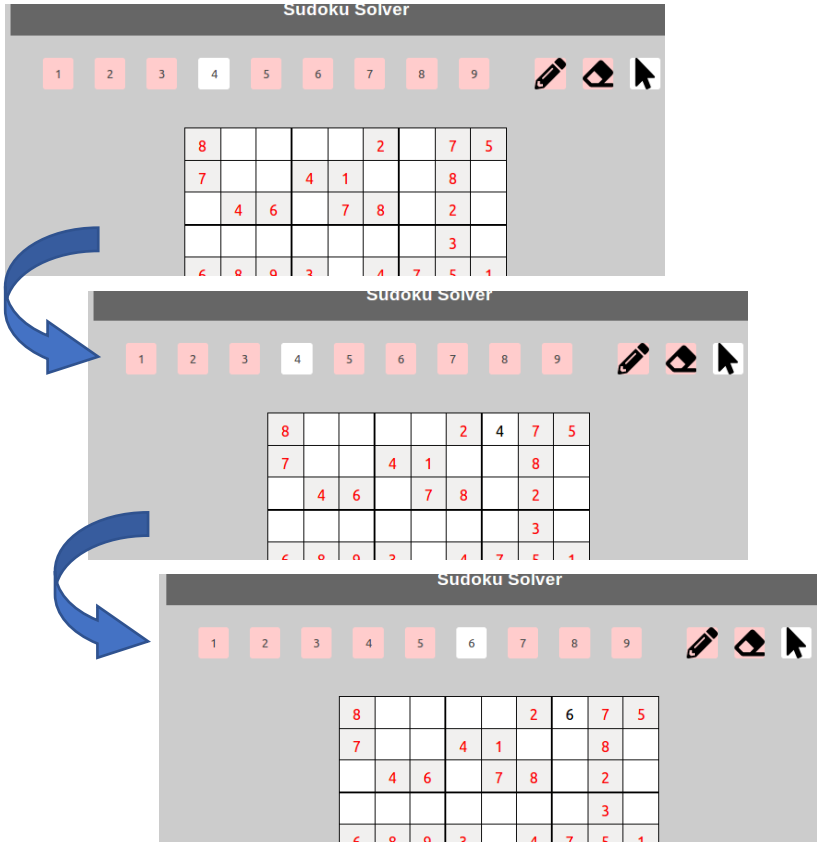


Figure 28. Mouse entry and overwrite

The user simply changes entry mode from mouse to keyboard by clicking the pointer icon.

To insert a value into a cell in keyboard mode the user simply selects a cell before typing in the digit to be inserted. Overwriting is done by first removing the value in the cell before entering another one.

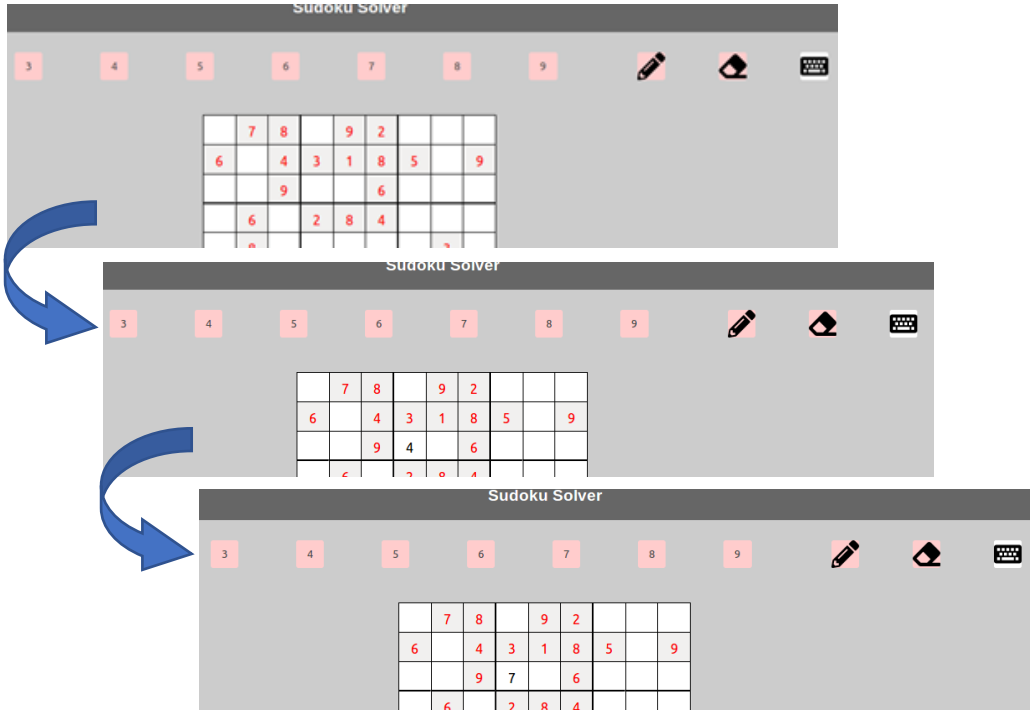


Figure 29. Keyboard entry and overwrite

If a value that is not found in the number bar, then the user is displayed with an error message and the cell's value is returned to nothing.

Figure 30. Keyboard entry data validation

The eraser mode works similar to the mouse entry; click the icon to select and then click the cell to remove its value. Empty cells remain empty.

Upon fetching a board, the number of instances of each value is recorded.

When the user enters values into the board these are added to this record. Figure 34 below shows this exhausted value dictionary and how it changes with entry.

Once a value has reached the threshold of 9 instances on the board then the value in the number bar is diminished.

This change in appearance can be reversed by removing one instance of the value.

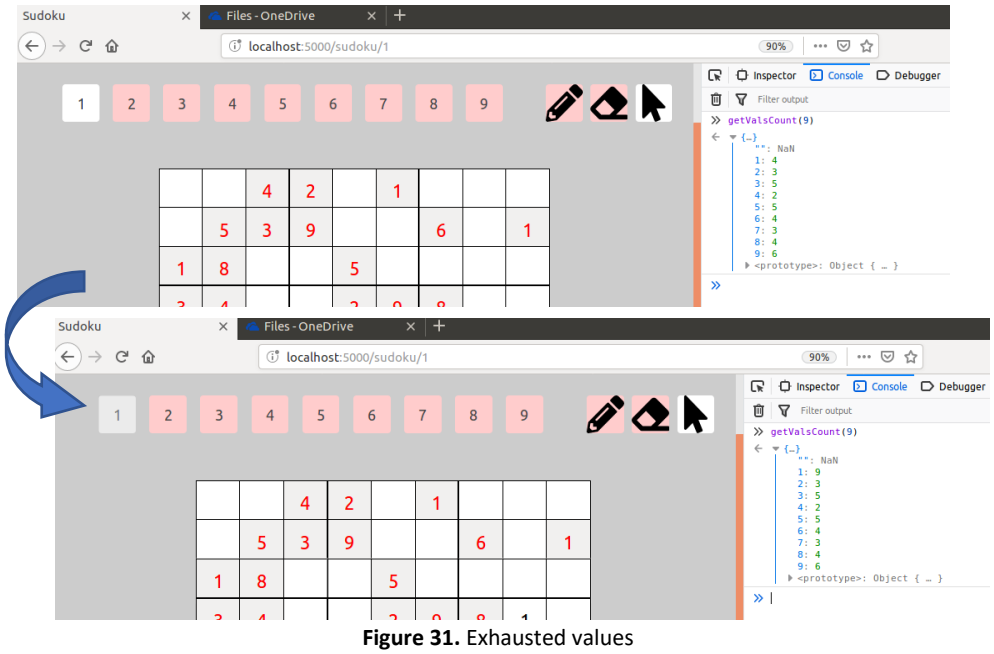


Figure 31. Exhausted values

The user can select to solve the puzzle on-screen by clicking the “solve” button.

From here the board is parsed and sent to the q server via an API call. The returned solved board is again parsed and overwritten onto the on-screen board.

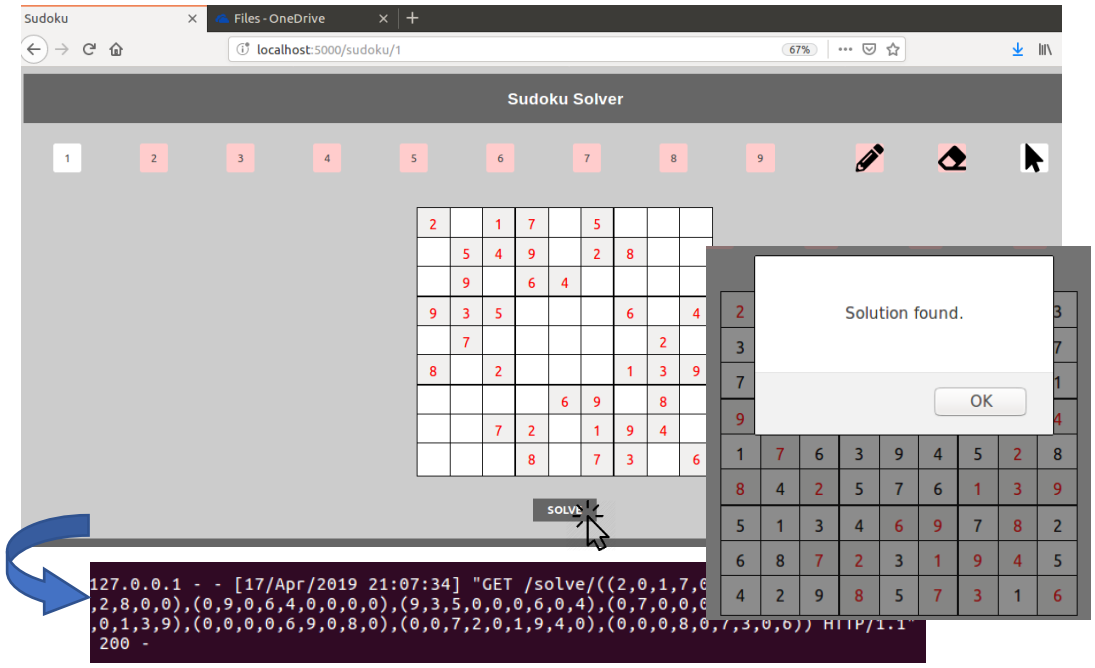


Figure 32. Solving the on-screen board

The solved board is also displayed within a new tab as shown below in figure 33

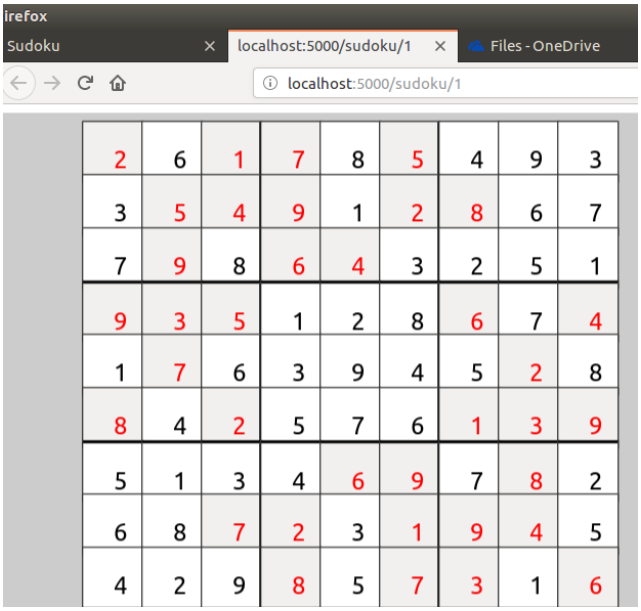


Figure 33. Screenshotted solved board

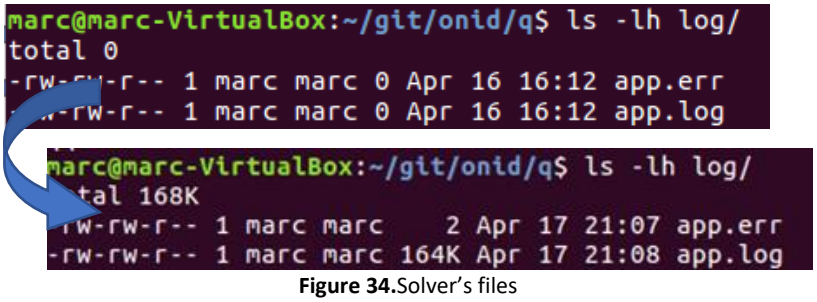


Figure 34. Solver’s files

Figure 34 shows the algorithm engine’s output to file from the solver.

Errors connecting to either: Python or q servers are handled and error message are displayed to the user.

8.6. Security

The project has been developed using globally used programming languages and frameworks.
This means that they are well tested and common issues have been resolved or are being so. The chances of errors being inserted through exploits within the de-facto software are extremely low and documentation allows for speedy development.

The Flask web server experiences function isolation for the solver as that is handled on the q server.
This lowers the number of functions executed on the webpage, reducing computational overheads and the potential for crashes under heavy strain.
The ongoing stack memory issue poses a problem, indicating further memory issues within this q setup. This could be exploited through memory exploits such as buffer overflow.
This has been an ongoing are of interest to resolve within the project, however its effects can only simply be moderated and controlled.

Q's server hosting abilities allow for users to access the server directly by hitting its server domain.
However, this address is not publicly given to any users and could only be discovered by experienced web users by analysing the network activity.
This exposure to the server was not identified as a risk and as such no user auditing or restrictions were applied. However, it would be worthwhile securing this connection up in further releases as experienced programmers can get access to the server as a superuser.

8.7. Solver Algorithm Evolution

The solver algorithm is the most important aspect of this project and as such has undergone several iterations.
An 80% solve rate on valid boards was set as per requirement R8 from section 6.1.2. To ensure this target was met the algorithm was tested against 1000 randomly sourced boards across a variety of difficulties. The solutions to these puzzles were known and therefore compared against the solver's output.

8.7.1 Algorithm v1.0

The first evolution of the algorithm was the application's prototype.
This algorithm had basic error and validation checking to ensure illegitimate boards were filtered out. The algorithm assigned values to cells and checked for their validity in a brute-force manner.

This algorithm was developed entirely in q over the course of sprints 1 and 2.

Table 10-1 below provides a high-level overview of the key functionality of the algorithm at separate milestones.

v0.1 • key functions • basic forward tracking • basic backtracking v0.2 • advanced forward tracking • advanced backtracking • validation checks v1.0 • recursive wrapper functions

Table 10-1. Algorithm V1.0 make-up

Figure 35-1 shows the performance of the mk1 algorithm in solving the 1000 randomly sourced puzzles.

This works out to be a 59.2% of the 1000 boards, meaning 408 boards weren't solved.

This falls well short of the required 80% solvability and so the algorithm had to be revisited.

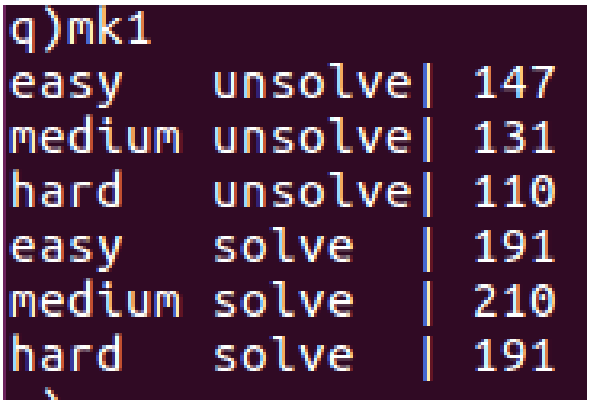


Figure 35-1. Mk1 algorithm's solving performance

8.7.2 Algorithm v2.0

During the testing phase of the prototype algorithm a stack memory issue was uncovered.

Therefore, an improved variant of the algorithm had to be created to adhere to requirement R8 of solving 80% of all boards.

This was developed in sprint 3; adjacent to the protype demo in which the issue was discovered.

To address this problem, the algorithm rotates the board and attempts to solve it on each of its sides.

This version of the algorithm also incorporated a wrapper for pyq to enable the Flask server to access it.

Table 10-2 below outlines the additional features that V2.0 included.

v1.1 • advanced logging • missed validation check v1.2 • recursive multiple attempt approach • try wrapper v2.0 • pyq wrapper

Table 10-2. Algorithm V2.0 make-up

Figure 35-2 shows the performance of the mk2 algorithm in solving the 1000 randomly sourced puzzles.

This works out to be a 94.5% of the 1000 boards, meaning just 55 boards weren't solved.

This is an improvement of 35.3% solving effectiveness, compared to V1.0.

This also exceeds the desired 80% effectiveness and so is accepted as the final algorithm variant.

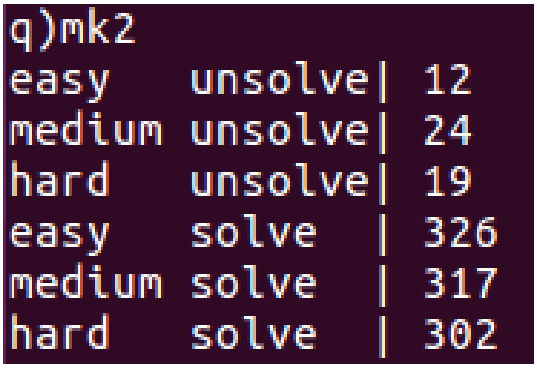


Figure 35-2. Mk2 algorithm's solving performance

8.8. Web Scraper

To test the performance of the solver algorithm and to fetch puzzles a web scraper was developed.

This web scraper was developed in Python using the BeautifulSoup library which is used to extract data from websites.
The following is a walkthrough of how it was feasibly identified, developed and tested for the purpose of this project.

Firstly, a suitable webpage had to be identified.

The developer was looking for a website that embedded its Sudoku board directly onto the page through HTML elements such as tables, sections, etc.

However, the service used, https://www.websudoku.com/ uses a proxy server.

The Sudoku board displayed on the page is a, 'frameset' element which is unsuitable for using BeautifulSoup (see figure 36-1). The site used was a proxy site, https://nine.websudoku.com/

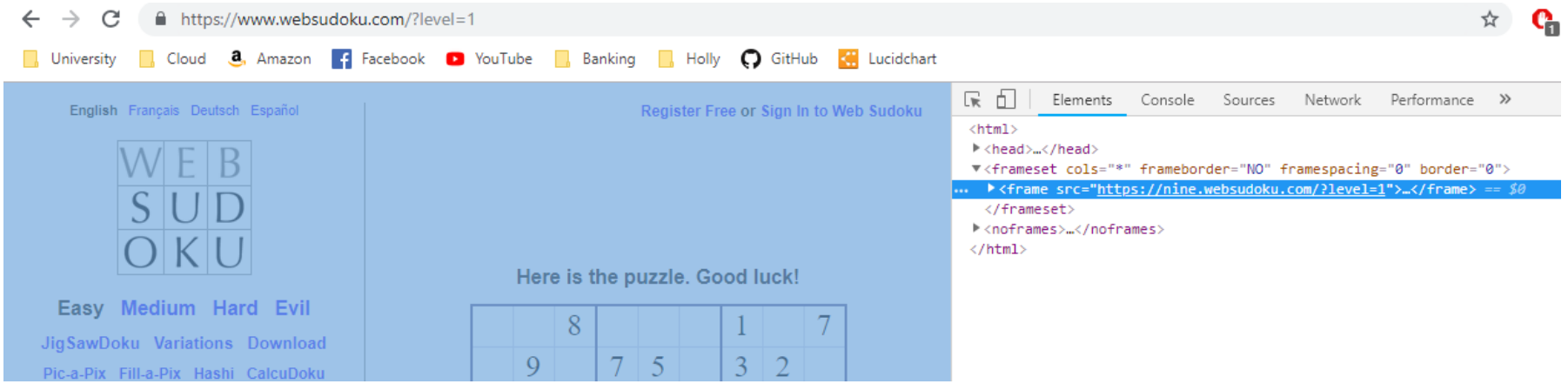


Figure 36-1. WebSudoku 'easy' puzzle HTML elements structure

However, upon closer inspection it can be seen that the, frameset element has a, 'src' attribute which points to another url, https://nine.websudoku.com/'; WebSudoku's proxy server.

The assumption was that the parent site made a call to a proxy site to retrieve its built puzzle. Therefore, the developer used the, 'Network' tab and reloaded the page.

When WebSudoku loads using the URL in figure 36-1, it makes a query call to its proxy server using the same query expressed in its URL, “?level=1” which can be seen below in figure 36-2 through the hover box.

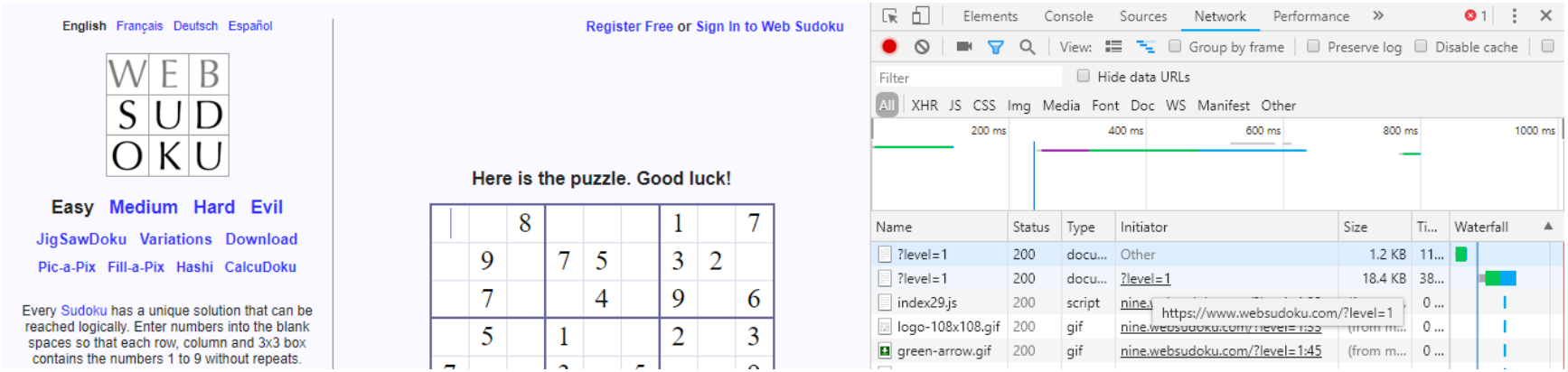


Figure 36-2. WebSudoku ‘easy’ puzzle retreival network activity

This confirms that that parent site calls the proxy site to generate/fetch a board which satisfies the query, “level=1”.

The src attribute on the frameset element matches the URL in the network’s activity, meaning that the frameset is populated using the proxy page call.

Figure 36-3 belows shows the result of entering the proxy's URL.

The puzzle is populated and is not a frameset element but rather a table element as the highlighted text shows. This was the format the developer was searching for and as such can be feasibly used for extraction using BeautifulSoup.

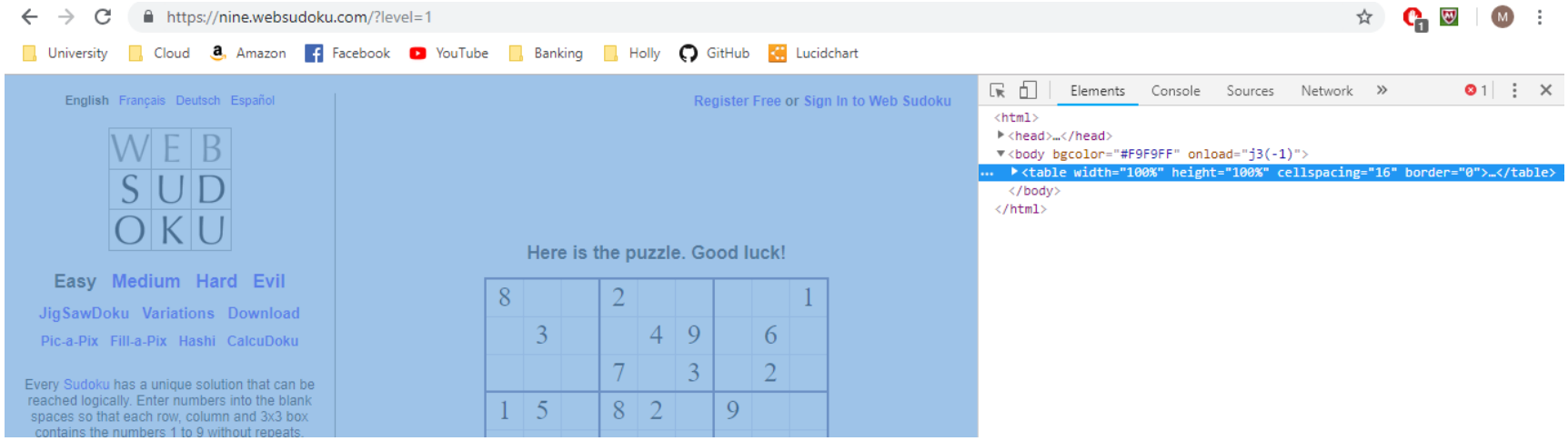


Figure 36-3. WebSudoku's proxy 'easy' puzzle HTML elements structure

Next, the structure of the table element had to be examined to extract the puzzle's values. To do this the developer selected a cell within the board in which the HTML element subsets were expanded upon until the element was isolated and visible. The adjacent parent elements were also examined to discover any groupings such as rows. The path to the cell element and any key adjacent elements were recorded as well as any key HTML attributes such as id' or 'class'.

This allowed the developer to use BeautifulSoup functions to isolate HTML items by id and then extract the value from those items see figure 37-1.

```
for td in grid.find_all('td'):
    for i in td.find_all('input'):
        try:
            puzzle.append(int(i['value']))
        except:
            puzzle.append(0)
```

Figure 37-1. Code snippet highlighting BeautifulSoup's filtering on HTML's input elements

Elements with no value attribute, i.e. undefined were set to 0 indicating a null cell.

In order for the algorithm to be validated that its calculated solution is the board's actual solution, this value had to be known. The solution array is recorded by the, 'cheat' input element as its value. Therefore, it simply needed to be extracted using BeautifulSoup's built-in functions as figure 37-2 shows.

```
solution = [int(d) for d in str(soup.find('input', {'id':'cheat'})['value'])]
```

Figure 37-2. Code snippet showing the Sudoku board's known solution being extracted.

9.0 System Verification

9.1. Reflection against Verification Plan

The original test cases laid out for the baseline requirements were mostly similar for verifying each component that was incorporated into the system.

The biggest difference laid in the front-end with the original layout tending towards a single page approach which encompassed fetching boards as well as playing Sudoku. The change to incorporate a 'home' page from which to navigate to the 'sudoku' page altered the original test cases, e.g. T1 stated, in its extreme test case for fetching a board, to do so whilst, 'the onscreen Sudoku board is not blank'. The changes to the GUI layout and structure meant that this case no longer existed and so was removed.
Test cases had to be drawn up for both mouse entry (which was erroneously not recorded) and the inclusion of showing solved boards in a new tab; an additional feature defined by a stakeholder through an accepted change form (see Appendix M). The valid, extreme and extreme cases for mouse input mode were created by simply moulding the existing keyboard entry cases for mouse entry, e.g. rather than, 'click an empty cell and enter the number 3' as would be a valid case for keyboard, the mouse case would be, 'click the number 3 from the number bar and then click an empty cell'. Cases may be inappropriate or additional ones needed, for this entry mode, e.g. the erroneous case of, 'entering the character, "a" into an empty cell' for keyboard entry cannot be replicated for mouse entry.
Test cases for the new feature had to be created and were done so in the same vain as the original test cases.

Carrying out verifying each component was done so, broadly, using the valid, extreme and erroneous test cases presented in the original plan.

Exceptions, again, being made with the undocumented input entry method (mouse entry) and the newly incorporated screenshot feature but once created the execution of verifying these features was done in the same manner.

9.2. Verification Results

Table 11 below shows the results of carrying out the test cases for each component including those from the original set and those derived during implementation.

Req#	*Test#*	*Description of Test*	*Data or Action Performed*	*Expected Behavior*	*Actual Behavior*	*Test Result*
R1	UT1	A board from the API is selected to be fetched	The radio button next to, 'Easy' is selected and the button, 'Fetch' is clicked	Page changes to the Sudoku page with an, 'Easy' board sourced	Page changes to the Sudoku page and an, 'Easy' board is present on-screen	●
R2	UT2	When in keyboard entry mode, attempt to remove a cell with a clue value	The cell is clicked on and backspace is entered	Clue remains	Cell cannot be highlighted for the key press to occur	●
	UT3		The cell is clicked on and a pasted whitespace is entered	Clue remains	Cell cannot be highlighted for the key press to occur	●
	UT4	When in keyboard entry mode, attempt to overwrite a cell with a clue value	Use same number	Clue remains	Cell cannot be highlighted for the key press to occur	●
	UT5		Use different number	Clue remains	Cell cannot be highlighted for the key press to occur	●
	UT6	When in mouse entry mode, attempt to overwrite a cell with a clue value	Use same number	Clue remains	Clue remains	●
	UT7		Use different number	Clue remains	Clue remains	●

R3	UT8	When in keyboard entry mode, attempt to insert a value into an empty cell	3	Number is added to cell	Number is added to cell	
	UT9		9	Number is added to cell	Number is added to cell	
	UT10		0	Number is rejected	Number is rejected	
	UT11		a	Input is rejected	Input is rejected	
	UT12	When in mouse entry mode, attempt to insert a value into an empty cell	Click an empty cell as soon as the page and board are loaded	1 is added to the cell	1 is added to the cell	
	UT13		Click 3 from the number bar and then click an empty cell	3 is added to the cell	3 is added to the cell	
R4	UT14	When in keyboard entry mode, attempt to remove a cell with a Sudoku value	The cell is clicked on and backspace is entered	Number is removed from cell	Number is removed from cell	
	UT15		The cell is clicked on and a pasted whitespace is entered	Input is rejected	Number is removed from cell	
	UT16	When in eraser mode, attempt to remove a cell with a Sudoku value	Click a cell with a value	Number is removed from cell	Number is removed from cell	

R5	UT17	When in keyboard entry mode, attempt to overwrite a cell with a Sudoku value	Use same number	Number is replaced with overwriting number	Number is replaced with overwriting number	●
	UT18		Use different number	Number is replaced with overwriting number	Number is replaced with overwriting number	●
	UT19		Paste number	Number is rejected	Number is replaced with copied number	●
	UT20	When in mouse entry mode, attempt to overwrite a cell with a Sudoku value	Use same number	Number is replaced with overwriting number	Number is replaced with overwriting number	●
	UT21		Use different number	Number is replaced with overwriting number	Number is replaced with overwriting number	●
R6	UT26	When in pencil mark mode, attempt to insert a value into an empty cell as a pencil mark	3	Pencil mark for number is added to cell	Pencil marks have not been incorporated into this product	●
	UT27		9	Pencil mark for number is added to cell	Pencil marks have not been incorporated into this product	●
R7	UT28	When in pencil mark mode, attempt to remove a value from a cell as a pencil mark	Number of pencil mark is entered	Pencil mark for number is removed from cell	Pencil marks have not been incorporated into this product	●

R8	UT29	A valid board is selected to be solved and displayed	An empty board is chosen to be solved	Board is solved with a valid solution	Solver returns no solution	🔴
	UT30		An API sourced 'easy' board is chosen to be solved	Board is solved with a valid solution	Board is solved with a valid solution	🟢
	UT31		An API sourced 'hard' board is chosen to be solved	Board is solved with a valid solution	Board is solved with a valid solution	🟢
	UT32		An API sourced board with correct values added to it is chosen to be solved	Board is solved with a valid solution	Board is solved with a valid solution	🟢
	UT33		A complete board is chosen to be solved	Board is displayed	Board is displayed	🟢
R9	UT34	Clues are differentiated from values by colour	An API sourced board is fetched	Clues appear in a different colour from user values	Numbers appear in a different colour from user values	🟢
R10	UT35	Pencil marks are added to cells and checked for neatness	1 is added as a pencil mark to a cell	1 is small and slightly transparent	Pencil marks have not been incorporated into this product	🔴
	UT36		1-9 are added as pencil marks to a cell	Numbers are small and slightly transparent	Pencil marks have not been incorporated into this product	🔴
R11	UT37	Values that are missing from the board are to be highlighted	An empty board is displayed on-screen	Numbers 1-9 are highlighted	Numbers 1-9 are illuminated	🟢
	UT38		A board with all 1s filled in is displayed on-screen	Numbers 2-9 are highlighted	Numbers 2-9 are illuminated	🟢
	UT39		A complete board	No numbers are highlighted	All numbers are not illuminated	🟢

R12	UT40	Values that have enough instances on the board are to be highlighted	An empty board is displayed on-screen	Numbers 1-9 are not dimmed	Numbers 1-9 are not dimmed	
	UT41		A board with all 1s filled in is displayed on-screen	Numbers 2-9 are not dimmed	Numbers 2-9 are not dimmed	
	UT42		A complete board	All numbers are dimmed	Numbers 1-9 are dimmed	
R13	UT43	A valid board is selected to be solved and displayed and the time taken for this action to complete is recorded	An empty board is chosen to be solved	Board is solved inside 1 minute	Solver returns no solution	
	UT44		An API sourced board is chosen to be solved	Board is solved inside 1 minute	Solver returns board inside 1 minute	
	UT45		An API sourced board with correct values added to it is chosen to be solved	Board is solved inside 1 minute	Solver returns board inside 1 minute	
	UT46		A complete board is chosen to be solved	Board is solved inside 1 minute	Solver returns board inside 1 minute	
	UT47		An invalid board is chosen to be solved	Board is recognized as invalid and returned	Board is recognized as invalid and returned immediately	
R14	UT48	Solved board is displays in a new tab	Solve button is clicked with valid puzzle on-screen	Board is displayed in a new tab	Board is displayed in a new tab	
	UT49		Complete board is clicked	Board is displayed in a new tab	Solve button must be clicked first before the solution is displayed in a new tab	

R17	UT50	Difficulties connecting to APIs are handled	Solver server is down	User is alerted of this	Alert is displayed to the user	●
	UT51		External puzzle generator server is down	User is alerted of this	Alert is displayed to the user	●
R18	UT52	Solver application should have 80% of its functionality tested and passing	Run q tests in session	Tests pass and cover 80% of the total number of functions	Test cases ran individually in a q session and all passing	●

Table 11. Verification tests and results

9.3. Conclusion

The system, having not implemented all of the stakeholders' agreed features, does not meet its requirements.

However, the core functionality of a Sudoku solver, does. Therefore, if this is considered the defining feature then the system could be argued to meets its needs.

The missing front-end pencil mark feature reduces the overall quality of the delivered product to a point where it could be argued that it is not adequate, however the other components present within the application are developed to an acceptable standard.

10.0 System Validation

10.1. Reflection against Validation Plan

The validation plan consisted of showcasing a prototype of the software to a panel of developers as well as the stakeholders through two separate Winter demo sessions.

These took place. The components, however to be validated by these users during these demo sessions were not the same as planned. Whilst the plan was executed in showcasing the core functionality – the solver – to the users, it did not bring up, and discuss, the UI.
This was discussed over meetings of the product over the course of sprints 6 and 7. By this stage, the developer had further familiarised themselves with the HCI of a system and incorporated this into their wireframes before presenting them to the end users.

Acceptance and usability meetings were also planned to take place at the end of each sprint (see Appendix C) in-order to validate the software.

However, due to other commitments with the developer these meetings were not held as regularly as every sprint. However, the develop did ensure that these took place every, at least, 3 sprints.

Performing these checks during the demo sessions happened as stated in the initial plan.

The demo sessions took place on the developer's laptop with all the software pre-installed.
The location for these demos took place in offices and home rooms which shared the characteristics outlined in the validation plan.

The few acceptance testing phases that took place did in accordance with the original plan.

The user was shown the software developed over the last few sprints and was asked for their comments and opinions on it.
If the software in question was UI related, then these meetings focused heavily on usability.
These meetings took place in the environment also defined in the plan.

10.2. Validation Results

Table 12 below shows usability testing of the delivered software at the end of sprint 7. The areas of interest / questions revolved around the use of the software and its delivery on all of the agreed baseline features.

Area of Interest	*Participant 1's Response and/or Action*	*Participant 2's Response and/or Action*	*Participant 3's Response and/or Action*	Test Result		
General use of the user interface – Home page						
Can you identify that this application allows for puzzles to be sourced?	Yes.	Yes.	Yes.	🟢	🟢	🟢
Can you identify that this application allows for puzzles to be custom created?	Yes.	Yes.	Yes.	🟢	🟢	🟢
Can you select for a puzzle to be sourced?	*Clicks source puzzle icon*	*Hovers over the source puzzle icon before clicking* Yes.	Yes.	🟢	🟢	🟢
Can you select a difficulty to be sourced?	*Clicks easy*	I'll choose easy	*Tabs to change radio button*	🟢	🟢	🟢
Can you select for a custom puzzle to be created?	*Clicks custom icon*	*Clicks custom icon*	Yes.	🟢	🟢	🟢
Playing with the user interface – Sudoku page						
Can you see the sourced puzzle?	Yes.	I can.	Yes.	🟢	🟢	🟢
Was this sourced in an appropriate amount of time?	Yes, it was fine.	Of course.	Yeah, there was barely a wait.	🟢	🟢	🟢
Can you see what method of input you are set as?	No.	I can't.	Is it the mouse icon? Yeah, but it's not easy to know.	🔴	🔴	🟠
Can you change this?	Do I just click it? *Clicks the mouse icon*	*Clicks mouse icon*	*Clicks mouse icon*	🟠	🟢	🟢
Can you select for pencil marks to be used?	*Clicks pencil icon*	Yeah, I'll just select the pencil icon.	Yes.	🟢	🟢	🟢

Can you select the eraser?	I can.	Yes.	No problem.	🟢	🟢	🟢
When using mouse entry, can you change the value to be entered into the board?	Do I use the keyboard? Oh right, is it just these numbers here [on the number bar] then?	Yes.	Yes.	🟠	🟢	🟢
When using mouse entry with pencil marks, can you change the note to be entered into the board?	Do I do the same as I did for the values?	*Clicks 6 on number bar*	Yes.	🟢	🟢	🟢
When using keyboard entry, can you enter a value into the board?	Yes.	Yes.	Yes.	🟢	🟢	🟢
When using keyboard entry, can you enter a string character into the board?	No.	No.	No.	🟢	🟢	🟢
When using keyboard entry, can you enter a number not present in the number bar?	No.	No, I get a warning.	Nope.	🟢	🟢	🟢
Can you differentiate between the sourced clues and your entered values?	Yes.	Yes.	Yes, it's very clear.	🟢	🟢	🟢
Can you erase any clue?	No.	No.	No.	🟢	🟢	🟢
Can you erase any entered value?	Yes.	Yes.	Yes.	🟢	🟢	🟢
Can you enter a pencil mark into a cell?	N/A	N/A	N/A	🔴	🔴	🔴
Are the pencil marks legible and adequate?	N/A	N/A	N/A	🔴	🔴	🔴
After entering all 1s into the board, can you distinguish that there are exhausted?	Oh yes, its disappeared from there [the number bar].	Yeah, it' been greyed out.	No.	🟢	🟢	🔴
After entering all the 1s into the board, can you erase an instance and see that they are no longer exhausted?	Oh look, yes, it's back again [on the number bar].	Yeah, that's nice I like that.	No.	🟢	🟢	🔴

Solver						
Can you see how to submit the onscreen board for solving?	Yes.	Yes.	Yes.	🟢	🟢	🟢
Is there adequate indication that the onscreen board is being solved?	Well it's pretty quick.	No but there's no need if it's that fast.	[No] solution loaded quick enough anyway.	🟢	🟢	🟢
Can you make it so that the solved board is displayed in a new tab?	Well it did it after I hit solve.	It already did [after hitting solve].	Yeah, it's just there *Clicks new tab*	🟢	🟢	🟢
Can you find the new tab the board is solved and displayed on?	Yes.	I can.	Yeah.	🟢	🟢	🟢
Is this displayed board adequate?	Yes.	Well it has missing numbers.	Yes.	🟢	🔴	🟢
Can you zoom in on the board in the new tab?	No.	Yeah.	Yes.	🔴	🟢	🟢
Miscellaneous						
Can you return to the home screen?	No.	Is it just the title?	*Clicks title*	🔴	🟠	🟢
Are there too many errors or display messages?	No.	No.	No.	🟢	🟢	🟢
Are the web pages consistently laid out?	Yes.	Yes.	Yeah.	🟢	🟢	🟢
Is the application initiative to use?	It's difficult for some parts but I could probably get use to how it works.	Well, I can use it and I'm not great on computers.	It's fidgety in places and maybe just missing a final polish but grand.	🟠	🟠	🟠

Table 12. Validation observations and results

The system experienced several fails with these validation checks.

The program fails to be intuitive regarding its input type. The users reported that they were not clear on what the default input type was, however, once it was pointed out to them, changing to the other entry method and using both, proved better results. This suggests that the application makes using the input methods rather intuitive, but it fails in pointing out initially to the user that this feature exists as it may be never before experience by the user or simply overlooked.

Returning to the home page from the Sudoku page was not obvious to the users. This suggests that explicit navigation buttons should be present to provide easier navigation between the pages.

Once the users triggered displaying the solved board in a new tab once user experienced that the board returned and displayed had 2 missing values. This problem was not able to be replicated consistently with the same, or any solved board, which indicates a parsing or visual glitch.
The ability to zoom in on the solved board in the new tab is solely down to the familiarity and experience of the user in the browser and not down to the product.
The program's ability to highlight the exhausted values for 1 user did not work as expected. This user made use of tab spacing to move between cells with affected the program's ability to count the number of instances.

10.3. Conclusion

Overall the application's ease of use for the users, which includes intuitiveness, proved to be adequate in its current state.

Changes could be made to the UI to aid in its intuitiveness such as: having an explicit zoom on the solved board tab and incorporating a 'tutorial' on start-up to highlight key features such as the method input option.
Issues would need to be resolved to improve the overall quality of the application. The parse or visual error could possibly be rectified by incorporating a wait to ensure the board has been parsed in its entirety before being displayed. The tab movement issue clearly does not set off the trigger in-place to return exhausted values and as such this would need looked into to solve.

Implementing these suggestions in future increments of the program would elevate its usable to the users.

Whilst including suggestions left out of the baseline requirements such as the ability to give hints, solve for different sized boards and incorporate a save and load feature, would increase the product's overall quality.
More advanced features not suggested by the user such as loading boards in by pictures or generating its own boards would further improve it.

11.0 Critical Evaluation

11.1. Critical Appraisal

The overall project is argued, by the developer, to be a success.

The core software functionality, which encompasses a wide array of technologies, was a chance to highlight the developer's software engineering skills. Understanding the capabilities and limitations of the software components used in the project such as libraries, frameworks and plugins within different programming languages enabled the developer to excel in achieving the ambitious aims set out at the start of the project.

The developer is also delighted with the diversity in technologies and programming paradigms incorporated into the final software.

Managing a sizeable project by a junior developer is also a huge undertaking. The developer is pleased with their usage of software management tools such as version control and issue tracking; utilising their interconnectivity led to well organised and traceable code which proved pivotal during busy code periods, as evident in Appendix G.

Their personal time management, organisational skills and continuous motivation to the project was also an impressive feat.

However, the project is not without its faults.

The limited planned interactions with stakeholders due to the developer's other commitments was a disappointment and could be attributed with the lacklustre validation results.

Not having time to incorporate the pencil marks feature into the delivered system on its agreed date showcases the misjudged management of the product, despite the developer using adequate software and tools.

The developer is also disappointed with the flexibility of the product; the ability to solve for different sized boards. Whilst the back-end was developed for this the developer's inexperience on front-end meant that it was not.

11.2. Reflection against Initial Project Plan

The original project plan modelled the project moderately well.

However, with the improvement suggestions during the Winter demonstration period a revised plan was needed as shown in section 6.2 which incorporated these changes well and modelled the project accurately.

The planning stage of the plan stated that the developer would upskill in the technologies anticipated to be used throughout the project.

The developer undergone a full-stack development course in which they transferred their learnt knowledge in Python Flask servers and HTML and JS into the project.

However, whilst the Flask server and JS training was suitable for this project, the skills in HTML and CSS were not. This led to more time and resources being used when developing the front-end of the system which ultimately resulted in it not being incorporated fully.

During the Winter demo session, a web scraper was suggested to be developed which involved using a Python's BeautifulSoup library. This was previously unknown to the developer which meant that they had little time to read up and use the module before implementing it into the system.

The analysis stage of the project was modelled well.

Splitting-up the requirements gathering activity mapped how these suggestions were obtained perfectly as the author had to set up meetings with all the stakeholders separately which took place over a number of days.

However, the back-end and front-end deigns created during this period were not well designed and had to be modified throughout before the final design as you see in section 7.1. This highlighted the developer's inexperience with designing system architectures.

The implementation stage did take place in 7 planned, approximate 2 weeks sprints.

The main development milestones were started on the sprints outlined in the revised Gantt chart in section 6.2.

However, on occasion these software components 'split-over' into the adjacent sprint impacting that sprint. The sprints rarely took the form outlined in Appendix C as acceptance testing was often not carried out due to the developer's other commitments.

The planned change freezes and delivery dates for deliverables remained fixed throughout.

11.2.1. Time and Effort Estimations

The time and effort for each software component and sub-components were managed by the use of the Excel issue tracker (see Appendix F).

The issue tracker maintained details such as 'item duration' for each issue. However, these values were arbitrary and thought of by the developer based on past-experiences. However, with the developer being a junior this often meant drawing on past experiences that were irrelevant or non-existent.
This resulted in many of the issues over-exceeding their recorded estimated work hours which had knock-on effects as tasks occasionally were needing completed in the following sprint, effecting the software component to be developed during it.
The impacts of these inaccurate timings were profound during the final sprint in which the developer was tasked of developing the front-end. This sprint was crucial due to its proximity the final delivery date. The developer, as stated had undergone a full-stack development course in-order to improve their front-end development prowess, however this was lacklustre. This resulted in the front-end not being developed fully or to a particularly high standard, resulting in many of the validation fails.
Work items within the issue tracker did not incorporate any difficulty or complexity matrix such as Fibonacci numbers, to indicate time and effort allocation. This may have also been a factor in tasks not being developed on-time.
Setting up the entirely manual issues for tracking was tedious and time-consuming.

11.2.2. Software Methodologies

The use of agile within this project was performed in a below standard manner. The project did break its key features into small, manageable work items, however, these were not drawn up particularly well as user stories and failed to indicate the complexity of each activity. Whilst this was tolerable in this one-person project (as the developer knew what each task referred to and the time to be spent on them), it would be inadequate in a real-world scenario with multiple developers.
However, the management of the tasks was handled extremely well with the use of a real-life Kanban board, especially when they were many different tasks ongoing at once (see Appendix G). This allowed the developer to see upcoming work along with progress which provided motivation. The use of work in progress (WIP) limits on certain stations meant that the developer was forced to finish with work items, before starting new ones. This resulted in a relatively constant 'flow' of tasks moving across the board. The board was also placed outside of the developer's work area which allowed them to take breaks away from software development, improving concentration and work productivity; on occasions the developer came up with more suitable ways of dealing with current difficulties during these periods.
The backlog of issues was recorded and managed using a custom created Excel issue tracker (see Appendix G). The problems with this tool was that it was too manual. The issues to be tracked and progress on these had to be inserted manually which was tedious and led to records being forgotten to be included by the developer, leading to unreliable data. Jira (on which the Excel issue tracker was based on) would have been a better way of managing these with its automated integration of its source code management system, BitBucket (something the Excel worksheet could never replicate).

12.0 Bibliography

Aiden, H. (2006) Anything but square: from magic squares to Sudoku. Plus Maths [online], 01 March. Available from: https://plus.maths.org/content/anything-square-magic-squares-sudoku [Accessed: 2018-Oct-18].

Agile Alliance (2019) 12 Principals Behind the Agile Manifesto. Available from: https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/. [Accessed: 2019-Jan-29].

Balosin, I. (2017) The Art of Crafting Architectural Diagrams. Info [online], 04 August. Available from: https://www.infoq.com/articles/crafting-architectural-diagrams [Accessed: 2019-Feb-02].

Bates, J. (2015) Brits will spend 12 days onboard flights in a lifetime. Airport World [online], 23 September. Available from: http://www.airport-world.com/news/general-news/5181-brits-will-spend-12-days.html. [Accessed: 2018-Nov-22].

BBC (2016) Two-hour daily commute 'on rise among UK workers'. Available from: https://www.bbc.co.uk/news/uk-38026625. [Accessed: 2018-Nov-02].

Colblindor (2019) Coblis – Color Blindness Simulator. Available from: https://www.color-blindness.com/coblis-color-blindness-simulator/. [Accessed: 2019-Feb-18].

Colour Blind Awareness (2019) Home. Available from: http://www.colourblindawareness.org/. [Accessed: 2019-Feb-16].

Cornell University (2009) The Math Behind Sudoku. Available from: http://pi.math.cornell.edu/~mec/Summer2009/Mahmood/Home.html [Accessed: 2018-Nov-12].

Cosmos Magazine (2015). Magic square. [image] Available from: https://cosmos-magazine.imgix.net/file/spina/photo/2602/231115_England_square1.png?ixlib=rails-1.1.0&h=401&w=657 [Accessed: 2018-Oct-17].

Davis, T. (2008) The Mathematics of Sudoku. Stanford Math Circle. Available from: http://precollegiate.stanford.edu/circle/math/notes08f/sudoku.pdf [Accessed: 2018-Nov-12].

Easybrain (2018) Why is Sudoku One of the Most Popular Games for Seniors? Available from: http://www.sudoku.com/how-to-play/why-is-sudoku-one-of-the-most-popular-games-for-seniors/. [Accessed: 2018-Nov-20].

England, J. (2015) How to solve a magic square. Cosmos Magazine [online], 23 November. Available from: https://cosmosmagazine.com/mathematics/how-solve-magic-square [Accessed: 2018-Oct-11].

Exeter Mathematics School (2016) The Mathematics of Sudoku. Available from: https://www.youtube.com/watch?v=loMZ6klLYtc [Accessed: 2018-Nov-02].

Gao, L. (2005) Latin Squares in Experimental Design. Michigan State University. Available from: http://compneurosci.com/wiki/images/9/98/Latin_square_Method.pdf [Accessed: 2018-Oct-09].

Gould, W. (2004) Sudoku (Version 1.1) [computer program]. Available from: http://waynegouldpuzzles.com/sudoku/ [Accessed: 2018-Oct-23].

Gould, W. (2018). Wayne Gould Puzzles. Available from: http://waynegouldpuzzles.com/sudoku/. [Accessed: 2018-Oct-18]

Grime, J. (2012) 17 and Sudoku Clues - Numberphile. Numberphile. Available from: https://www.youtube.com/watch?v=MlyTq-xVkQE [Accessed: 2018-Sep-29].

hackerdashery (2014) P vs. NP and the Computational Complexity Zoo. hackerdashery. Available from: https://www.youtube.com/watch?v=YX40hbAHx3s [Accessed: 2018-Nov-23].

Hinkler Books (2014) . Perfect Puzzles Sudoku. Hinkler Books.

Institution of Mechanical Engineers (2018) STEM skills gap costs the UK £1.5bn a year. Available from: https://www.imeche.org/news/news-article/stem-skills-gap-costs-the-uk-1.5bn-a-year. [Accessed: 2018-Nov-02].

Jones et al (2012) The Structure of Reduced Sudoku Grids and the Sudoku Symmetry Group. International Journal of Combinatorics, 2012. http://dx.doi.org/10.1155/2012/760310

Kemmochi and Mikami. (2007) Sudoku 'godfather' wants to puzzle more. The Japan Times [online], 16 June. Available from: https://www.japantimes.co.jp/news/2007/06/16/national/sudoku-godfather-wants-to-puzzle-more/#.W_VrD6f7SUl. [Accessed: 2018-Nov-16].

kevinhhl. (2013) Sudoku Solver (Version 1.3) [android app]. Available from: https://play.google.com/store/apps/details?id=com.gmail.kevinhhldev.sudokusolver [Accessed: 2018-Oct-20].

PopulationPyramid.net (2018). Population Pyramids of the World from 1950 to 2100. [image] Available from: https://www.populationpyramid.net/japan/1975/ [Accessed: 2018-Sep-22].

PopulationPyramid.net (2018). Population Pyramids of the World from 1950 to 2100. [image] Available from: https://www.populationpyramid.net/japan/2015/ [Accessed: 2018-Sep-22].

Randstad (2013) Savvy "Super Commuters" spend time wisely. Available from: https://www.randstad.co.uk/about-us/press-releases/randstad-news/savvy-super-commuters-spend-time-wisely/. [Accessed: 2018-Nov-02].

Salter, J. 2015. Helpful Apps for Seniors. Next Avenue. Available from: https://www.nextavenue.org/8-great-apps-for-our-elders/. [Accessed: 2018-Nov-16].

Sieger, C. (2013) The Surprising Origins of Sudoku - Stuff of Genius. Stuff of Genius – HowStuffWorks. Available from: https://www.youtube.com/watch?v=Hku_zGTIOn0 [Accessed: 2018-Oct-04].

Smith, D. (2005) So you thought Sudoku came from the Land of the Rising Sun… The Guardian [online], 15 May. Available from: https://www.theguardian.com/media/2005/may/15/pressandpublishing.usnews. [Accessed: 2018-Oct-09].

Sommerville, I. (2011) Software Engineering. 9th ed. Boston, MA: Addison-Wesley.

Sudoku of the Day (2015). Sudoku puzzle. [image] Available from: https://www.sudokuoftheday.com/wp-content/uploads/2015/06/TrickyPuzzle-1030x1030.png [Accessed: 2018-Sep-22].

Sudoku Solutions (2018) Online Sudoku Solver and Helper. Available from: http://www.sudoku-solutions.com/. [Accessed: 2018-Oct-22].

Telwalker, P. (2015). How Many 3×3 Magic Squares Are There? Sunday Puzzle. Mind Your Decisions. Available from: https://mindyourdecisions.com/blog/2015/11/08/how-many-3x3-magic-squares-are-there-sunday-puzzle/ [Accessed: 2018-Oct-22].

The Guardian (2005) G2, home of the discerning Sudoku addict. Available from: https://www.theguardian.com/theguardian/2005/may/13/features11.g2. [Accessed: 2018-Nov-03].

The Telegraph (2018) Telegraph Puzzles. Available from: https://puzzles.telegraph.co.uk/search_results. [Accessed: 2018-Sep-05].

Tuchkov, I. (2018). Color blindness: how to design an accessible user interface. Medium [online], 22 August. Available from: https://uxdesign.cc/color-blindness-in-user-interfaces-66c27331b858. [Accessed: 2019-Feb-15].

Walters, R. (2018) Solving the UK Skills Shortage. Jobsite. Available from: https://www.jobsite.co.uk/news/wp-content/uploads/2018/08/Solving-the-UK-Skill-Shortage-PDF-1.pdf [Accessed: 2018-Nov-03].

Watkin, C. (2017) Minimum computer specs for general users? CNET [online], 23 March. Available from: https://www.cnet.com/forums/discussions/minimum-computer-specs-for-general-users/. [Accessed: 2018-Oct-09].

Web Sudoku (2018) Web Sudoku. Available from: https://www.websudoku.com/. [Accessed: 2018-Oct-28].

Weisstein, E.W. (2018) Latin Square. Available from: http://mathworld.wolfram.com/LatinSquare.html [Accessed: 2018-Oct-02].

Wilson, R.J. (2006). How to solve Sudoku: A Step-by-step guide. New York: Sterling.

World Puzzle Federation (2018) World Sudoku Championships. World Puzzle Federation. Available from: http://www.worldpuzzle.org/championships/wsc/ [Accessed: 2018-Oct-14].

York University (2013). Latin square. [image] Available from: https://www.yorku.ca/mack/RN-Counterbalancing-f3.jpg [Accessed 2018-Oct-16].

13.0 Appendix

13.1. Requirements Appendix

Requirement Number:	R1	Requirement Type:	Functional
Description:	Puzzles from an API can be fetched as either: easy, medium or hard difficulty		
Rationale:	Allows users to play Sudoku		
Originator:	Stakeholder – Sudoku group		
Fit Criteria:	The application must be able to fetch Sudoku puzzles in each of the 3 difficulties: easy, medium and hard		
Priority:	#15	Dependencies:	-
Supporting Materials:	-		
History:	Original baseline requirement *creation* [09/11/2018], Revised baseline requirement *alteration* [01/02/2019] from: Description: Puzzles are fetched from an API to: Description: Puzzles from an API can be fetched as either: easy, medium or hard difficulty **justification: remove ambiguity**		

Requirement Number:	R2	Requirement Type:	Functional
Description:	Clues cannot be altered		
Rationale:	Prevents originally valid boards being made invalid		
Originator:	Developer		
Fit Criteria:	The user is unable to remove, overwrite or in any way alter any original starting clue of a board		
Priority:	#12	Dependencies:	-
Supporting Materials:	-		
History:	Original baseline requirement *creation* [09/11/2018]		

Requirement Number:	R3	Requirement Type:	Functional
Description:	Values can be added to cells		
Rationale:	Allows users to play Sudoku		
Originator:	Stakeholder – Sudoku group		
Fit Criteria:	The user is able to enter any valid Sudoku number into an empty cell		
Priority:	#1	Dependencies:	-
Supporting Materials:	-		
History:	Original baseline requirement *creation* [09/11/2018]		

Requirement Number:	R4	Requirement Type:	Functional
Description:	Values can be removed from cells		
Rationale:	Allows mistaken values to be removed		
Originator:	Stakeholder – Sudoku group		
Fit Criteria:	The user is able to remove any previously entered value from any cell		
Priority:	#1	Dependencies:	R3
Supporting Materials:	-		
History:	Original baseline requirement *creation* [09/11/2018]		

Requirement Number:	R5	Requirement Type:	Functional
Description:	Values can be overwritten in populated cells		
Rationale:	Quick way to overwrite mistakes, adding to the playing experience		
Originator:	Developer		
Fit Criteria:	The user is able to enter another valid Sudoku value into cell with a previously entered value, including the same value		
Priority:	#2	Dependencies:	R3
Supporting Materials:	-		
History:	Original baseline requirement *creation* [09/11/2018]		

Requirement Number:	R6	Requirement Type:	Functional
Description:	Pencil marks can be added to cells		
Rationale:	Players' working out can be added, adding to the playing experience		
Originator:	Stakeholder – Sudoku group		
Fit Criteria:	The user is able to enter any valid Sudoku value into an empty cell as a pencil mark and are able to do so for multiple valid Sudoku values		
Priority:	#8	Dependencies:	-
Supporting Materials:	Appendices I & K		
History:	Original baseline requirement *creation* [09/11/2018]		

Requirement Number:	R7	Requirement Type:	Functional
Description:	Pencil marks can be removed from cells		
Rationale:	Players' working out can be rectified, adding to the playing experience		
Originator:	Stakeholder – Sudoku group		
Fit Criteria:	The user is able to remove a pencil mark value from a cell containing that value		
Priority:	#8	Dependencies:	R6
Supporting Materials:	Appendices I & K		
History:	Original baseline requirement *creation* [09/11/2018]		

Requirement Number:	R8	Requirement Type:	Functional
Description:	80% of valid puzzles can be solved		
Rationale:	Defining feature to deceiver whether the application is useful		
Originator:	Developer		
Fit Criteria:	The solver is 80% likely to solve any given board and is tested by supplying 1000 randomly sourced boards of varying difficulty		
Priority:	#16	Dependencies:	-
Supporting Materials:	-		
History:	Original baseline requirement *creation* [09/11/2018], Revised baseline requirement *alteration* [01/02/2019] from: Description: Valid puzzles should be solved to: Description: 80% of valid puzzles can be solved **justification: remove ambiguity and make testable**		

Requirement Number:	R9	Requirement Type:	Non-functional
Description:	Clues are a different colour to values		
Rationale:	Adds to the user experience by providing useful assistance		
Originator:	Stakeholder – Sudoku group		
Fit Criteria:	The original starting clues are a disguisable different colour to that of any user entered values		
Priority:	#4	Dependencies:	R2, R3
Supporting Materials:	-		
History:	Original baseline requirement *creation* [09/11/2018]		

Requirement Number:	R10	Requirement Type:	Non-functional
Description:	Pencil marks are small and do not clutter cells		
Rationale:	Allows users to see cell values and pencil marks at all times, adding to their playing experience		
Originator:	Developer		
Fit Criteria:	Entering all of the valid Sudoku values into any given empty cell does not exceed the cells' boundaries and all values are legible and can be read clearly		
Priority:	#5	Dependencies:	R6, R7
Supporting Materials:	Appendix K		
History:	Original baseline requirement *creation* [09/11/2018]		

Requirement Number:	R11	Requirement Type:	Non-functional
Description:	Values remaining to be filled in are displayed		
Rationale:	Adds to the user experience by providing useful assistance		
Originator:	Developer		
Fit Criteria:	All of the valid Sudoku values are clearly and legibly displayed to the user		
Priority:	#8	Dependencies:	R2, R3, R4, R5
Supporting Materials:	Appendix K		
History:	Original baseline requirement *creation* [09/11/2018]		

Requirement Number:	R12	Requirement Type:	Non-functional
Description:	Exhausted values are highlighted		
Rationale:	Adds to the user experience by providing useful assistance		
Originator:	Developer		
Fit Criteria:	Any valid Sudoku value which appears the maximum number of times within the board is distinguishable from the remaining other Sudoku values		
Priority:	#3	Dependencies:	R11
Supporting Materials:	Appendix K		
History:	Original baseline requirement *creation* [09/11/2018]		

Requirement Number:	R13	Requirement Type:	Non-functional
Description:	Solvable puzzles are solved inside 1 minute, regardless of difficulty		
Rationale:			
Originator:	Developer		
Fit Criteria:	For any given board – which is solvable – is solved by the application ad displayed to the user in under 60 seconds		
Priority:	#14	Dependencies:	R8
Supporting Materials:	-		
History:	Original baseline requirement *creation* [09/11/2018], Revised baseline requirement *alteration* [01/02/2019] from: Description: Any difficulty puzzle is solved inside 1 minute to: Description: Solvable puzzles are solved inside 1 minute, regardless of difficulty **justification: remove ambiguity**		

Requirement Number:	R14	Requirement Type:	Non-functional
Description:	Clicking a solved board displays it in a new tab		
Rationale:	Allow for cross-checking of attempted puzzle solves against its solution		
Originator:	Stakeholder		
Fit Criteria:	The user is able to click any returned solved board and have the same board displayed in a new tab on the same browsing window		
Priority:	#7	Dependencies:	R8
Supporting Materials:	Appendix M		
History:	Revised baseline requirement *creation* [01/02/2019]		

Requirement Number:	R15	Requirement Type:	Non-functional
Description:	Web interface should be initiative to use		
Rationale:	Allow users of any technical ability to feel comfortable using the system		
Originator:	Developer		
Fit Criteria:	The user, at no time when operating the web application, feel confused or disorientated and after a number of uses, should be able to navigate and perform activities quicker than previously		
Priority:	#6	Dependencies:	-
Supporting Materials:	-		
History:	Original baseline requirement *creation* [09/11/2018]		

Requirement Number:	R16	Requirement Type:	Non-functional
Description:	Web interface should be consistent across webpages		
Rationale:	Aid in the user's familiarity of the product to enhance user experience		
Originator:	Developer		
Fit Criteria:	The webpages should have a similar theme, styling and overall feel across each webpage		
Priority:	#13	Dependencies:	-
Supporting Materials:	-		
History:	Revised baseline requirement *creation* [01/02/2019]		

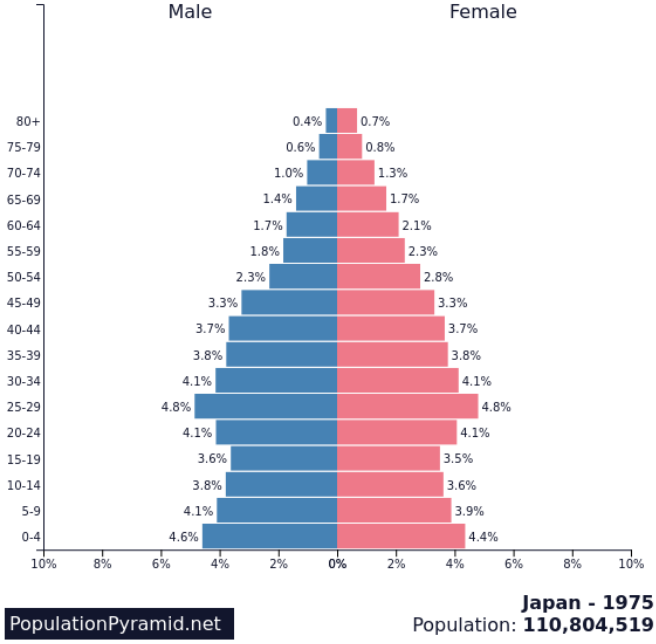
Requirement Number:	R17	Requirement Type:	Non-functional
Description:	Difficulties connecting to APIs should be handled		
Rationale:	To prevent erroneous functionality of the system which can hinder the user's experience		
Originator:	Developer		
Fit Criteria:	At no point should any errors communicating with the incorporated API be highlighted to the user in an inappropriate and menacing manner, including: crashing or freezing the web browser, error messages – which have not been designed by the application, displaying a Sudoku board with incomplete or inaccurate values		
Priority:	N/A	Dependencies:	R1
Supporting Materials:	-		
History:	Original baseline requirement *creation* [09/11/2018]		

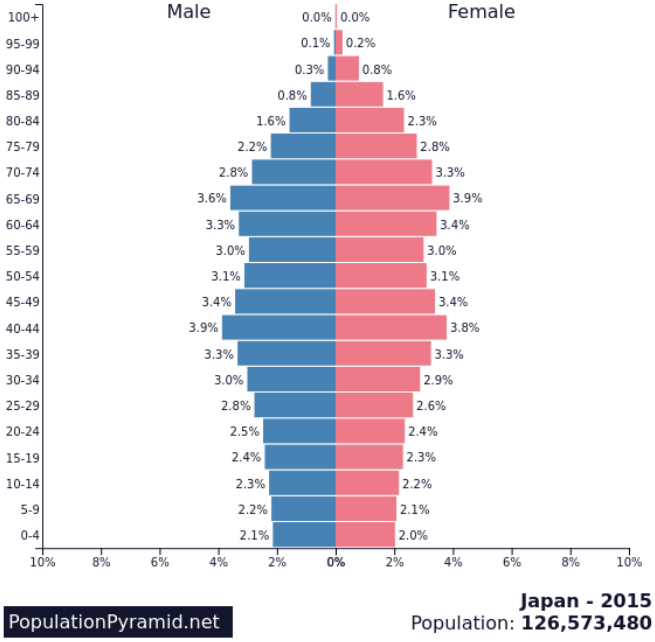
Requirement Number:	R18	Requirement Type:	Non-functional
Description:	Application should have 80% of its functionality tested and passing		
Rationale:	To ensure that the program works as expected in a plethora of scenarios		
Originator:	Developer		
Fit Criteria:	80% of the application's functions are manually tested and checked for success in a test environment		
Priority:	N/A	Dependencies:	-
Supporting Materials:	-		
History:	Original baseline requirement *creation* [09/11/2018], Revised baseline requirement *alteration* [01/02/2019] from: Description: Application should be well tested to: Description: Application should have 80% of its functionality tested and passing **justification: make testable**		

13.2. Additional Artefacts

Appendix A

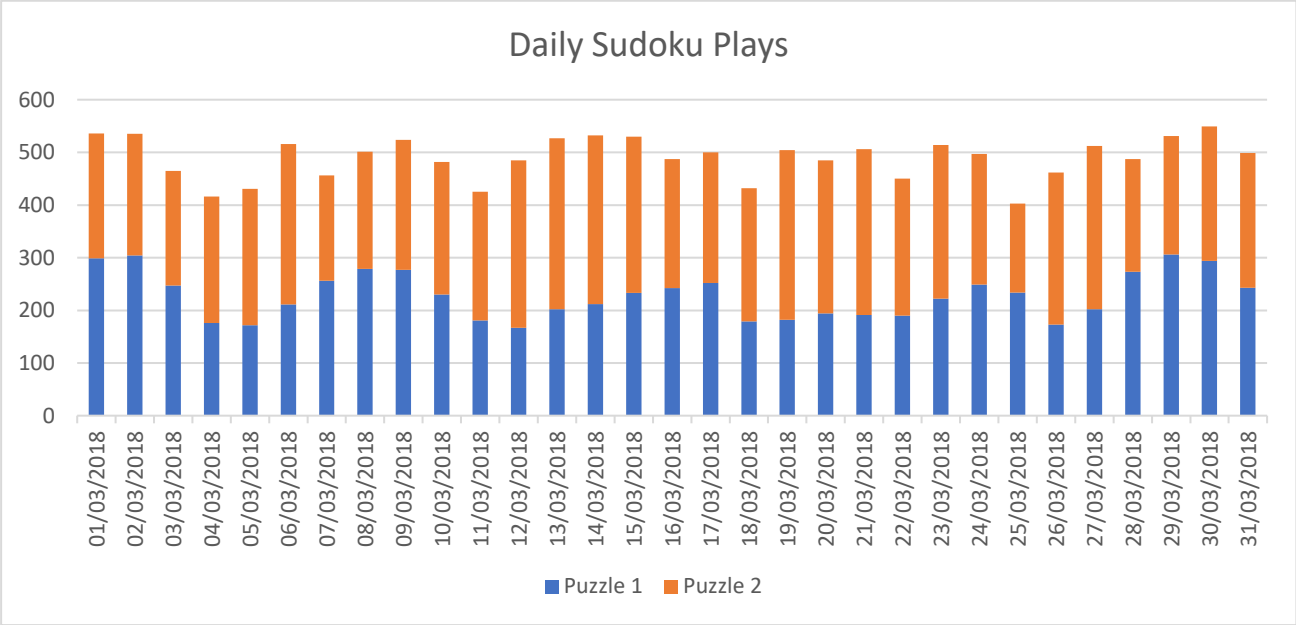
Japan's population pyramids for the year 1975 and 2015 obtained from PopulationPyramid.net (2019).





Appendix B

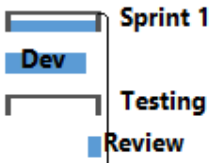
The daily Sudoku plays from online Telegraph (2018) puzzles across the month of March 2018.



Appendix C

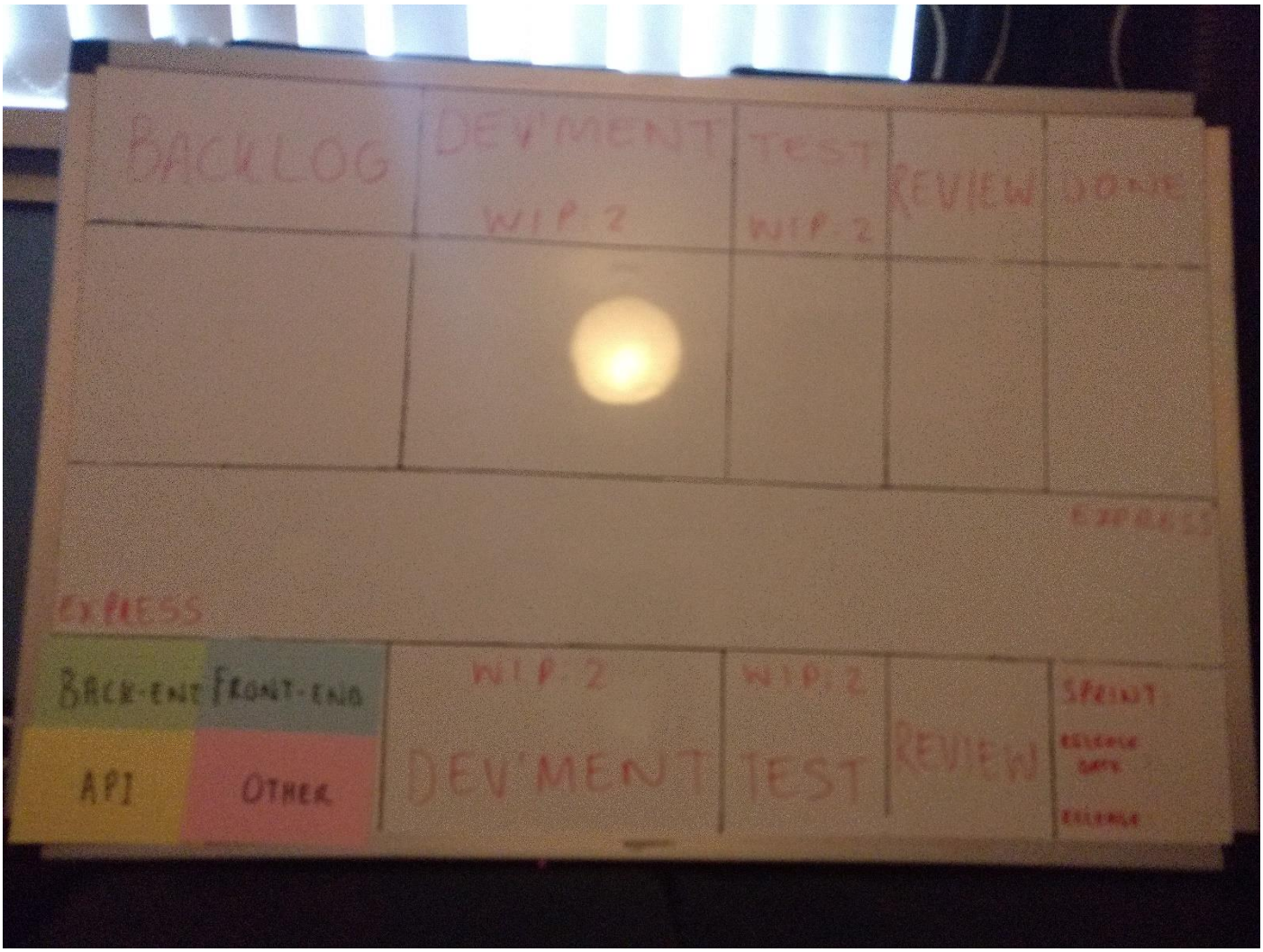
The sprints used throughout the project follow the following structure.

Sprint 1	**10 days**	**Mon 12/11/18**	**Fri 23/11/18**
Development	8 days	Mon 12/11/18	Wed 21/11/18
Testing	**10 days**	**Mon 12/11/18**	**Fri 23/11/18**
Sprint Review	1 day	Fri 23/11/18	Fri 23/11/18



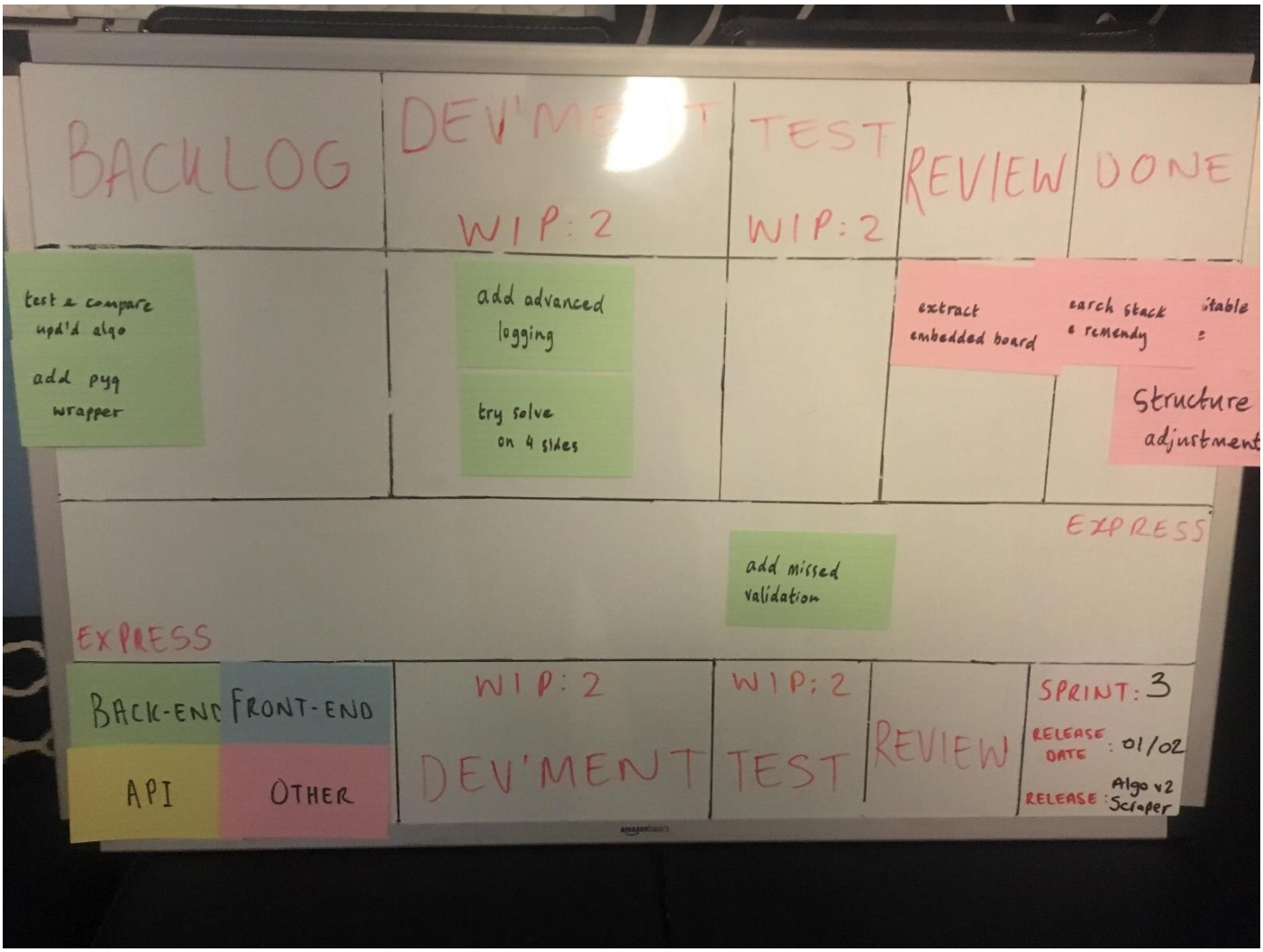
Appendix D

The real-life Kanban board setup used throughout the project.



Appendix E

Kanban in-use during sprint 3.



Appendix F

Issue tracking setup in Excel used throughout the project.

Item Number:	1
Item Name:	item_showcase
Item Priorty:	medium
Item Type:	feature
Item Duration:	8h
Item Start Date:	19/09/2018
Links To:	-
Item Description:	item created to test my sy
GIT Branch:	feature:item_showcase

Start Date:	19/09/2018
Duration (hrs):	8
Est. End Date:	19/09/2018
Work Done (hrs):	8
Est. End Date:	18/04/2019
Finished:	TRUE

ITEM TRACKING SYSTEM

Item Number:	1
Done:	TRUE
Date:	19/09/2018
Work Done (hrs):	6
Git Commit(s):	34drc2
Comment:	initial development of my service
Date:	20/09/2019
Work Done (hrs):	2
Git Commit(s):	6sd42
Comment:	finalising my service

Appendix G

Issue tracker in use during sprint 3.

Item Number:	16	Start Date:	21/01/2018
Item Name:	structure_adjustment	Duration (hrs):	16
Item Priorty:	low	Est. End Date:	22/01/2018
Item Type:	feature		
Item Duration:	2d	Work Done (hrs):	5
Item Start Date:	21/01/2018	Est. End Date:	21/01/2018
Links To:		Finished:	TRUE
Item Description:	adjust the current setup t		
GIT Branch:	feature+structure_adjustr		

Item Number:	17	Start Date:	23/01/2018
Item Name:	create_web_scraper	Duration (hrs):	24
Item Priorty:	low	Est. End Date:	25/01/2018
Item Type:	feature		
Item Duration:	3d	Work Done (hrs):	15
Item Start Date:	23/01/2018	Est. End Date:	23/01/2018
Links To:		Finished:	TRUE
Item Description:	create a web scraper to at		
GIT Branch:	feature+create_web_scrap		

Item Number:	18	Start Date:	23/01/2018
Item Name:	adjust_algo_to_deal_with	Duration (hrs):	72
Item Priorty:	high	Est. End Date:	31/01/2018
Item Type:	feature		
Item Duration:	9d	Work Done (hrs):	90
Item Start Date:	23/01/2018	Est. End Date:	21/01/2018
Links To:		Finished:	TRUE
Item Description:	research and find a soluti		
GIT Branch:	feature+adjust_algo_to_d		

Item Number:	19	Start Date:	04/02/2018
Item Name:	updated_algo	Duration (hrs):	64
Item Priorty:	high	Est. End Date:	11/02/2018
Item Type:	test		
Item Duration:	8d	Work Done (hrs):	46
Item Start Date:	04/02/2018	Est. End Date:	04/02/2018
Links To:		Finished:	TRUE
Item Description:	test the updated algo agai		
GIT Branch:	test+updated_algo		

Item Number:	20	Start Date:	18/02/2019
Item Name:	logging	Duration (hrs):	24
Item Priorty:	low	Est. End Date:	20/02/2019
Item Type:	feature		
Item Duration:	3d	Work Done (hrs):	12
Item Start Date:	18/02/2019	Est. End Date:	18/02/2019
Links To:		Finished:	TRUE
Item Description:	create log files for algo		
GIT Branch:	feature+logging		

Item Number:	16
Done:	TRUE
Date:	21/01/2018
Work Done (hrs):	5
Git Commit(s):	c93b8c2
Comment:	basic structure setup

Item Number:	17
Done:	TRUE
Date:	26/01/2019
Work Done (hrs):	15
Git Commit(s):	1c94c92
Comment:	scraper added + ran 1000 times

Item Number:	18
Done:	TRUE
Date:	29/01/2019
Work Done (hrs):	90
Git Commit(s):	978c558
Comment:	no solution to stack issue - boards now rotated 4 times and attempted to be solved at each

Item Number:	19
Done:	TRUE
Date:	16/02/2019
Work Done (hrs):	46
Git Commit(s):	3f326bc
Comment:	adjusted solver tested and compared

Item Number:	20
Done:	TRUE
Date:	24/02/2019
Work Done (hrs):	12
Git Commit(s):	8258fa9
Comment:	advanced logging added

Appendix H

Interview transcript used during the interviews with the business analyst and the stakeholders to gather needs and constraints of the system.

Interviewee: _____ Date: / /

Interviewer: _____

Recreational Background

Understand their leisure background and the activities they enjoy playing.
e.g. what recreational activities do you get up to, during work breaks do you participate in any recreational activities, how do you pass time when commuting, etc.

Sudoku Involvement

Gain an understanding of the level of involvement and skill the interviewee has with Sudoku.
e.g. how often they play Sudoku, what difficulty is best suited to them, etc.

1

Technology Usage Understand the usage of technology used by the individual. e.g. do you own a PC, do you have internet, are you confident in using a PC to browse the internet, do you use a mobile device and what do you use it for, etc.

Other

Appendix I

Completed interview transcript for one of the stakeholders.

Interviewee: GEORGE T. Date: 25/10/2018

Interviewer: MT

Recreational Background

Understand their leisure background and the activities they enjoy playing.
e.g. what recreational activities do you get up to, during work breaks do you participate in any recreational activities, how do you pass time when commuting, etc.

- PLAYS PUZZLES IN NEWSPAPER → READS NEWSPAPER DAILY
- GOES ON WALKS
- RETIRED
- WORKED PREV. IN I.T.
- OWN SUDOKU

SUDOKU, CROSS-WORDS, WORD PUZZLES

BOOKS => ENJOYS PROBLEM SOLVING

AIR FLIGHTS

↳ WOULD BRING ALONG, BUY NEWSPAPER FOR COMMUTES

Sudoku Involvement

Gain an understanding of the level of involvement and skill the interviewee has with Sudoku.
e.g. how often they play Sudoku, what difficulty is best suited to them, etc.

- DAILY THROUGH NEWSPAPER
- EASY IMED OCCASIONALLY HARD → WILL LOOK UP ANS TO HELP HARDS
- DOESN'T TIME THEMSELVES
↳ MAY COME BACK TO PARTLY SOLVED → THESE ONES PRESENT IN PUZZLE BOOKS
- OCCASSIONALLY WRITES HELPER NOTES
WOULDN'T DO SO FOR NEWSPAPERS

- LOGICALLY WORKS THRO. EACH #
- THEIR HELPER NOTES OFTEN CONFUSED THEM FOR ACC. ANS

1

Technology Usage

Understand the usage of technology used by the individual.
e.g. do you own a PC, do you have internet, are you confident in using a PC to browse the internet, do you use a mobile device and what do you use it for, etc.

- PREV'LY IN I.T. (+20 YRS)
  ↳ COBALT

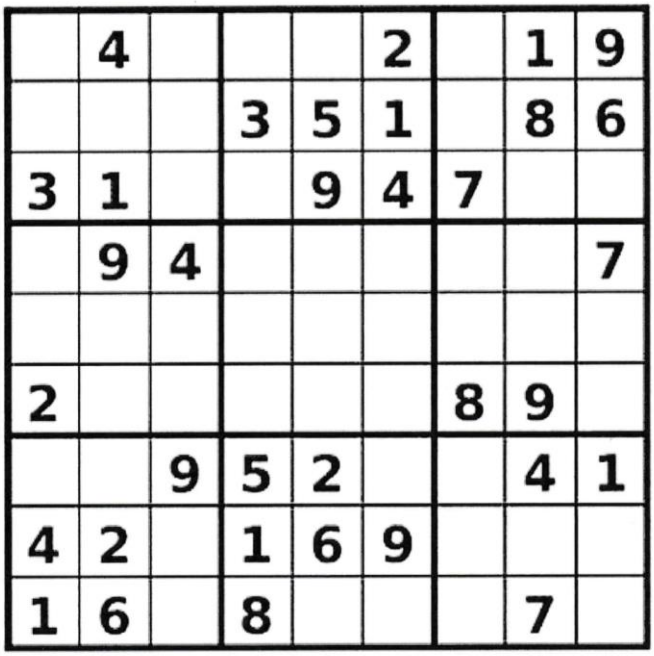
HAS UNLIMITED INTERNET

- OWNS PC AND IS COMPETENT BROWSING INTERNET
- OWNS BASIC PHONE ⇒ NO INTERNET ACCESS ON IT
  ↳ DOESN'T USE ITS I.NET ABILITIES

Other

Appendix J

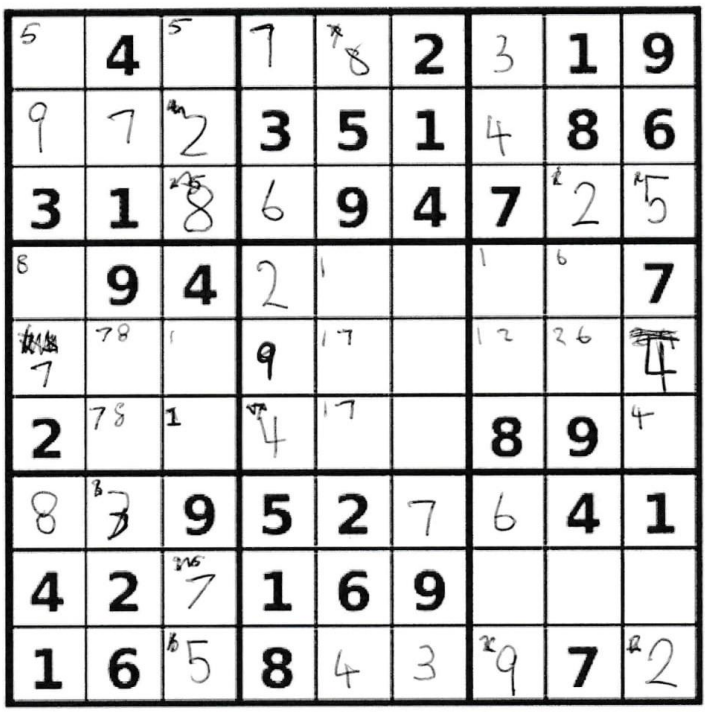
Sudoku puzzle used during the observation section of requirement gathering to uncover any additional wants.



	4				2		1	9
			3	5	1		8	6
3	1			9	4	7		
	9	4						7
2						8	9	
		9	5	2			4	1
4	2		1	6	9			
1	6		8				7	

Appendix K

Partially completed puzzle by one of the stakeholders from the observation section.



Appendix L

Change form that was given to each stakeholder to request, change or delete requirements.

CHANGE FORM

PART 0

Read the following in its entirety understanding: the brief process to complete this activity, your role and responsibilities within the process, deliverables/discussions or otherwise to be provided to the applicable personnel, those involved in the process at any given stage & how the application is reviewed and ultimately accepted or declined.

This is a change form and is used to request a change to an existing/on-going project.

Changes may include, but are not limited to:

New requirements;
Additional features or functionality;
Adjustments to UI;
Improving security measures;
Scaling;
etc

The equiree should fill in part 1
The project manager will review the submitted change in part 2
As part of the change request the doodle-banger will be put through a priority engine to determine whether it is accepted (Weiger's analysis)
Should the change be accepted by the project manager it will be handed over to the applicable individual(s) and an estimated date of delivery/demoing will be established
Please note: delivery dates may vary from initial estimate
In all cases the project manager will inform you of this, along with a casual analysis for the
delay and a revised estimated delivery/demo date

Change freezes will be in-place between:

12th November 2018 – 14th January 2019;
18th March 2019 – 12th April 2019

Note: Changes cannot be requested, implemented or delivered during the above date(s)

CUSTOMER

PART 1

A OUTLINE

PROJECT:

CHANGE NUMBER: To be filled in by Project Manager

B ENQUIRY

i. TYPE OF ENQUIRY (select which best applies): requirement | feature | security scaling | other

if other, please define:

ii. DESCRIPTION OF ENQUIRY:

C SIGN-OFF

CHANGE REQUESTOR:

DATE RAISED: / /

PROJECT MANAGER

PART 2

A OUTLINE

CHANGE OVERSEER:

CHANGE OVERSEEN: / /

B IMPACT

EXPECTED IMPACT:

C PRIORIY ASSESSMENT

ITEM	WEIGHTINGS							
	+VAL	-VAL	VAL %	COST	COST %	RISK	RISK %	PRIORITY
	The priority threshold for / / is:							

D ADDITIONAL INFORMATION

i. ADDITIONAL INFORMATION NEEDED: YES ○ NO ○
if YES, answer part(s) 2Di, 2Dii & 2Diii.

ii. INFORMATION GATHERED VIA: TAKEN PLACE: / /

iii. ADDITIONAL INFORMATION GATHERED:

E DECISION

i. PROCEED TO TAKE ON FURTHER: YES ○ NO ○
if NO, proceed to part 4.

REASON:

ii. NEXT STEPS:

IMPLEMENTOR

PART 3

A OUTLINE

ASSIGNED TO: / /

B RELEASE

SPRINT RELEASE
INCLUDED IN:

DECLARATION

PART 4

A SIGN-OFF

CHANGE CLOSED:

PROJECT MANAGER	/ /
IMPLEMENTOR	/ /
OTHER	/ /

Appendix M

Accepted change form for a new feature.

CUSTOMER

PART 1

A OUTLINE

PROJECT: ONID

CHANGE NUMBER: 01 (To be filled in by Project Manager)

B ENQUIRY

i. TYPE OF ENQUIRY (select which best applies): ~~requirement~~ | feature | ~~security~~ ~~scaling~~ | ~~other~~

if other, please define:

ii. DESCRIPTION OF ENQUIRY: HAVE SOLVED SUDOKU BOARD DISPLAYED IN TAB ONCE CLICKED

C SIGN-OFF

CHANGE REQUESTOR: George T.

DATE RAISED: 02/07/19

PROJECT MANAGER

PART 2

A OUTLINE

CHANGE OVERSEER: MT

CHANGE OVERSEEN: 08/02/19

B IMPACT

EXPECTED IMPACT: Add task to front-end dev.
Convert board to img

C PRIORIY ASSESSMENT

ITEM	WEIGHTINGS							
	1.0	2.0		1.0		2.5		
	+VAL	-VAL	VAL %	COST	COST %	RISK	RISK %	PRIORITY
	6	2	2.0	1	3.2	1	0.69	0.407

The priority threshold for 29/01/19 is: 0.336

D ADDITIONAL INFORMATION

i. ADDITIONAL INFORMATION NEEDED: YES ○ NO ⊗
if YES, answer part(s) 2Di, 2Dii & 2Diii.

ii. INFORMATION GATHERED VIA: TAKEN PLACE: / /

iii. ADDITIONAL INFORMATION GATHERED:

E DECISION

i. PROCEED TO TAKE ON FURTHER: YES ☒ NO ○
if NO, proceed to part 4.

REASON: PRIORITY WAS WITHIN THRESHOLD

ii. NEXT STEPS: Add to issue tracker

IMPLEMENTOR

PART 3

A OUTLINE

ASSIGNED TO: MS 01/04/19

B RELEASE

SPRINT RELEASE INCLUDED IN: 7

DECLARATION

PART 4

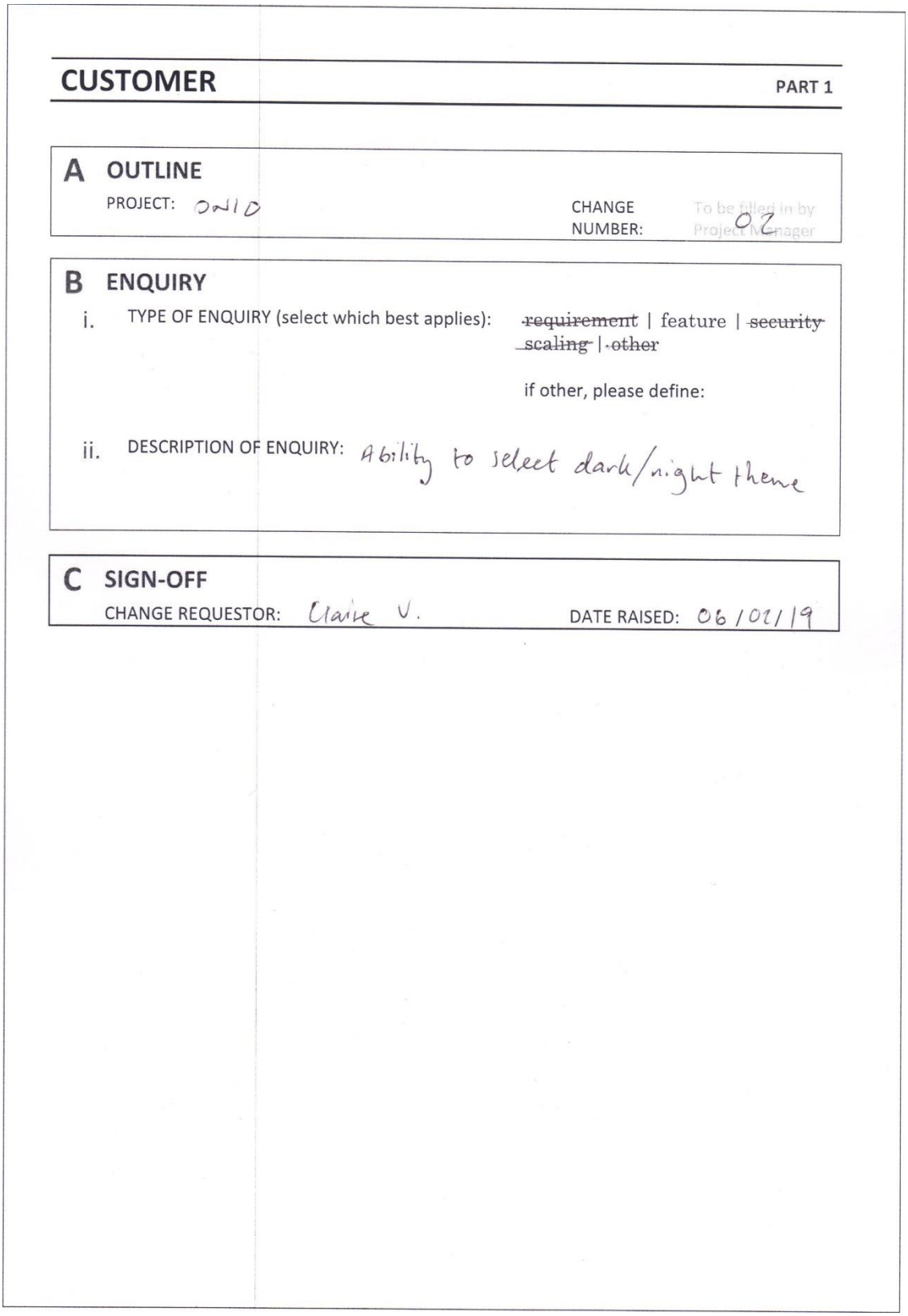
A SIGN-OFF

CHANGE CLOSED:

M Templeton	09/02/19
M Templeton	12/04/19
OTHER	/ /

Appendix N

Rejected change form for a new feature.



CUSTOMER

PART 1

A OUTLINE

PROJECT: OXO

CHANGE NUMBER: 02 (To be filled in by Project Manager)

B ENQUIRY

i. TYPE OF ENQUIRY (select which best applies): ~~requirement~~ | feature | ~~security~~ ~~scaling~~ | ~~other~~

if other, please define:

ii. DESCRIPTION OF ENQUIRY: Ability to select dark/night theme

C SIGN-OFF

CHANGE REQUESTOR: Claire V.

DATE RAISED: 06/02/19

PROJECT MANAGER

PART 2

A OUTLINE

CHANGE OVERSEER: ~~Mr~~

CHANGE OVERSEEN: 08/02/19

B IMPACT

EXPECTED IMPACT: moderate CSS modifications/additions

C PRIORIY ASSESSMENT

ITEM	WEIGHTINGS							
	1.0	2.0		1.0		2.5		
	+VAL	-VAL	VAL %	COST	COST %	RISK	RISK %	PRIORITY
	7	2	7.2	1.0	3.2	3	2.1	0.264

The priority threshold for 29/01/19 is: 0.336

D ADDITIONAL INFORMATION

i. ADDITIONAL INFORMATION NEEDED: YES ○ NO ✓
if YES, answer part(s) 2Di, 2Dii & 2Diii.

ii. INFORMATION GATHERED VIA: TAKEN PLACE: / /

iii. ADDITIONAL INFORMATION GATHERED:

E DECISION

i. PROCEED TO TAKE ON FURTHER: YES ○ NO ○

if NO, proceed to part 4.

REASON:

ii. NEXT STEPS:

DECLARATION

PART 4

A SIGN-OFF

CHANGE CLOSED:

PROJECT MANAGER	M. Templates 09/02/19
IMPLEMENTOR	/ /
OTHER	/ /