Question 1.
The StudentPortalHelper has low cohesion because due to having too many processes within the class.

a) A class should have high cohesion. The reason it should have high cohesion is that it will make sure code is simple to test and run. It will also make it easier to decide when to make another class when there are too many different tasks in one class.
b) Since StudentPortalHelper has low cohesion I would focus on making sure that each part is split into its own class. So GPA calculation, CSV export to disk, Email formatting, Date formatting, Payment processing, Password Strength check, and Ad-hoc caching would become individual classes that will be coded separately. This way if there are errors in the code it can be easier to identify where they came from and fix. It will also improve testability if new code is implemented for how to fix it.

Question 3

a) The current structure does not support this because the software does not support changes in the trim level. To change the trim level they would have to start over which would get rid of all the information prior to that point because of how it is structured
b) To allow for trim level to change dynamically we would change them from subclasses to attributes to the car. That way if the customer changes trim level it can be changed without getting rid of prior data.

Question 4
Device is an abstract class because it is the start of a device but can't be a total device on its own.

Question 5

1) I use AI to support my learning by having it explain to me any code I take from it in digestible chunks. If it can't explain it to me or I can't understand it I don't accept it. That way I minimize errors while understanding where I can change and modify code as needed
2) A benefit would be that work can be completed significantly faster but a limitation is that long term memory suffers from not going through the longer process of typing things out and reviewing errors making it so I dont remember syntax as reflexively.
3) I expect AI will change how I solve problems by changing how I code. I believe that as time goes on coding languages will matter less while understanding will matter more because I will be able to get a skeleton of what previous people have done for a project while I will have to change the specifics of my project to make sure it runs as intended.