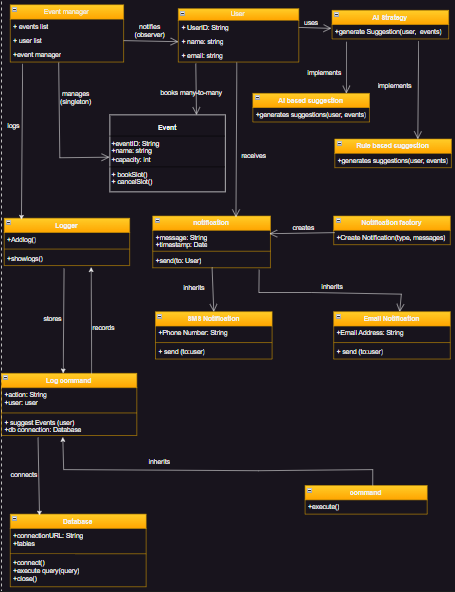
**Reflection**

For this system, we chose design patterns that best matched the core requirements. The Singleton pattern ensures a single Event Manager handles all events, simplifying multiple simultaneous registrations. The Observer pattern allows users to receive instant updates, the Factory pattern creates different notification types, the Strategy pattern supports AI suggestions, and Command/Logging logs all actions efficiently.

The main tradeoff was balancing simplicity vs. scalability. Using patterns keeps the design clear and maintainable, but very high user loads may need extra optimization. Concurrency is addressed with locks to avoid double-booking. Ethical considerations like user privacy were applied in notifications and logging. AI suggested Observer for real-time updates, which we adapted for efficiency.



**Conclusion**

This design models both static structure and dynamic behavior of a community event app. Patterns ensure maintainability, concurrency is handled, and real-time updates work effectively. It meets the assignment objectives, demonstrates ethical data handling, and provides a clear foundation for implementation.