The abstract factory pattern is a creational design pattern used in software development to address the following primary objectives.
It encourages encapsulation by abstracting the process of object generation. Clients using the abstract factory interface are not required to be aware of the specific classes being instantiated. This helps to disguise the intricacies of object generation and encourages a separation of concerns. It is useful when you need to confirm that the objects you build are compatible and belong to the same family. For example, you may need to develop a number of interconnected items, such as GUI components in a specific look-and-feel (e.g., Windows or macOS). The abstract factory guarantees that the goods produced by a factory are uniform and work well together.Abstract factories can be used to switch between multiple sets of related classes (object families) without having to change the client code. This is especially useful when you need to support multiple platforms, databases, or external services.
You may encounter issues, if you are not using the abstract factory.
The first one could be increased coupling, which means that without the abstract factor,it could be difficult to replace or extend classes because changes in one class can affect other parts of the application. You will also face reduced flexibility which means that, if you directly instantiate concrete classes it can be challenging to switch between different implementations or objects. You might need to modify a lot of parts in your code base to accomplish this. The Open-Closed Principle states that software entities (classes, modules, functions, etc.) should be open for extension but closed for modification. When you directly create instances of concrete classes, you frequently need to modify existing code when introducing new implementations, which violates this principle. Abstract Factory contributes to the Open-Closed Principle.

In conclusion, the Abstract Factory pattern improves maintainability, flexibility, and code organization by allowing the creation of families of related objects with a consistent interface. It can result in more modular and adaptable software when used correctly. Its necessity, however, is determined by the specific requirements of your project. If you only need to create simple objects and don't anticipate variations or the need for different object families, the Abstract Factory pattern may not be necessary.