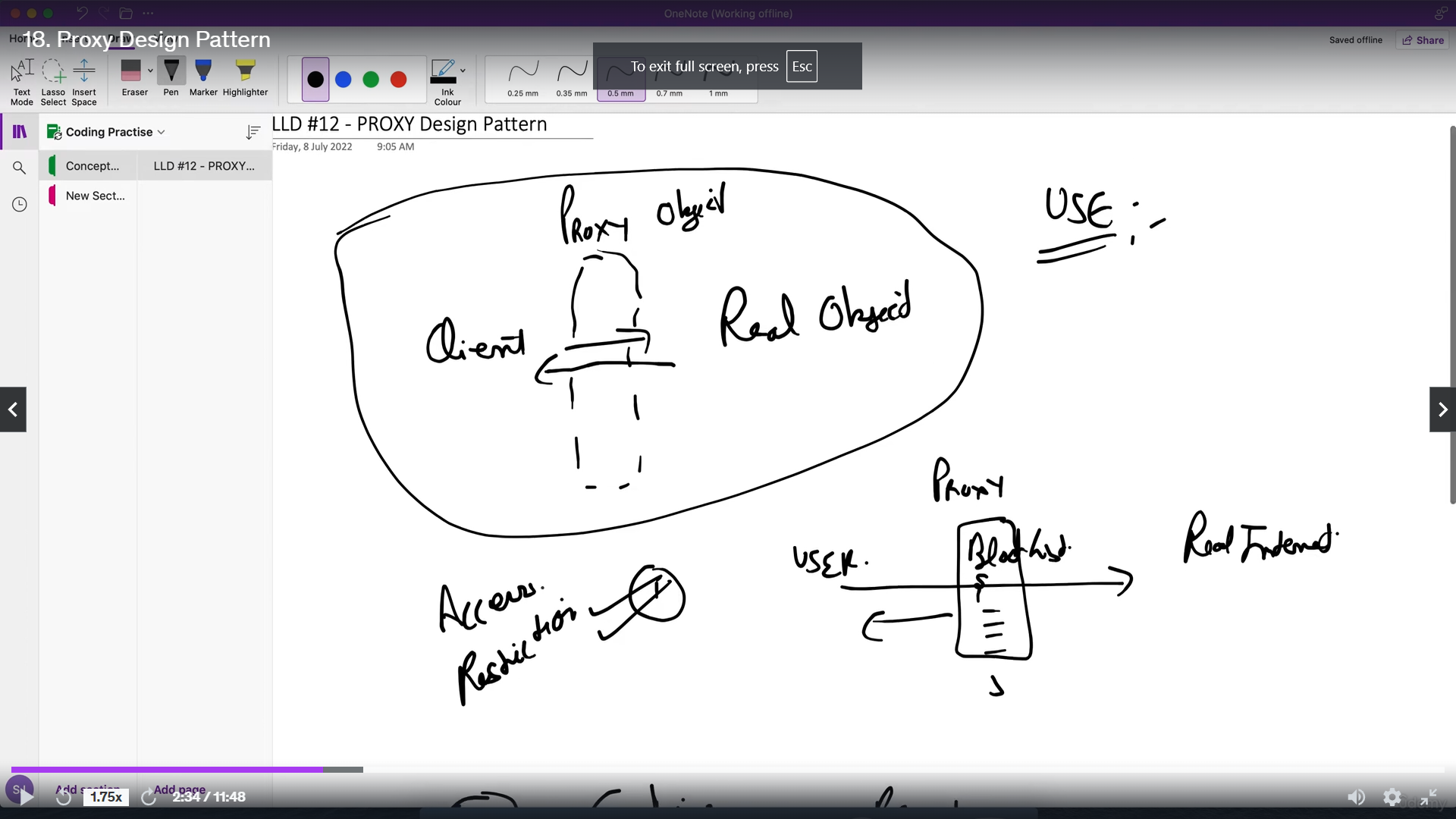
**Proxy Design Pattern**

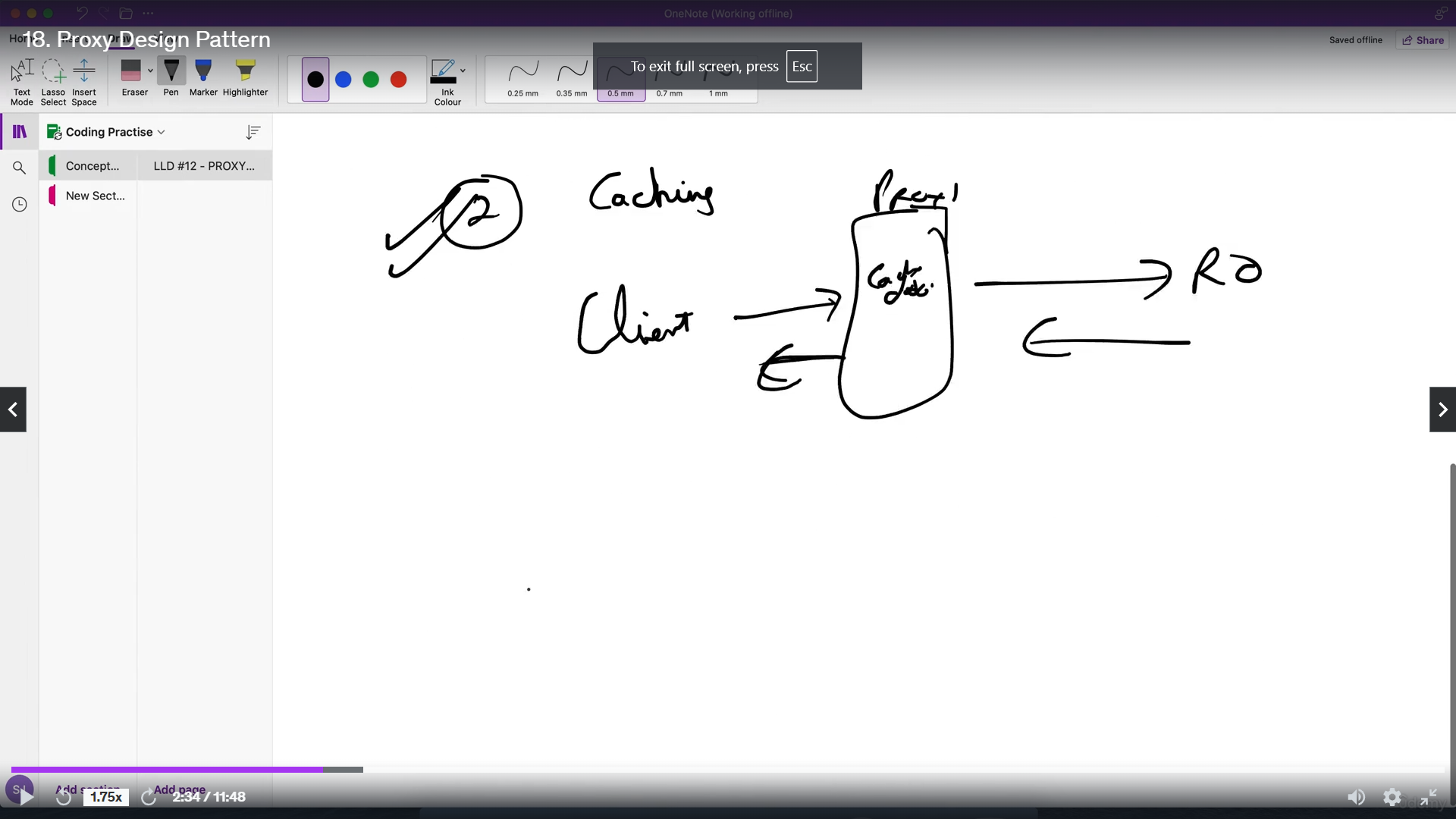
The Proxy design pattern is a structural pattern that provides a surrogate or placeholder for another object to control access to it. This pattern helps manage the complexity, security, and performance of accessing an object by interposing a proxy class that represents the real object.

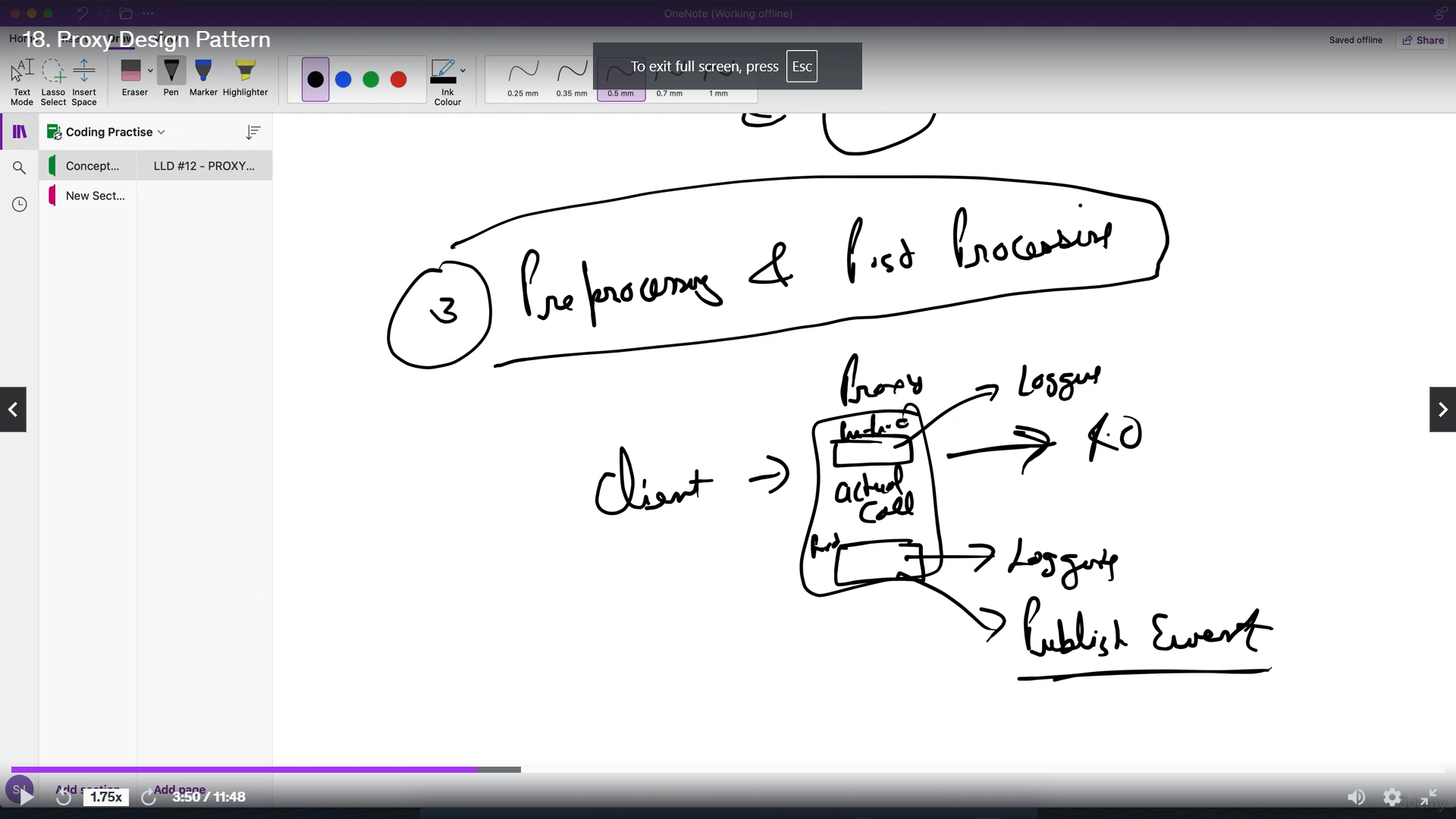
**Definition**

The Proxy pattern allows you to provide a substitute or placeholder for another object. This proxy object controls access to the original object, allowing you to perform operations such as lazy initialization, access control, logging, or caching.

**When to use**: 1) Adding internet restriction bw user and website. 2) Add caching in proxy layer so the object is not accessed every time. 3) PreProcessing and postprocessing.

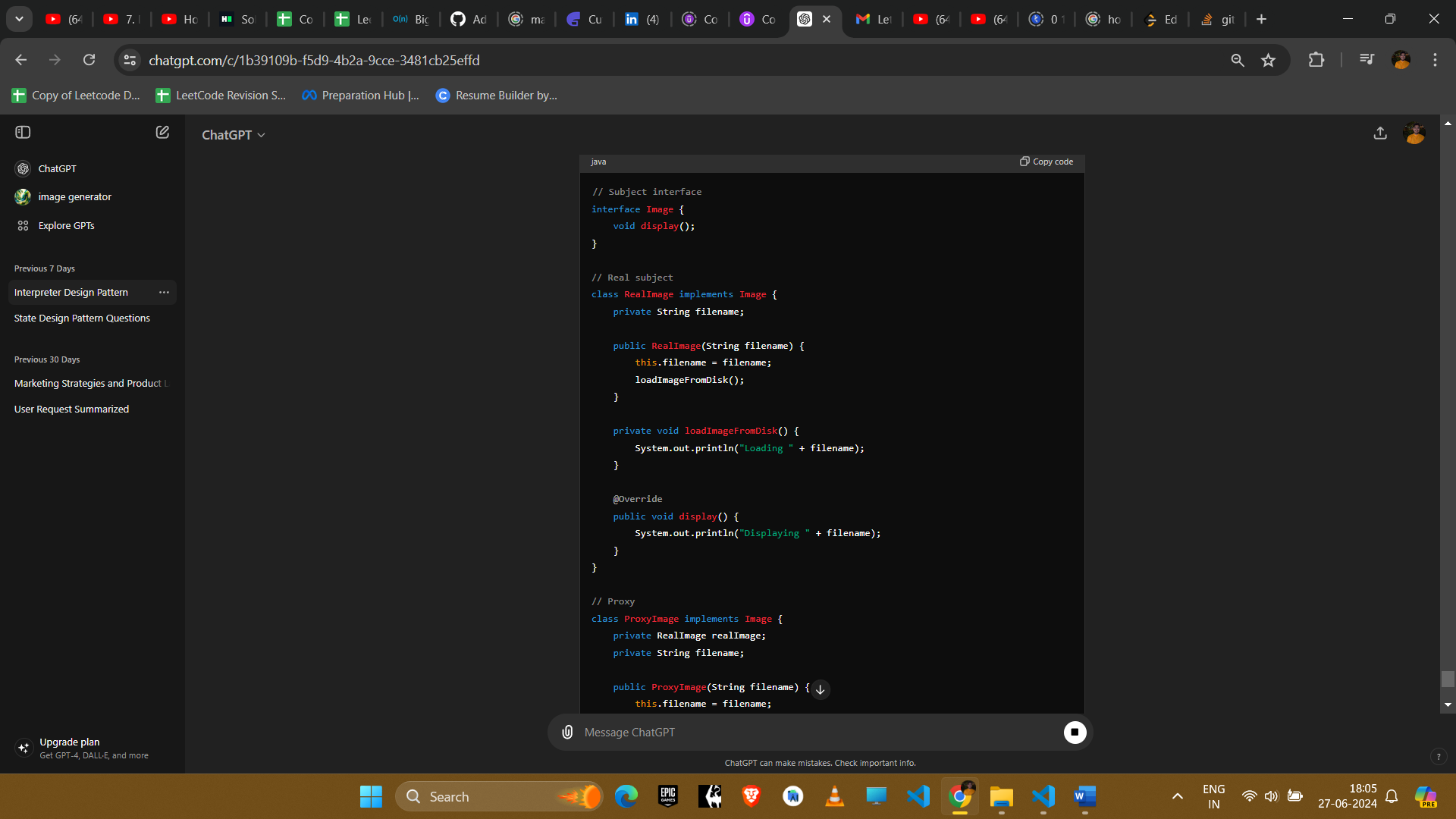


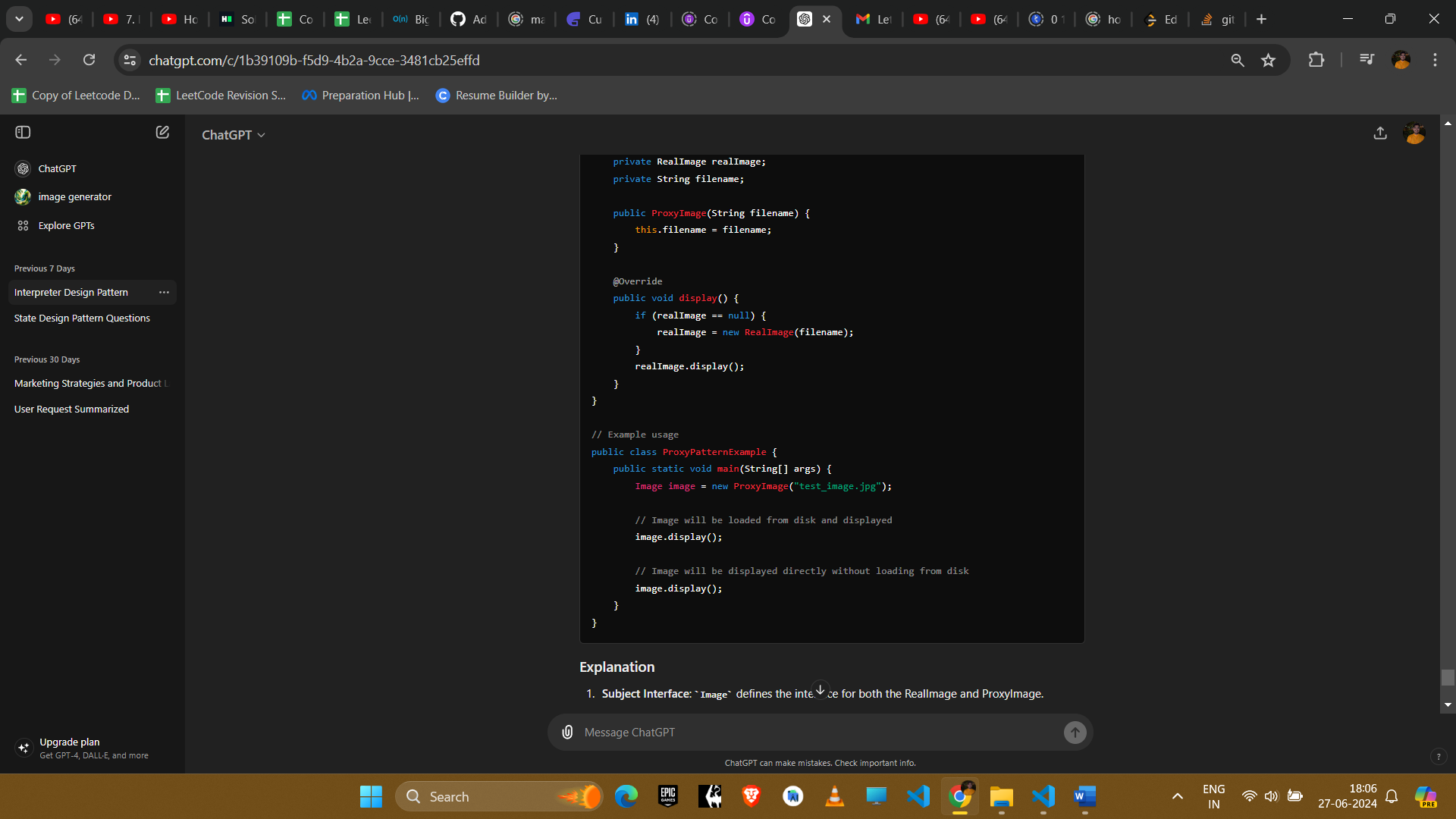




**Example**

Consider a scenario where we have a resource-intensive object, such as a large image that needs to be loaded and displayed.





**Explanation**

1. **Subject Interface**: Image defines the interface for both the RealImage and ProxyImage.
2. **Real Subject**: RealImage represents the actual object that is resource-intensive to create.
3. **Proxy**: ProxyImage controls access to the RealImage. It creates the RealImage only when necessary (lazy initialization).

**Example Uses in Amazon Interviews**

1. **Virtual Proxy**
   * **Scenario**: Delaying the creation and initialization of expensive objects until they are needed.
   * **Implementation**: Use a proxy to instantiate the real object only when it is required.
2. **Protection Proxy**
   * **Scenario**: Controlling access to sensitive or critical objects by checking permissions.
   * **Implementation**: Use a proxy to verify if the caller has the appropriate access rights before delegating the request to the real object.
3. **Remote Proxy**
   * **Scenario**: Encapsulating the details of remote interactions with objects in different address spaces (e.g., network communication).
   * **Implementation**: Use a proxy to manage the details of remote method invocations, handling the communication between client and server.
4. **Smart Proxy**
   * **Scenario**: Adding additional behavior when accessing an object, such as logging, caching, or reference counting.
   * **Implementation**: Use a proxy to add these additional behaviors transparently to the client.

**Conclusion**

The Proxy pattern is useful for controlling access to an object, managing the complexity of object creation, and enhancing performance and security. It provides a flexible way to introduce additional behavior when accessing an object, making it a valuable tool in various scenarios encountered in software development and system design.