Question 1: Reflection on the design methodology and lifecycle

State what methodology you're going to use for your project. The methodology has to support your alternative use cases too.

Validating and justifying by comparing to other methodologies (Use maybe 2 or 3 other ones to justify why you're not using them and why you chose a particular methodology and how it's used). How the risks and mitigations are addressed in your methodology has to be explained too.

Evaluate the methodology for each lifecycle throughout the project and for each scenario/use case, explain the adaptability of the methodology.

Question 2: Class diagram and its justification

Justifying can be either explanation or sequence diagram.

However, if you used sequence diagrams as justification, you would still need to give an explanation as you can't just leave the diagrams standing alone. Therefore, sequence diagrams would only be used as further validation and there are no extra marks provided for that. In conclusion, sequence diagrams are not mandatory and if you feel you want to include sequence diagrams to explain better in some areas, you can do so. Since sequence diagrams are not compulsory, you can even just go for explanation alone.

<u>For the first 10 marks:</u>

1. To validate your classes, you should relate them to the functional requirements and explain why you used them.

2. Justify the relationships/associations between the classes (if you used real world examples for the explanation, it would be ideal).

3. Validate your main classes with the main user requirements and explain why you did so/used them.

<u>For the next 10 marks:</u>

Justify the attributes and explain the multiplicities used (If you used OOP concepts like abstraction/inheritance, you can justify their uses too).

Explain how code reusability, readability and flexibility are adapted too.

For example: If you used interfaces/managing classes, they can help for code reusability and being able to make changes in one area to change everywhere else can count for flexibility.

For the next 5 marks:

Architecture testing – refer the architecture testing slide for this.

Question 3: Testing of your class design – class structure

Research what all the testing methodologies are, choose what's relevant to your project and explain.

- For example, you can test the hierarchy levels and if you find you have more than 3, it's not a good class diagram.

You can use test classes (to show how relevant your output is to the actual output)

- You can use the template from Week 10's lecture for this (You can make minor changes to it according to your project but the template has to still remain nearly the same).

You can also use previous question's testing between class diagram and functional requirements to explain too.

You don't have to do syntax error testing.

This question is only for 2 main classes from your class diagram but the testing methods should be relevant to your entire class diagram.

There is 6 marks allocated for explanation of the modifications from CW1 (in the use cases or the functional requirements). Also you can even state all the requirements you are cutting down from cw1 with an explanation.

Question 4: System Architecture and Question 5: Testing of your design - Architecture

For all justifications and testing, it's the same pattern. You include how relevant the architecture you chose is to your project and why you did not choose other systems and so on.

Question 6: Considerations on the Software Engineering Principles in your design

There is a lecture on the software engineering principles, you can use that for the first part.

For the second part: Stating if you can use external libraries/frameworks in your project and explaining why they're being used. You can also include process managements and memory managements and how they are handled.

Random points:

So, you will have 1 complete class diagram and 1 complete architecture diagram.

Minimum 5 use cases need to be selected for class diagram.

You can go for any system architecture. Extra note: If you went for MVC: for each use case, you'll get 4 classes in the architecture. So, going by 5 use cases minimum, you will have minimum 20 classes. (If you didn't get what I said here, forget it, don't come for me)

Coming up with a creative test case for your own project and being able to justify them will give you marks.