1. **Methods, Variables, classes & Parameters**

* Produce a table that will show the methods that will make up your solution. Remember, each method is a function and should complete a single task only e.g. add two numbers together; or write to a file etc.
* Within your table, you should have identified any variables to be used within your program. Remember, these will be local variables only. You should also incorporate any parameters (i.e. inputs that will be needed to run the method.
* Have a column that tells the user the class that each method belongs to.
* Finally, include a description column that describe the purpose of that method i.e. what task is it to perform when it is run?

1. **Algorithms – 2/3 parts**

* Produce a set of flowcharts for each of the functions you have identified during the decomposition (within your analysis) stage.
* Underneath each of your flowcharts, you need to justify that method/ function. Why is it needed? How does it work?
* Mention any validation that may exist within that function i.e. how will you ensure that the user doesn’t produce an error that breaks the program.
* You may wish also to incorporate pseudocode alongside your solution to present each method differently.

1. **Usability – 6 parts**

* Learnability: how easy it will be to for users to accomplish tasks the first time they use the solution.
* Efficiency: once learners are familiar with your solution and its design features, how quickly will they be able to perform their intended tasks?
* Memorability: how easy will users be able to re-establish proficiency, when returning to the solution after a period of absence?
* Errors: how often are users likely to make errors when using your solution? For example, if using a command line, how easy will it be for users to remember specific instructions; or when using a graphical user interface, how easy will it be to click the wrong link or input the wrong data?
* Satisfaction: how pleasant is the design of your solution for users? You may want to carry out some sort of survey as evidence.
* Produce a series of design drawings to show what your solution will look like. These can either be hand drawn or electronic.

1. **Validation**

* Explain what features or code you will include to prevent the users from making any errors that will render your program unusable.
* Consider methods of correcting the user once they do make a mistake. Will you use some sort of error message telling the user how to correct their error or what steps to take next?

1. **Testing during Iterative Development**

* What data will you use to ensure that you tests are carried out properly? Are there specific inputs that need to be given to the system during iterative development?
* Talk about the different types of test you will use when testing the final implementation of your product with justification i.e. white box testing (carried out by a knowledgeable expert of the system) black box testing (carried out by someone with little knowledge of the solution, who looks predominantly at the user interface), or destructive testing (used to try and break the system when in full use).

1. **List any other data essential to the post development phase.**