Demonstrate an understanding how middleware via DI simplifies your code:
To understand how middleware via DI simplifies your code, first we must understand what is DI. DI is a design pattern, which is used by many middleware libraries such as Java spring or C# .net. It allows for loosely coupled programs, allowing for clear modular code.

When a class requires a dependency, the middleware "injects" the dependency for you. There is no need to initiate a new object. It will be pushed in by the middleware. So for example for our backend, the model layer has a dependency on the inbuilt memory database. Rather than initiating the dependency, we simply ask the middleware, hey I need this dependency. The middleware will look for it and give it to you, should it exist.

How does this simplify code? When the code gets larger, there would be more dependencies on each other. Dependency hell occurs, where everything is very tightly coupled and prone to breaking and bugs. Dependency injection reduces all the dependencies on each other, by allowing it to be modular. Breaking the code into pieces, makes it easier to manage and thus simplifies the code.

Demonstrate an understanding of why middleware libraries made your code easier to test:
Middleware libraries make your code easier to test, by breaking functionalities into separate libraries/layers that are independent from one another. This means each library is independent, has one purpose and that one purpose can be easily tested. There is a common principle called the single responsibility principle. This is an example of it.

For example, lets look at the Spring Java framework. https://spring.io/projects
There are many libraries, each with its own purpose. EG Spring Oauth, Spring Data, Spring GraphQL…etc. Notice that each library has one purpose. Middleware library's make your code easy to test because it splits off a complex system into subsystems.

In addition, middleware libraries reduce the boilerplate code needed to get the functionality working. This makes it easier to test, as many of the hard work is done for you by the middleware library and is well tested. This means you can focus only on your code and implementation, reducing the amount of code you need to test.