4. YAGNI - You Aren't Gonna Need It.

Do not implement functionality until it is actually needed.

- Building only what is required now.
- Avoiding speculative features.
- Focusing on current requirements.

Why YAGNI.

Developers often.

- Predict future requirements incorrectly.
- Add unused code "in case"
- Increase complexity without benefit.

YAGNI helps ⇒ reduce code size, Reduce bugs., Speed up delivery.

wrong design

```
class PaymentService{
  void payWithCard(){}
  void payWithUpi(){}
  void payWithCrypto(){}//not required
}
```

correct design

```
Class PaymentService{
  void payWithCard(){}
}
```

When upi required → add it then.

Eg. Don't build 5 extra rooms because "may be guest will come"

- Build when there is an actual need.

YAGNI prevents premature complexity by discouraging implementation of unused or speculative features.