**Functional Requirements**

|  |  |
| --- | --- |
| Display hours | This was not fully completed, the calculations and list view that was made to display it was completed. But the formatting for the hours done was not made in time meaning that the hours that is displayed looks as if there is an extra two 0s. |
| Store Staff data | This was fully completed; the staff data is all stored in one text file. In the text file 14 lines are stored for each member. Last name, first name, ID, password, and the time in and out from Monday to Friday. |
| Edit/Remove data | A separate form was made to complete this requirement, this was done to reduce clutter in the Admin form, so it was more efficient to do this specific process. A process that was inputted in the code to make the overall textile to read was a ‘IF’ statement that ensured that the total amount of lines was divisible by 14. This reduced the amount of potential errors that could occur in the programme when the file was read and loaded into the list views. Overall increasing the efficiency. |
| Add data | A separate form was made to complete this requirement, this was done to reduce clutter in the Admin form, so it was more efficient to do this specific process. Only one step of validation was created in this form as it is hard for users to create errors. If users were to forget to input in the text box a blank line is created instead. But to make sure that the non-functional requirement of the text file only stores up to 30 members the text boxes are all disabled, and the add button, at 420 lines in the text box. |

**Non-Functional Requirements**

|  |  |
| --- | --- |
| User friendly | This cannot be fully evaluated; the login screen was not complete but was still made for the intent that users can record their time in and out. The Login was designed simply so users could log in as only 1 option is given to the user. If the program was complete and the staff was able to go to the staff form. In the staff form only 2 functions are shown, to be able to go back to the login screen without any edits and to be able to record hours. Because of the limited options this is easy to navigate for users making it as user friendly as possible. The Admin form was not made so user friendly as the admin of the program should have a good understanding of navigating programs. |
| Maintainability | This was achieved by adding two different forms adding more functionality and maintainability to the text file of the program. A Edit/Remove and Add form was made so the Admin could easily access the contents of the file without having to go into source code. |
| Store to 30 members | This was easily achieved by capping to number of users the admin can create in the add form, as this is the only place new users can be created efficiently. |

**Development Model**

The development model that was used for the project was the water fall model. This model was used for several reasons that helped assist me throughout the project. This model divided the project into four parts. These four phases are also very similar to the progression used in the VCE Applied Computing: Software Development criteria plan, both follow four main phases - Analysis, Design, Implementation (Development), Testing (Development) and Maintenance (Evaluation). This made it so all criteria of the project could be done in the same order as they were due, and I did not have to repeat any stage over. And because of this in this model documentation is well done and thoroughly as this is required due to there being no redoes of any section of each phase. This is great to have all documentation done well as unexpected expenses are less likely to appear in the final product. Testing is also well done and simple due to the progression of the model as it is done in an order and before moving on to the next part of each phase the first must be done forcing the testing of the project to be done properly the first time. So because of the linear progression of this model there is no repeat of any stage, this is help full as it reduces and confusion for when the project is done.

**Critical and Creative thinking**

Designs:

In the designs a large range of creative thinking is shown, in each design multiply drafts were made to suit the functional and non-functional requirements that was requested. In the admin form 3 different designs were made to create a functional screen that is easy to understand. In the first draft a list view and 3 buttons are used for each function needed. But because of the amount of space that the buttons would have to take up they would look out of place. So, in the other 2 drafts a search function was added to take up some of the extra space making the button smaller. The addition of the search bar also filled the non-functional requirement of being user-friendly as is allows the admin to quickly search through the list without having to scroll by hand and pick users out by eye.

Coding:

There was little creative thinking that actually made it into the final project of the project, when trying to create the login form I had to be able to get the program to read and remember the ID and password of each user and match it up to the textboxes. First a list view was used but I was unable to get it to remember a staff member across to the next form. I then spent a while learning to use arrays to get the program to remember the staff members, I created two attempts of the arrays but unable to learn how to work it. But because of the time I had this had to be scraped, these attempts can be seen in the comments of the code in the login screen at the bottom.