### Mediator Design Pattern (Adding mediator b/w objects interaction)

The Mediator design pattern is a behavioral pattern that promotes loose coupling between objects by encapsulating how they interact and communicate with each other. It defines an object (the mediator) that acts as a central hub through which all interactions between participating objects (colleagues) are handled. This pattern facilitates communication by decoupling objects from each other, thus promoting easier maintenance, scalability, and flexibility in the system's design.

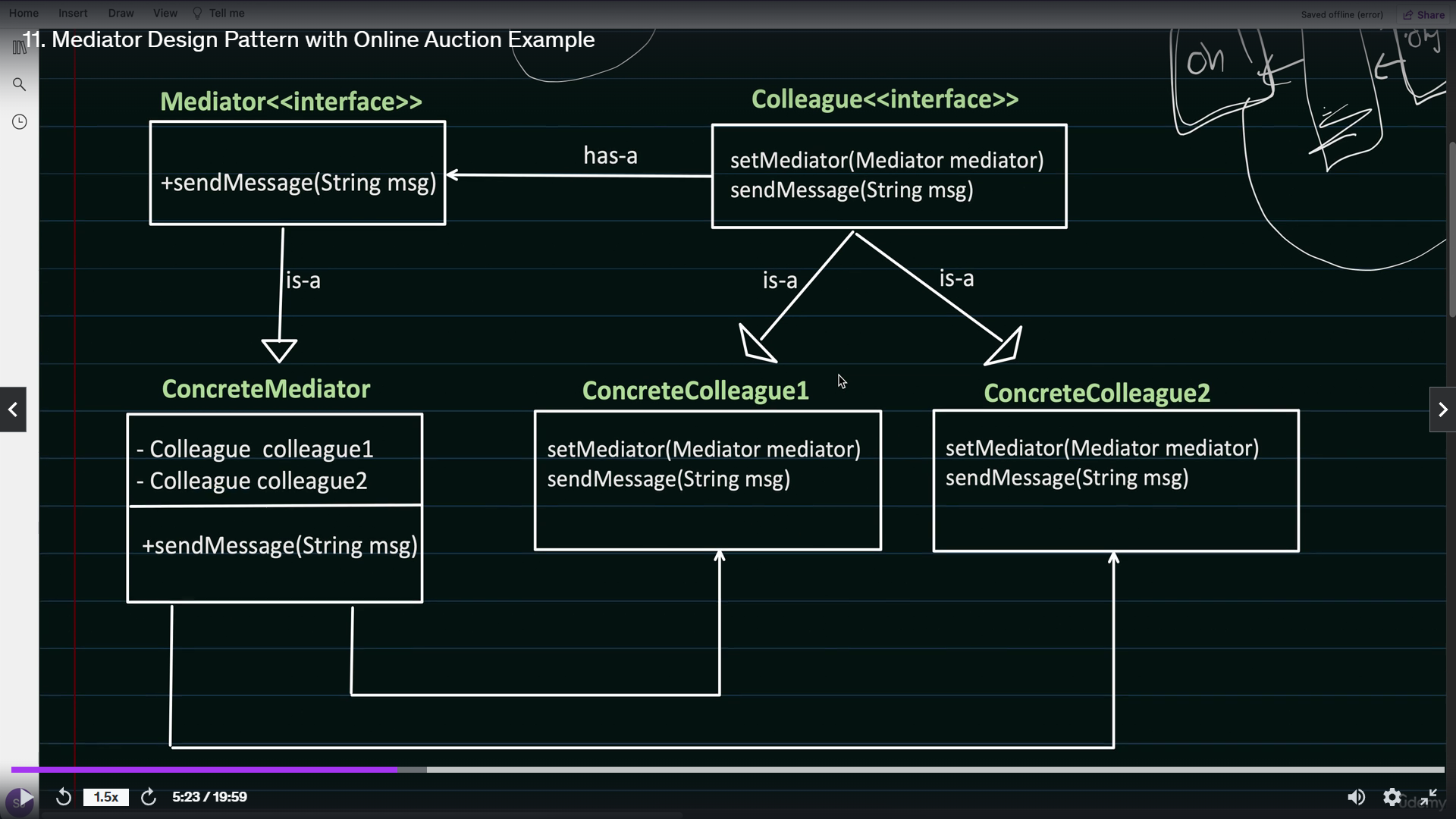
Eg usage: Airline Management system, Online Auction system

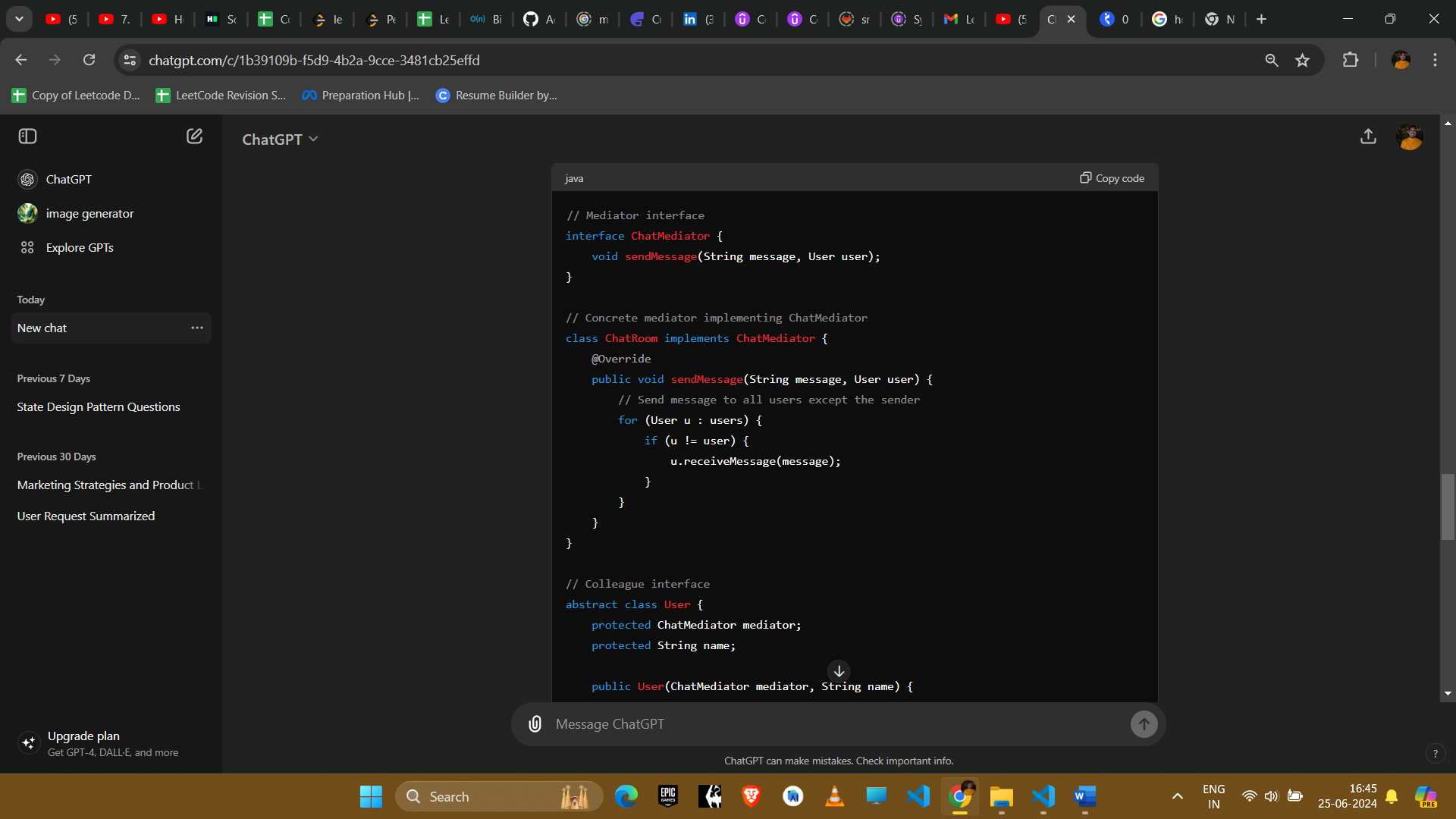
#### Explanation:

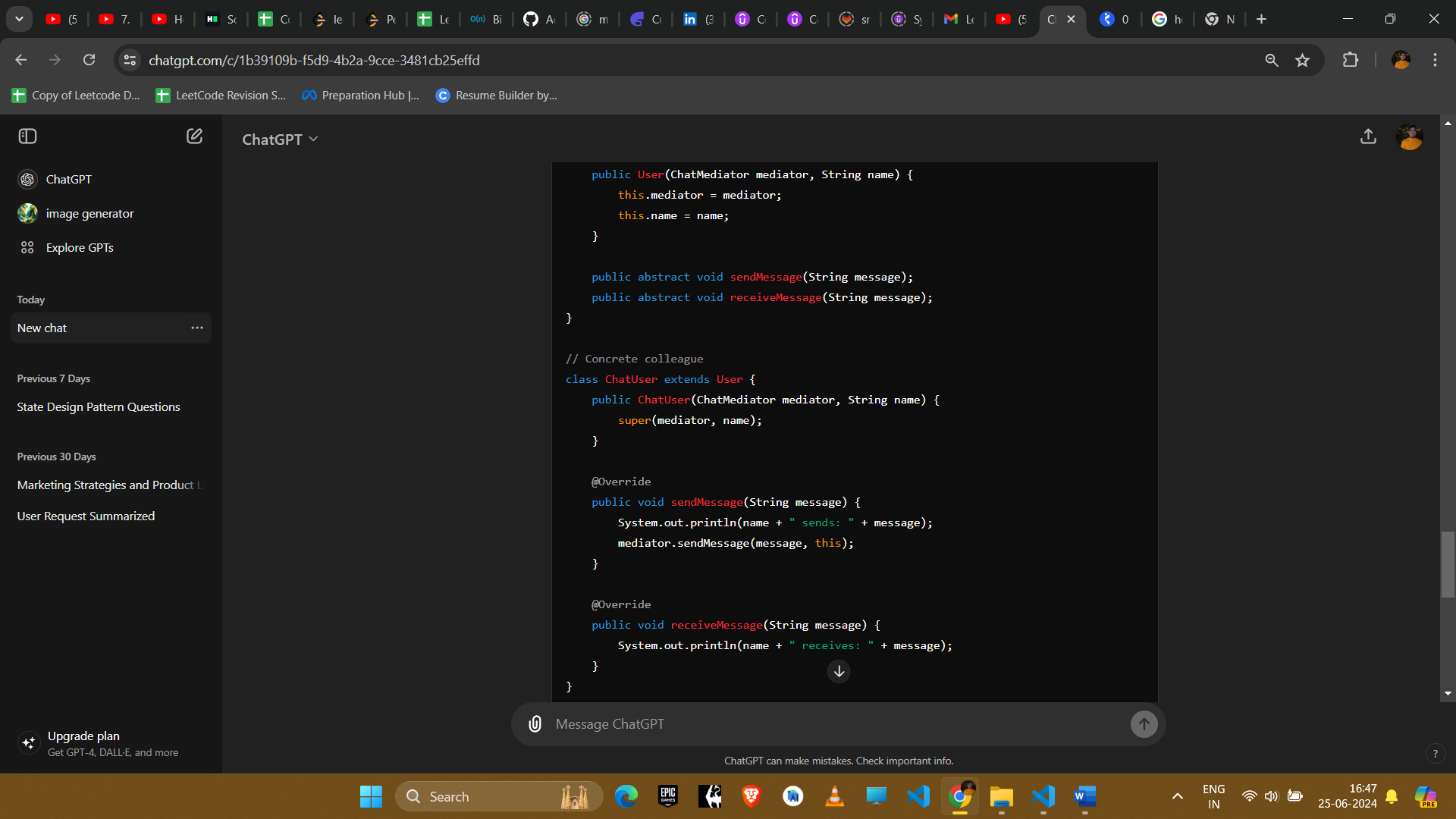
**Mediator Pattern Definition**: It defines an object (the mediator) that encapsulates how a set of objects interact. It promotes loose coupling by keeping objects from referring to each other explicitly and allows for their interactions to be varied independently.

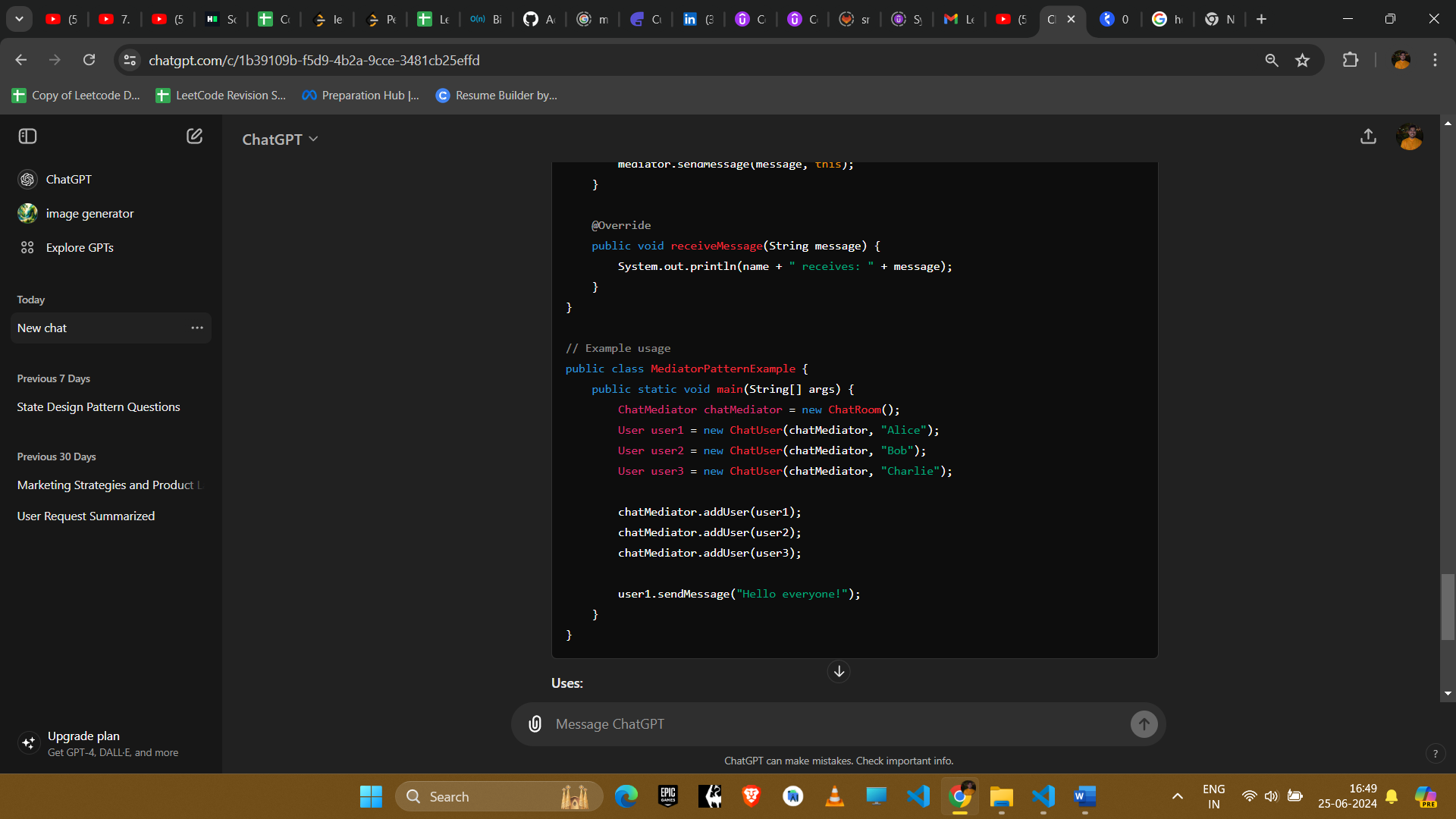
#### Example:

Imagine a chat application where multiple users can send messages to each other. Instead of each user directly communicating with every other user, a central chat room (the mediator) handles message distribution.









#### Explanation

1. **Mediator Interface**: Defines communication methods between colleagues.
2. **Concrete Mediator**: Implements the mediator interface and manages communication among colleagues.
3. **Colleague Interface**: Defines methods for colleagues to communicate through the mediator.
4. **Concrete Colleague**: Implements colleague behavior and interacts with other colleagues through the mediator.

#### Example Uses in Amazon Interviews

1. **Chat Application**
   * **Scenario**: Multiple users need to communicate in a chat room.
   * **Implementation**: The chat room acts as a mediator to manage message broadcasting among users.
2. **Flight Booking System**
   * **Scenario**: Booking components (e.g., flights, hotels) need to interact without direct coupling.
   * **Implementation**: A booking mediator coordinates the interactions between different booking modules.
3. **GUI Components**
   * **Scenario**: GUI elements (e.g., buttons, text fields) interact based on user actions.
   * **Implementation**: A GUI mediator manages event propagation and response handling among UI components.

#### Conclusion

The Mediator pattern enhances maintainability and scalability by centralizing complex communication logic. It reduces dependencies between objects and supports the addition of new functionalities without modifying existing components, making it suitable for large-scale systems requiring flexible and decoupled interactions.