Part 2 — Message Filtering in ChatRoom

Objective

Extend your existing ChatRoom so that **each observer can choose which messages to receive** by registering a **filter function (predicate)** when they subscribe.

What You'll Learn

- Per-observer filtering logic
- Overloaded subscribe() method
- Applying std::function predicates safely inside snapshot delivery
- Keeping all your Part 1 safety (RAII, weak_ptr, mutex)

Functional Requirements

1 Interfaces (No change in IObserver)

```
class IObserver {
public:
   virtual void update(const std::string& sender, const std::string& msg) = 0;
   virtual ~IObserver() = default;
};
```

2 Extend the Subject Interface

Add an **overloaded subscribe()** that accepts a filter:

```
class ISubject {
public:
   // previous
   virtual Subscription subscribe(const std::shared_ptr<IObserver>& obs) = 0;
// new overload with filter
   virtual Subscription subscribe(
      const std::shared_ptr<IObserver>& obs,
      std::function<bool(const std::string&, const std::string&)> filter
   ) = 0;
virtual void sendMsg(const std::string& sender, const std::string& msg) = 0;
   virtual void removeAll() = 0;
   virtual ~ISubject() = default;
};
```

ChatRoom Changes

3 Internal Slot Structure

Each subscriber now stores both a weak_ptr to the observer and its **filter** function:

```
struct Slot {
   std::weak_ptr<IObserver> obs;
   std::function<bool(const std::string&, const std::string&)> filter;
};
std::unordered_map<std::size_t, Slot> subs_;
```

4 Overload subscribe()

Implement two versions:

- One without a filter → calls the second overload with a nullptr filter (meaning "no filter → receive everything").
- One that takes a filter and stores it.

Return a **RAII Subscription token** exactly like Part 1.

⚙ 5 Sending Messages (sendMsg)

When a message is sent:

1. Build a **snapshot** of live observers (same as Part 1).
2. For each live observer:
    - If that subscriber has no filter (nullptr) → deliver the message.
    - If it has a filter → call if (filter(sender, msg)) → deliver only if true.
3. Deliver outside the lock (same safe snapshot principle).
4. Remove expired weak_ptrs during snapshot creation.

6?? Sample Observers

Keep it simple — all extend IObserver:

Class	Description	Example Filter
UserDisplay	Prints everything	none
KeywordAlert	Only messages containing "urgent"	msg.find("urgent") != npos
BlockSender	Ignores messages from "SpamBot"	