**Introduction**

I want to create a project that will generate mazes using different algorithms like recursive backtracker and prim’s algorithm, and can find path in the mazes using A\* algorithm. It’ll also have maze saving functionality that allows mazes to be recreated later on. I will also add a user authentication system to the program that will allow users to create accounts and have their information be securely stored. There will be an admin panel that allows to see all users that have logged in.

**Why I’m doing this**

I was always interested in mazes and how they are created, so doing this for my computer science A level project just seems like a great idea. I think it will be fun to implement all the different algorithms that are required for the program to work, as well as the general code like user authentication and user interface, and trying to tie it all together.

**Technologies & Tools**

I plan on doing my project fully in python. I have realised that I would need a database management system such as SQL, and that would be a headache to implement and have it work together with python, thankfully I don’t have to do that. There is a python library called sqlite that can mimic the way SQL works whilst not requiring to install any extra software.

I’ll also be using the tkinter library for GUI and bcrypt to securely hash users’ passwords, this will improve the security of the program as the passwords themselves are not directly stored in a database but rather their hashes are, and when the user enters a password, it is hashed and compared against the hash stored in the database.

**Problem Recognition**

This won’t be easy, so I have outlined several challenges that I might face when attempting to create this project:

1. I would have to implement all the algorithms myself. I plan on finding ways of how other people did it online, and using the same logic, but I want to write all the code myself to keep it authentic.
2. I will have to learn how to code in python as currently I am a beginner and this would require me to increase my skill level drastically. Thankfully there are lots of resources available online, like the [python documentation](https://docs.python.org/3/) and documentations for individual libraries like [tkinter](https://docs.python.org/3/library/tk.html), [bcrypt](https://www.npmjs.com/package/bcrypt) and [sqlite](https://www.sqlite.org/docs.html) so I won’t be left alone.

**Decomposition**

My project isn’t too big, but it is pretty large and could have hundreds of success criterion and possibly could take me tens of hours to make. To help myself, I’ve tried decomposing the main program into several key points.

1. I will need to implement 2 maze generation algorithms, recursive backtracker and prim’s algorithm
2. I’ll also have to create a user authentication system that accepts user login and securely stores it in the database
3. To manage the program, there will also be an admin console that will be able to see all users that are currently registered and delete users
4. Mazes could also be configured to have different height and width , so perhaps it would be best for me to create a “config page” for the maze before it is created, where all the parameters for the maze are set
5. Mazes would have to be solved, so I’ll also have to implement the A\* algorithm to search the maze
6. I also want my program to be able to save mazes and store all information about the saved mazes, like the maze layout
7. For this, there would also have to be a menu where the user can see all saved mazes and delete them or recreate the mazes
8. To make the application easier to navigate, I will also add go back buttons that will send users to the previous menu
9. Finally, to ensure consistency in my program, I’m going to add limitations on what kind of usernames and passwords people can create when they sign up. For example, they will be able to only use passwords within a certain length, like 3-16 characters. And they will only be able to use alphanumeric characters.

**Divide & Conquer**

#1: The algorithms that will generate the maze, and the algorithm that will path find the maze will not be connected to the user authentication system in any way. First of all, this will decrease the likelihood of bugs and make making changes to the program easier. Secondly, it will make the code easier to read and add changes to it in the future if required.

#2: My project will be progressing in stages, I’m not going to be doing the entire project in one sitting, for example it could go like this:

* At first, I’ll implement the main menu with the basic 3 buttons
* Then I’ll solely focus on making the maze configuration menu
* After that I’ll implement the algorithms that generate the maze
* ... And so on

This could take several weeks or even months to accomplish, as I would be effectively learning while I am coding. So it is important for me to try to comment my code so I can read it in the future and not spend 30 mins being confused.

**Abstraction**

Using abstraction in my project:

1. Separate the front-end and the back-end to allow changes in the GUI without breaking the program
2. The maze itself could be stored as a PNG or something very complex, but it doesn’t have to be. I can just store the maze as a list of lists, for example

**Why is a Program Suitable for Solving This Problem?**

#1: Mazes are pretty much complex puzzles that can be represented using maths as a list of lists. So this makes them ideal to be solved computationally with algorithms because to a computer a maze is just a series of 1s and 0s, walls and paths.

#2: Programs do millions of calculations per second, which makes it significantly easier for them to perform maze-generation algorithm like recursive backtracker and prim’s, compared to a human possibly having to spend several hours doing those algorithms with pen and paper when a computer can do it in under a fraction of a second. Additionally, algorithms always have the same logic. They do not make mistakes (if they are fine-tuned to edge cases), so this will make my mazes always be consistent.

#3: Programs can handle very large amounts of data, for example as I previously said they do millions of calculations per second or even more, so it will be relatively easy for almost all computers to run algorithms such as the recursive backtracker or the A\* algorithm

#4: I can integrate many different features into my program like user authentication, graphics interface, maze creation, maze navigation, saving the mazes, etc. without the program crashing, despite all the different modules of the program being so different from each other they can work together.

#5: Once I develop my code, it will be significantly faster at creating mazes than people, possible millions of times faster, and it will never get tired. I could modify my code and create, export tens of millions of mazes per day and have all of this be done automatically, while a human could probably create 2 or 3 complex mazes using the same algorithm and be tired afterwards.