Explain why you have applied a particular design pattern for your software architecture

We have used factory design pattern in our architecture and are not going to use singleton design pattern and prototype design pattern.

The factory design pattern is used because, in our architecture, we will create View, Command and Strategy many times. Using the factory design pattern can reduce the coupling between "user" classes and them. At the same time, doing so can perfectly comply with the single responsibility principle and the opening and closing principle. We can put the creation of specific classes in a separate factory, making the creation of classes easier to maintain. At the same time, we can complete the replacement of the creation without changing the existing framework. For example, for our current BoardView, it is responsible for rendering the chessboard to the Terminal. If we implement a new implementation class, its function is to render the chessboard into the GUI. Then we only need to modify the implementation class generated in the factory. The rest of the frame will not be affected in any way.

The reason for not using the singleton design pattern is that the singleton design pattern violates the single responsibility principle. And this will make them face the problems of multiple reuse and multiple class influences. This is very likely to cause problems with the running security and stability of the program.

The reason for not using the prototype design pattern is that our class does not involve the cloning function. Not only that, but we have many classes that need to be able to be compared. For example, between different players, even if they have the same name and the same data, they should be different players. Because they have different uids. We should not allow clones to appear on the basis of existing frameworks, which will lead to instability during the operation of the framework.