Evaluation

# My Thoughts

## Login System

I feel that I have implemented a good login system for the scope of my project. The login system as it is provides a flexible way to link data to a specific user via their userID, which I have already used extensively throughout the project. I think the security is strong enough for the complexity of the system – if the system were to be expanded into a wider puzzle system such as my client’s, a more robust password storage system might be advantageous. This could be implemented by adding 2 Factor Authentication via APIs, or using a hash table so password hashes are not aligned with their respective user. I think that my implementation of registering an account is fast and easy, which is appropriate for my solution as it is only operating over a LAN network. However, if it were to be expanded to operate over a website for example, it might be desirable to add more features to registering, such as an email field, using a mail server to send confirmation emails, or a security question for password resets. Finally, the option for the user to delete their account could be implemented easily. The SQL “ON DELETE CASCADE” could be used in database creation commands so when a user’s account is deleted, perhaps with confirmation/security requirements implemented with the methods discussed, the relevant fields in other tables are also deleted, removing redundant data.

## Network Functionality

I think the network implementation with gRPC was the strongest choice for my network. The protocol file system allows new services to be defined without interfering with the current network operation, and the ability to control which services are mapped to the server allows controlled testing of new services. Additionally, I think the speed is suitable for the maximum 150-person load. Considering that most of the time only a fraction will be using the software, and that the services tend to complete quickly (See [TEST]), even if many requests were sent through at once, the server could handle it. The three second deadline on each server call I have implemented allows ample time for even the slowest architecture to support the server. gRPC does support webservers, so adding fully online functionality to the server rather than LAN operation is possible with some changes, but the deadline on server requests might have to be altered to allow for the extra latency caused by the change. However, the REST system is much more commonly used for exposing online services, so it could also be an option to port the system to this instead.

## Algorithms

I believe I have chosen a suitable variety of algorithms for my system. The generation algorithms provide a variety of complexity and looks to fully demonstrate the capabilities of the system: recursive backtrack creates simple mazes with long corridors and sparse branches, growing tree is the opposite, generating many branches of short corridors, and Wilson’s algorithm generates a uniform graph. These algorithms not only provide visual variety, but also technical as well: Their different generation algorithms allows the code to demonstrate how adding a new algorithm to the main generation service can be done in a different sequence of steps depending on the algorithm (See [PAGE]). The solving algorithms are similar in their variety: depth-first search is simple and fast, the maze-routing algorithm is a good middle-ground algorithm, and breadth-first search is slower but finds the best path. This algorithmic variety helps showcase the capabilities of my OOP model. Adding new algorithms is simple, as they can be tested in a separate environment, then added to the server thanks to the abstract class standardizing its output. Adding new algorithms to the server is as easy as adding new cases for them in some switch statements.

## User Interface

I think the UI for the system is ok, but somewhat cluttered. I have made a compact UI with the designer, and by providing hard limits to input fields in control properties, I have eliminated a lot of hard-to-read error handling code. The event hander system provides a readable interface for handling input, which should make adding new features related to interaction with the form easier: I have already implemented some visual features into the form, such as the MazeParameter form greeting the user, or the dynamic server connection label on the same form. However, I don’t think the forms designer was meant to handle a graphical project like this; early on into the project I ran into many issues with subpixel alignment causing the maze to draw incorrectly onto the table, which is why the dynamically sized display form is required. Additionally, double buffering the form, whilst it does stop the table layout panel from flickering on redrawing the maze, does cause some input lag on movement in larger mazes and causes flickering on other components, such as placeholder text in a text box. I think that if I were to approach the project again, I would consider writing it as a Windows Presentation Foundation script to allow for smoother and more robust transitions between forms. Additionally, the XAML code and Model-View-ViewModel structure system would allow for even more organized code and finer control over the properties and rendering of maze graphics. Perhaps even a game engine such as Unity or Godot could be a route to consider for the project, considering the powerful features they bring, in both their libraries and rendering capabilities.

## Robustness of System

I think the overall system is resistant to fatal runtime errors, despite all the potential vectors for them to appear. I have caught multithreading exceptions and used deadlines on all server requests, with specific catch blocks for that status code. This way, I only catch specific errors, so if any other errors arise that are more serious, they can be noticed and fixed faster. Additionally, by adding an error label to each form, it is easy to add more catch blocks to each error handler and display relevant information to the user regarding that error. Since there are several multithreaded components of my system, it does again point towards the use of a different engine for a future version of this system, where multithreading is more directly supported, to eliminate the need for catching multithreading errors.

# Client’s Thoughts

After finishing and testing the system, I provided Greg with it to get his impressions on the project, and what potential features could be added in the future.

## Description of client’s interaction

Greg initially started the client and server and opened the register form via the link label to register a new account. Greg liked the concise account creation process and noted that the initial omission of email confirmation was nice and suitable for his userbase size, as the email could be added later in an account settings page. He then logged in and opened the parameter form. He noted that the initial look of the form was somewhat overwhelming with the number of interactive elements and could be reorganized to better suit his target users. He proceeded to request a 25x25 border-exit maze with Wilson’s algorithm and requested it from the server. He noticed that the parameter form stayed open and suggested that it should hide or have more of its options lock to simplify the visuals for the user. He then proceeded to solve the maze, testing both the on-screen buttons and keyboard controls in the process. Greg liked the presentation of the buttons and the ease of switching between the 2 control methods but mentioned that the label should specifically state the controls are WASD. Once he finished the maze, I directed him to save the maze to test out the load functionality of the system. He first saved a png image of the maze using the local save feature and liked that it used the system file manager rather than a preset path. He then saved the maze to the server and closed the form. He then tried to load the maze using the central buttons on the form. Whilst he thought it was a self-explanatory button layout, he commented that the form should fetch the mazes automatically rather than with a dedicated button. He requested a solve of the loaded maze with the breadth-first solve and was impressed with the responsiveness despite the network system. He mentioned that an additional feature could be to allow for different path colours. He then closed the form and deleted the maze. Finally, he tested out the statistics section of the parameter form. He first requested the local mazes generated and liked the 3D pie chart presentation. He then requested the global best times. He suggested that I could add a button to request the maze a certain leaderboard user solved.

## Evaluation of their comments

I thought Greg’s interaction with the system went well and represented how a user would approach using the system. He suggested many features, which I will discuss below.

### Features Suggested

1. Adding additional account credentials in a settings page.
2. Reorganization of the parameter form.
3. Simplify the visual presentation of the client by only having 1 form visible at a time.
4. Clear labelling of controls.
5. Automatic fetching of mazes from the server.
6. Different path colours.
7. Request a leaderboard user’s maze.

Some of these features are trivial to add but allow for further features to be added in their wake. Some are harder to add or would result in less clean code.

### Discussion of Features.

1. The settings page itself would not be hard to add. It can be another form accessible from the parameter form, which on loading, pulls the user’s data and displays it, with buttons and input fields to allow changing it. The more difficult aspect of this feature is inputting data like emails or changing passwords. Both would require some form of confirmation: the email confirmation wouldn’t be too hard to send, as it could just be done via a mail server, but getting the system to recognise that the user has interacted with the confirmation email is difficult with the current setup. Changing passwords is easier in comparison; it just requires a security question or some similar system at account registration. Alternatively, it could implement an API to access a 3rd party two factor authentication method, such as Authy.
2. Reorganizing the forms is a much easier task. The designer has many organization controls, like panels, split panels, and even the option to obscure entire panels of controls into separate tabs. Rearranging the already present controls into a new format with these would not produce any significant programming challenges, but may introduce some UI design challenges, since the form designer can be difficult with complex forms. Manipulating control margins or docking/anchoring to panels can introduce many issues when done with more dense elements like tables.
3. This is just an extra line of code, telling the parameter form to hide until the display form is closed. This is a good feature to add, as interacting with the parameter form while the display form is open, while limited as much as possible, could introduce unexpected behaviour.
4. This feature at its simplest is just changing the text on the label. However, this feature could be combined with feature 1, implementing customisable keyboard controls in the settings menu and dynamically updating the label to reflect the current control layout. This could be done by having a static class with static KeyCode properties for the 4 directions. These could be updated with the relevant keycodes from the settings menu and passed to the input handler function to check for the relevant keys. Each keycode could be matched with an alphanumeric character in an array so the label text could be changed dynamically.
5. This feature requires moving the request server call to another event handler. I think the best would be a focus related hander. When the form is focused, fetch the mazes into the drop down. This is better than something like the on mouse enter event handler for the drop-down list, as it is more reliable – if the user interacts with the drop-down list while it is fetching the mazes, it could introduce unexpected behaviour.
6. Similarly to feature 4, this would use a property in a static class – this time a Colour, which would be passed to the CellPaint event of the table layout panel where the maze is drawn.
7. This is a difficult requirement. A button array would have to be precisely generated dynamically next to the leaderboard users. When the buttons are pressed, they fetch the maze associated with that user. This means the global stats table would also have to store the mazeID of the top 10 times as a foreign key.