Feasibility Study

**Contents**

2 – Learning outcomes

2-3 – Project background

3-5 – Aims / Objectives

5 – Problems

6 – Required resources and research

7 – Project deliverables and conclusion

8 – Schedule and references.

**Learning outcomes:**

My personal goal is to increase my knowledge with mobile application development specifically in programming a logical and intuitive program that could be used in real world scenarios. I will be doing this by:

* Managing my time effectively and completing necessary research to aid in both the development and relevancy of my project.
* Work on developing new skills such as working with things I was previously not that adept to such as 2D arrays.
* Evaluate my own work independently by having a look at similar apps to see what I can do that mainstream similar apps also can do.
* Apply what knowledge I have gained from my previous programming unit (Advanced Programming)
* Potentially cross platform my program, although unnecessary I think it is a good goal I want to set myself to get this app working in both desktop and mobile versions.
* Learn the hardware limitations of the device I will be working on, the current system I am programming on might have more computational power than the device the program is aimed to finish on.

**Project Background:**

**“**Sudoku puzzle was invented by American Howard Garns in 1979. In the year 1984 Maki Kaji of Japan has published in the magazine of his puzzle company ‘Nikoli’ and has given the name Sudoku to the puzzle game, which means “Single Numbers”.” **Job, Dhanya & Paul, Varghese. (2016). Recursive Backtracking for Solving 9\*9 Sudoku Puzzle. Bonfring International Journal of Data Mining. 6. 07-09. 10.9756/BIJDM.8128.**

The project is to create a Sudoku solver application for mobile, this involves creating a program that can solve a sudoku from a template with potentially multiple solving algorithms. Note that the solving algorithm can be incorporated further into a game that can be played by the end user, with the ability to recursively backtrack to an earlier completable state being achievable.

“Backtracking method is probably the most basic sudoku game solving strategy for computer algorithms. This algorithm is a brute-force method which tries different numbers, and if it fails backtracks and tries a different number”

**Job, Dhanya & Paul, Varghese. (2016). Recursive Backtracking for Solving 9\*9 Sudoku Puzzle. Bonfring International Journal of Data Mining. 6. 07-09. 10.9756/BIJDM.8128.**

This project is interesting due to the amount of technicality involved to reach the end goal, there are many solving algorithms the simplest ones might work but not as effectively on harder to solve sudokus. Currently I have been working on an algorithm that recursively backtracks, in way brute forces its way to complete a sudoku. As shown in this research paper it can be achieved by following a basic set of rules and applying them in a loop until the sudoku is completed.

**Aim:**

The aim for my project is to get a fully functioning sudoku game that uses a more advanced solving algorithm where the program remembers its last completable state and allows the user to backtrack to this state manually.

I also potentially want the ability to randomly generate completable sudoku templates, although this seems a bit more technical I think it could be possible via backtracking a randomly generated completed sudoku to the point only a few numbers remain.

I will be aiming to track the progress of my project and log anything that needs to be done or any issues with GitHub, giving me an idea on how to work on a project in an environment that could potentially be open source.

**Objectives:**

**Research objectives:**

* To look at different solving algorithms such as recursive backtracking and dancing link algorithms.
* To understand the rule-based methods and logic that can be used to rule out numbers that are impossible, similar in the way people think.
* To research current real-world applications that apply similar algorithms and what features they implemented that could work from these algorithms.
* Look into how an empty sudoku grid gets generated, although templates are available online it would be beneficial to find out if these are generated by hand or by a program.
* Complete research on existing sudoku software on mobile iOS/Android, understand common features that are shared between sudoku applications.

**Requirements objectives:**

* Select what platform this will be completed on, whether just on mobile or ported from a desktop application.
* Create a questionnaire to find out the following:
  + - 1. End user usability.
      2. End user system requirements
      3. Typical platform type used by end user.
      4. Relevancy of potential additions and whether they would be beneficial for an end user.

**Design objectives:**

The design should be usable by multiple age groups and accessible enough that a non-technical user will be able to operate the program effectively.

This will be accomplished by:

* Creating a good front-end with intuitive features.
* Giving the program a “Memory” by allowing the user to backtrack if required.
* Incorporating multiple features such as different sized sudokus and levels of difficulty.
* A scoring system that can give the end user hints on to how they are doing.
* A warning system that can be toggled that would allow the user to know that the sudoku in its current state is incompletable.

**Implementation and testing:**

Testing will be the most difficult part of this project, I will need to get multiple people to help test the system as others may do unpredictable things that could produce unpredictable results in the program itself. My goal is to eliminate any of these unpredictability’s.

* Use GitHub to keep track of earlier software revisions in case any new bugs or errors come up.
* Collect several people to complete white box and Blackbox testing for the software on different revisions.
* Create multiple platforms for the program, the solving and front end may be separate for the beginning.
* Collect any test results and keep a note of them potentially on GitHub, allowing me to fix any upfront problems.
* Work on cross-platforming my program, although currently designed on desktop the end goal is to incorporate this into mobile.
* Test the product on multiple android phones, see how the front-end may scale differently on different devices and how this can be fixed.

**Evaluation:**

* Create a questionnaire for my testers on the usability and effectiveness of the program.
* Collect opinions of multiple people on the front-end design and usability.
* Ensure the program does what I have set as my goals at the start of the project.

**Problems.**

One of the main issues is making sure the solving algorithm can solve all solvable puzzles of the same difficulty and making sure that the algorithm works efficiently and as fast as It could be, with a simple back tracking algorithm unnecessary resources would be used as progress is deleted and restarted. Without the solving algorithm working properly the program will not function as intended.

Another problem would be ensuring cross compatibility for multiple mobile devices, having more experience with desktop programming this will be a bigger challenge as it is a whole new environment. Although the program may work on one device it may look wrong on another, or it may not even have the hardware requirements to run the program. This is where keeping an efficient solving algorithm would be most beneficial.

**Required Resources:**

**Hardware:**

* Mobile device to test the application on, perhaps multiple of different brands and sizes.
* Desktop PC for initial programming of logic and solving algorithms.
* Compatible cables to transfer the program to a phone for testing.

Note: No extensive or unprocurable hardware should be needed.

**Software:**

* IDE for programming and compiling on desktop.

1. NetBeans IDE.
2. Eclipse.
3. Android Studio.

* GitHub to keep track of and manage version revisions and backups.

**Research:**

Research will be required to find out an efficient solving algorithm that could be used in the scope of my project requirements.

Front end research, developing a usable and efficient front end will require research which could be done by having a look at existing similar applications, especially on mobile, seeing how they scale on different devices and what features they offer could prove beneficial.

* Current sudoku applications.
* Current solving algorithms.
* Pros and Cons of solving algorithms to determine which one would work best for my project.

Current sudoku application research:

* Front end design and features of mobile apps
* Mobile scaling and input methods
* Additional features/difficulty
* Type of help offered to users if any.

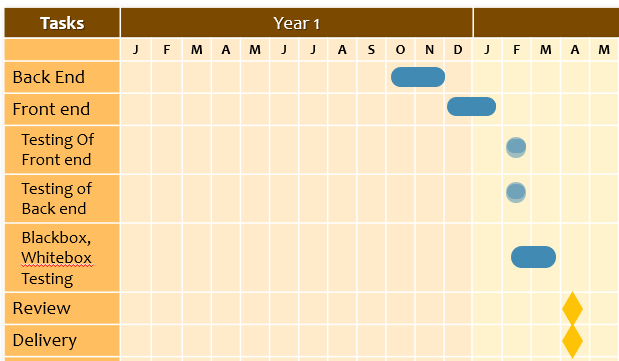
**Project deliverables**

1. A simple working application.
2. Results of testing from other individuals.
3. A fully complete application on mobile.
4. A report on:
   * 1. What objectives I have achieved.
     2. The progress and development of my program throughout the year
     3. Final report.
     4. Anything that I could not achieve that potentially could have been.

**Conclusion**

The project will be interesting as there are many ways to complete it, there’s no set method of achieving the goal and it will be interesting to see how the application can fare to current similar applications. Research will mostly be done into the different types of solving algorithms as sudoku itself is basic in its definition.

**Schedule:**



**References:**

[https://www.researchgate.net/publication/303553939\_Recursive\_Backtracking\_for\_Solving\_99\_Sudoku\_Puzzle [1](https://www.researchgate.net/publication/303553939_Recursive_Backtracking_for_Solving_99_Sudoku_Puzzle%20%5b1)]