Software Evaluation

## Efficiency & Effectiveness

Communication with the client was positive and effective and a suitable software solution was produced to meet the client’s needs. The client reacted positively to the software solution in the video (usability test 2). The software solution was deemed to have met all functional and non-functional requirements for the client. The software solution was effective in performing all the tasks required of it during the software demonstration. The code for the software is efficient as it doesn’t use any algorithms of high time complexity and SQL database queries are limited to an in-process SQLite database rather than a separate database server which reduces complexity. The design of the software is simple and efficient for the end user by presenting students with a single page and clear instructions for filling out the new event form. The software is efficient in meeting solution requirements as it satisfies software constraints.

## Limitations and future improvements

* The username, password and teacher name are hardcoded into the software with example values such as ‘Example Teacher’. This means that modifying that information requires editing the source code. This process is not user friendly and could be improved by loading user credentials from a file or from the SQL database which would be accessed with a change username and password page in the software.
* The calendar and lunchtime calendar files need to be updated manually ever year to reflect the calendar of the new school year. The user experience for this could be improved by automatically downloading the files from the internet when they update every year.
* The data model could be changed from a global approval, single user system to an independent approval, multiuser system which would make the approval process for the users of the software more streamlined. However, there are technical hurdles to accomplish this.
* The software needs to be accessible to other students in Suzanne Cory High School to submit events and teachers to approve events, but currently, the software only runs locally on a laptop and is not accessible on the school intranet or the internet. This limitation could be addressed by installing the software on the school intranet, so it is accessible to students and teachers.

## Changes

* The new event page was changed to have just a text reference to the SCHS cash handling guide rather than a link to it since no online resource for the guide was available.
* The Edit Names page was changed to a more versatile and easier to implement complete event editing page which would allow teachers to edit all details of an event rather than just the names (i.e., if adjustments need to be made depending on the circumstances). This better satisfies the user needs and goals compared to the original design.
* The visual design was made more consistent with the red borders maintained between all pages of the software. Additionally, the SCHS was maintained the same between pages rather than adding additional text to specify the page. The information is instead specified in a page header. This makes the software more consistent and easier to use compared to the original design.
* The original design of an independent approval, multiuser system proved too complex technically for the data model of the software. The design was modified to a single user, global approval state system which still satisfies the solution requirements of allowing a teacher to validate student submitted events. Coordination between different teachers can be done outside of the software, i.e. over email. The software is also compatible with a highly limited multiuser setup (by adding new credentials and names to the dictionary in the source code), but the approval state of an event is global across all teachers. That would mean that one user approving an event would set the event as approved by all users, which is not a useful behavior in most circumstances.
* Software doesn’t add the approved event to a calendar since connecting with existing calendar software is too technically complex and creating a custom calendar GUI for the software is outside the scope of the project.
* The software does not modify the school calendar docx with newly approved events due to technical limitations since modification of the file while it’s being used to perform other tasks such a data validation would decrease the reliability of the system and could lead to data corruption or loss.

## Modification from usability test

Optional form fields in the new event page of the software were modified from including “(if applicable)” to “(leave blank if applicable)” since during usability testing with a student, it was identified that there would be confusion whether to write “N/A” in optional fields or leave them blank. This change conveys the correct input to the user more accurately.

## Security

Pages which perform privileged operations in the software such as the manager page are password protected and will redirect the user away if they do not have the appropriate login credentials. Tokens stored in the browser as cookies are encrypted with a secure key. These prevent unauthorized access, modification, or deletion of the data.

## Effectiveness of the development model

While the waterfall development model was effective during the analysis and design phases of the problem-solving technology, the sequential and strictly linear nature of the development model was ineffective during the development phase and evaluation phase of the problem-solving methodology. This is because it was often found more productive to complete steps such as development and testing in parallel rather than sequentially as had been assumed in my implementation of the development model in the Gantt chart. Additionally, it was necessary at points to go back to prior steps such as analysis and design to add new features to the software or make changes that were not in the original design due to technical limitations. As such, there were points in the development of the software where the development model was not strictly adhered to. The development model of the software could be improved by switching to the spiral development model which continuously goes back to prior steps in a spiral to update the software which would have solved the linearity issue of the waterfall model.

