Essay 1: Explain the difference between HTML, CSS and JS.

**HTML:** Stands for HyperText Markup Language, and is used as standardized language for web browsers and creating website. HTML deals with the structure of a website which means marking up text of code to give the browser instructions about how to render a web page. HTML is essentially a language that the web browser understands.

**CSS:** Stands for Cascading Style Sheets, and is the language that includes the presentation and styling of a website. CSS is the visual part of the web page building process for example the coloring, use of fonts, and overall look.

**Javascript:** Contributes with the function of a website such as

Essay 2: Separation of Concerns in relationship to HTML, CSS and JS?

Imagine you have a program which has been released. A customer comes along and offers to pay you for a enhancement to one of its features. In order to get the money, you will need to change your program to add the new feature. Some of the things that will influence what your profit margin is are:

1. how much code you have to change
2. how easy it is to make the changes
3. how likely you are to break existing features that are being used by other customers
4. how much you can reuse you existing model/architecture

Separation of concerns helps you to get more positive answers to these questions.

1. if all of the code for a particular behaviour of the application is separated out, then you will only have to change code directly associated with your new feature. Which should be less code to change.
2. if the behaviours you are interested in are neatly separated from the rest of the application it is more likely you will be able to swap in a new implementation without having to fully understand or manipulate the rest of the program. It should also be easier to find out which code you need to change.
3. Code that you do not have to change is less likely to break than code that you do change. So splitting up the concerns helps you to avoid breakage in unrelated features by preventing you from having to change code that they could call. If your features are mixed up together you might change the behavior of one by accident while trying to change another one.
4. If your architecture is agnostic to technical or business logic detail then changes to implementation are less likely to require new architectural features. For example, if your main domain logic is database agnostic then supporting a new database should be as easy as swapping in a new implementation of the persistence layer.