My role for this project is the Software Manager. As mentioned in the QA manual role descriptions, I am responsible for monitoring and playing a big part in the coding process, testing the code, and managing the code repository. My role also extends to making sure the user requirements are satisfied and developing user stories based up these requirements.

During the first iteration, I spent a lot of time researching the approach to this project and how to tackle the requirements for the app. I decided it would be best to host the data on a server and the app would request from the server the data it required, which includes recipes files, presentation files, and other media files like video and images. It is a simple python server that uses the Flask framework to serve files through HTTP requests. The server can be hosted from any machine, preferably a Linux machine since the file serving is a lot simpler (and the group ran into multiple issues when running it on Windows machines). I also handled the client to server communication. Using the Android intent system to call the service to connect to the server, the app can request files through a remote file manager that stored all the data from the server. The app would go to the remote file manager first and if the data wasn't there it would go to the server and request the data.

When writing the code to handle the PWS, I found that the structure of the PWS allowed for child elements to inherit from the parent. Properties such as colours and text sizes must inherit from their parent if they are not defined in the child element, therefore the class structure for the presentation had to reflect this. Using recursion in the parsing process, I was able to determine the child elements and easily obtain parent properties that the child could inherit from. Once the parsing has finished, the presentation could be draw and all elements positioned according to the PWS file.

Code contracts were a challenge for us as very few groups were developing in Android Java and were instead developing with JavaFX. Because of this, the libraries that our group received were incompatibility and essentially had to be rewritten for Android Java. I researched ways to get JavaFX to work with Android Java, however with many attempts to find a solution there wasn't an easy or nice way of doing it so I decided it wasn't worth the time. On the flip side, I managed to create the code for the contracted video player.

For the most part, I was able to fulfill these responsibilities during the development of the app. However, it seemed at some points that I acted more as a lead software developer rather than an actual manager. And this is a potential reason why we didn't meet as many user stories as expected. I found it difficult to balance the coding tasks I had to do with the managing of tasks and, for example, organising programming sessions to help others. As such, there were points during development that had a lack of focus and direction, like team members getting mixed up with tasks or not knowing what to do because of miscommunication and guidance. Additionally, the team were completely new to Android Java and the development tools that accompany it, so some of the development time was used to become familiar with the new environment.

Overall, the team adapted well to the new environment and we managed to fulfill the mandatory user stories and some of the optional and desirable requirements for the app (which I am very pleased with).

List of contribution to deliverables

- First iteration - The majority of my contribution in this project was in the code for the first iteration and final iteration of the app. Contents of the code include:
    - HTTP server for file storage.
    - Data handling between the client and server.
    - Parsing the data into a Java object so the app can handle it (recipes, videos, presentations, images etc.).
    - The presentation viewer; interpreting the data parsed from the PWS to create a presentation, and handling screen taps for slide transitions.
    - General help and input for most coding tasks such as the class system, search and filter functions, and favoriting recipes.
    - Testing, debugging, and merging the code together from different branches and managing the GitHub repository.
    - Contract code for the other groups: https://github.com/JackoRo/SWEngG3VideoPlayer

- Final iteration - The tasks for this iteration were to finish the remaining user stories and try to get some of the optional/desirable tasks complete. Since the core tasks were completed in the first iteration, the final iteration tasks were more straightforward to complete and were completed quicker than the first iteration task. Contents of the code include:
    - Manual refreshing the server with a ‘pull to refresh’ on the app home screen.
    - File restructuring of the server and recoding it to handle this.
    - Handling adverts in the presentation file so they draw to the slide if ‘advert=1’ in the pws file of the slide elements.
    - Merging in and altering the contract code to work for Android so the presentation can handle shapes, images, videos etc.
    - Streaming the instructional videos on the server.
    - Merging all the code together for the release version of the app, testing it satisfies the user requirements, and debugging and refactoring code such as the shopping list checkbox handling code.

- Final test and integration plans - The test cases for each user story and feature.

- User Manual - Wrote the FAQ and provided the screenshots for the document. Helped with the descriptions of installation and some of the features.

- Financial reports - Filling in timesheets and giving some help with the filling in of hours in the analysis of timesheets.