Singleton Pattern

The Singleton Pattern ensures that a class only gets one instance and it provides a global access point to that instance.
The application will have only one instance which is very beneficial for the system because this will save the memory and when the memory is saved the application will run faster. Another benefit of the singleton pattern is that it manages resources efficiently, if we would not have been using the singleton pattern then the system would create multiple instances of such resources which would lead to resource exhaustion and it could have serious performance issues. Creating objects/Instances can be relatively expensive in terms of memory and CPU usage. A Singleton minimizes this overhead by reusing the same instance throughout the application. While these benefits alone are compelling reasons to embrace the Singleton Pattern, there are additional advantages as well. However, should you choose not to utilize it, you may encounter several challenging issues like Resource Wastage, where you may create unnecessary instances of resource-consuming objects, leading to increased memory usage and potential performance problems. Also to mention that there can be Concurrency issues, where when you dont use Singleton pattern in Multi-threaded applications, that can result in race conditions(This is when the outcome of the program is unpredictable because it depends on the relative timings of the events.) and data corruption.

In summary, the Singleton Pattern is a powerful tool for optimizing resource usage, promoting efficient resource management, and reducing object creation overhead. Its implementation guards against issues such as resource wastage and concurrency-related problems, ensuring the robustness and stability of your software.