U37HUSJ4R : so on the UI there are 3 buttons
U37HUSJ4R : a hang up button, a pause button and a hold button (to play hold music and mute users mic)
U37HUSJ4R : lets pretend I have all my possible combined states
U37HUSJ4R : (I can't remember them all now :P)
U0LPMPL2U : The hung up status is already captured by whether the call is active or not right?
U23SA861Y : if if you are in a call, it can be onhold, paused or both
U37HUSJ4R : no, if a customer is on hold, then hang up will be false
U0LPMPL2U : That's the display value though, not the model state
U23SA861Y : `CallStatus = {onHold: Bool, recordingPaused: Bool}``CallState = HungUp | Active CallStatus`

U37HUSJ4R : (you can't hangup on a customer if they are on hold for example)
U0LPMPL2U : Could you have:```
type Call
  = HungUp
  | Active PauseStatus HoldStatus
```

U23SA861Y : also works
U23SA861Y : you can limit impossible states, but impossible state transitions need to be encoded into the update
U37HUSJ4R : interesting
U37HUSJ4R : so what would `PauseStatus`?
U0LPMPL2U : ```type PauseStatus = Paused | Unpaused
type HoldStatus = OnHold | Live
```

U23SA861Y : bools would also work there with type aliases
U23SA861Y : the named constructors are a bit more descriptive however
U0LPMPL2U : The type gives you extra safety though
U0LPMPL2U : because you can't accidentally pass the hold boolean when you meant to pass the paused boolean
U0LPMPL2U : You might also model this as:```
type RecordingStatus = Paused | Recording

type Call
  = HungUp
  | OnHold RecordingStatus
  | Active RecordingStatus
```

U0LPMPL2U : That way the tags on `Call` represent all the call states, each of which may or may not have an associated `RecordingStatus`
U37HUSJ4R : very interesting!
U37HUSJ4R : guess to make it more confusing though :wink: what happens if I need the statues to be `Maybe`?
U37HUSJ4R : because we don't always know the state of the call
```