Task 1 - Automatic Snooker Scoring System

https://github.com/DJLad97/Automatic-Snooker-Scoring-System

This is the project that I made for my dissertation at university. This project is a prototype for a digital scoring system for snooker and works by tracking snooker balls, detecting if the balls have been potted and if so, adjusting the score accordingly.
The computer vision system was created in python and open CV and the displaying of scores through a basic react native app.
A laravel backend API was created for communication between the app and the computer vision system.
You can see it in action here: https://www.youtube.com/watch?v=gQ-oATIvN84

One of the most challenging aspects of this project was creating the actual computer vision system itself as computer vision is not something I had any experience in at all before going into this. With this being the case I was quite proud with the end result and especially at some of the custom calculations and systems I had to develop to make it as accurate as possible.

The pot confidence system is one of these features, this had to developed due to balls getting obstructed by external objects. Two main issues had to be addressed: detecting covered balls and identifying balls that reappeared after being hidden or potted. The system initially relied on two variables, the number of detected red balls and the expected red count, to determine potting. However, the approach was refined to consider any change in the number of detected red balls and establish a confidence meter, making it more adaptable to fluctuations in ball detection. For non red balls a different approach had to be taken due to them being potentially added back to the table after being potted. This approach relied on motion to distinguish between balls being potted and those covered by external objects.

Another problem I faced was how to handle ball clumps, to get a count for the number of red balls, the system would count the number of contours (i.e circles) it can see. It finds these contours based on binary mask which is calculated using a colour space configured for just the red balls. Balls that clumped together will form one big contour and, using the perimeter of the balls calculated during an earlier calibration phase, it can calculate how many balls are in the clump (while not fully accurate, it turned out accurate enough to reliably get the number of balls). However, this wouldn't work for balls clumps not in a straight line (e.g a triangular shape) but the confidence system mentioned above can determine false pots (which would be the case here) and when the balls unclump, the confidence system would correctly recalculate.

It's hard to say what I could do differently if I were to come at this project again, given I had a strict timeline for initial development, I could look at every aspect of the vision system and research if there was more accurate methods to achieve desired results since I would have as much time as I want.

Task 3 - Document

In relation to the point: ***"The website contains highly sensitive jobs and has been the target of malicious attempts in the past.***
" There are things you can implement on the backend to help with this:

- Having the site use HTTPS and SSL to help secure communication between the client and server
- Validating submitted data on both the front and backend, and sanatize the inputs as well to remove any potential malicious code from strings
- Enable Content Security Policy (CSP) on the site, doing this mean you specify yourself the site can and can't load (e.g only javascript from google analytics etc and my site)
- Making sure to use CORS to prevent CSRF requests
- Making sure dependencies/libraries are updated to make sure you don't have third party code that has security holes.

Using pinia (state library for vue 3) and persistating the data, I was able to create pseudo database which helped with authentication and applying for jobs.

Since this was quite a small project, a simple structure of components, views, and layouts could be used, if this site was to developed further I would consider the atomic design approach. This is were you have atoms, molecules an organisms, templates and pages (starting from an icon atom, then a icon text molecule, and then a list of icon text components for an organism). It can be tricky at first to try and workout what should live where but, in my opinion, if you're not too strick with yourself (i.e struggling to decide between if a component should be an organism or molecule), I think it can work quite well structuring your project.
Another thing that would help make development smoother is implementing a design system. This will help focus developers when it comes to creating the frontend as it would remove the time developers spent thinking about what colour something should be, what spacing should be used, how big or small the font is. If all this is decided during the design phase, development speed will be increased.

To make sure the site remains as bug free as possible, two types of testing can be done. Component testing and end to end tests (which you would you something like playwright or cypress). With this approach you'll have more component tests than e2e tests, e2e tests will cover critical site functionality where if this area of site was to fail it would almost render your site unusable. Component tests would cover more granular functionality such if I click this button, then this component should appear etc.

To work on this project as team, it would probably be best to following the agile methodology of having sprints, sprint planning and stand ups. While still not fully accurate the sprint planning will help with delivering estimates to the project's client.