Critical Evaluation

The program does not satisfy the constraints within the requirements document. The program can show a teachers timetable, and display a class list, displaying the students of a given class and the teacher of the class. The ability for a teacher to view tasks, add task marks for a student, and add behaviour notes for a student was not satisfied. The program does not have this functionality on the client side.

For the design objectives, the program does not satisfy all the requirements. The program does not have all of its functionality available on the client side, and therefore is not easy for an end user to use. The program is not fast because the way the timetable requests are performed, the server needs to query database multiple times, the operation takes a while and slows down the application. The application is modular and secure.

The program does not satisfy the functional and non-functional requirements set forth. The program does contain multiple user types, but only teachers and admins can do anything of meaning, and the teachers are the only users that can do anything of meaning on the server side. The idea to create separate users each with their own unique functionality was not a good idea because it resulted in taking development time away from the functionality that was important to the application. The graphql api was also not a good idea, rather rest could have been a better solution because it is more mature and there are more resources about making a rest api. The ability to have multiple dashboards and multiple users that redirect to those dashboards was unnecessary and had nothing to do with the clients needs. Although the technology and libraries used did not pose a problem in development, the addition of so many unnecessary requirements for the application caused issues for development.

The application still does have the ability to implement certain functions in a future release on the server side. The frontend would have to be rewritten as it does not contain multiple dashboards. The implementation of the frontend was not done well because the libraries used (react-Apollo-hooks) have the ability to have a global state but was never used. The state is only stored in individual components, which meant that it was hard to share information between screens.

The programming language choice in go was alright, it accomplished the goal set forth. The programming language choice in javascript was not quite alright mostly due to it having to be compiled by a program named babel, which I did not know how to use. The decision to switch to typescript came about mostly because the typescript compiler was easier for me to configure.

The plan and gannt chart was not suitable due to possible problems faced not being thought of at the time. For example in the gannt chart, the frontend and the backend are developed separately, where they should have been developed concurrently. More time should have been allocated into the production of the application. I did not follow the gannt chart and resulted in me falling behind in implementing certain features.

In summary, the program does not satisfy the requirements mostly due to the over complication of the project and the lack of time management. With better planning ahead of time, and more focus on the tasks that actually needed accomplishing I could have finished easily, but the diversion from the users needs and the addition of unnecessary components led to the eventual unfinished product.