7 essential templates for software engineers

Have you ever wondered how the most tenured engineers achieve so much with the same or fewer hours than you?

It’s because their baseline work is raising the bar. Their worst work is already great

They achieve this by setting up systems.

Here are my 7 most used templates to ensure my worst work is already high quality.

You can find more practical advice in my newsletter.

#1 Email meeting invite

> 💡 Obtain the Notion version at this link

❓ When to use?

When scheduling a meeting for a document review or discussion

📝 Template

Attendance at this meeting is optional. Sending it to everyone for visibility.
I'll share meeting notes in this email thread.

Hello team,

In this meeting, we will review <>

Estimated reading time: XX (XX words, ~200 words-per-minute)

Objective: <>

Agenda:

- Read the document - 20 minutes
- Asking questions and discussion - 40 minutes

Document: <> **May receive modifications until the day of the meeting*

Thanks,
Fran

#2 Email meeting notes

> 💡 Obtain the Notion version at this link

❓ When to use?

Just after the meeting ends.
Click “reply-all” in the meeting invite to send it to all interested people

📝 Template

Thanks everyone for attending.

Document: <>

Attendees: <>

Meeting notes:
<>

Action Items:

- <Owner>: <>

Thanks!
Fran.

#3 Email introduction

> 💡 Obtain the Notion version at this link

❓ When to use?

When someone asks for help, and you know the person who can provide the answers.

📝 Template

Hi <A> and <B>

- <A>: <B> is <B background + relationship with me> <Reason of connection>
- <B>: <A> is <A background + relationship with me> <Reason of connection>

I'll leave you to connect, hope you can catch up soon! Feel free to move me to cc in the thread

Cheers,
Fran.

#4 1-pager doc

> 💡 Obtain the Notion version at this link

❓ When to use?

When you find an obstacle an agreement on the next steps

📝 Template

1. What is the problem?

<1-line **Purpose**, without detail on the solution>

(First paragraph - **background**)
1-line historic data / current state
1-line What is the problem with it (overview problem)
1-line Why is this important (overview why do we have to do it)
1-line what outcome we want

(second paragraph)
1-line what are we doing to solve it
1-line what is the improvement expected (mini success criteria)

2. What is our proposal?

1-line Present tense of the proposal

Actions

1-section Actions. A numbered list of individual actions, owner, and ETA

Risks

1-section call out risks. For each risk, add risk mitigation actions

Support required

1-section call support required and clear support provider

3. How do we measure success?

Pair of Metric and objective (aim for concrete numbers)

4. Appendix

- FAQs
- Data supporting the sharp text above
- Contact info: How to stay connected to the initiative

#5 Backlog task

> 💡 Obtain the Notion version at this link

❓ When to use?

When creating a new task in the backlog for anyone to execute

📝 Template

=== Description ===

<>

=== Implementation Steps ===

1.

=== References ===

- Product/Business requirements document: <>
- Technical design: <>

=== Acceptance criteria ===

-

=== At completion do ===

-

#6 Code checklist

> 💡 Obtain the Notion version at this link

❓ When to use?

Open this checklist when you are reviewing someone else's code.

It's also useful to review your commits before publishing for review

📝 Template

==Clarity==

- [ ] Easy to read
- [ ] Consistent Style
- [ ] Concise
- [ ] Documented
- [ ] Nothing Tricky
- [ ] Appropriate Abstraction
- [ ] No Debug Leftovers
- [ ] No Dumb Comments

==Correctness==

- [ ] No Obvious Bugs
- [ ] Unit Tests
- [ ] Test Coverage
- [ ] Thread Safe

==Design==

- [ ] Makes Sense
- [ ] Will Integrate
- [ ] No Duplication
- [ ] No Need to Refactor
- [ ] Not redundant

==Instrumentation==

- [ ] Appropriate Logging
- [ ] Appropriate Log levels
- [ ] Appropriate Metrics

==Constants==

- [ ] Config actually needed
- [ ] No Magic Numbers
- [ ] No Assumed Defaults
- [ ] Config Documented

==Versioning==

- [ ] Backward Compatible API
- [ ] Backward Compatible flows
- [ ] No Deployment Issues

==Dependencies==

- [ ] No risky new dependencies
- [ ] Safety mechanisms new network calls
- [ ] No unused dependencies

==Handling==

- [ ] Resilient to garbage return values
- [ ] Resilient to garbage input values
- [ ] Edge cases covered
- [ ] Clear exception strategy

==Performance==

- [ ] No premature optimization
- [ ] No obvious performance problems

#7 Technical Design Checklist

> 💡 Obtain the Notion version at this link

❓ When to use?

Open this checklist when you are reviewing someone else's technical design.

📝 Template

My own understanding of the problem:

<>

Key design decisions:

<>

Questions to answer in the review

<>

==Start it right==

- ☐ Functional requirements completion
- ☐ nonfunctional requirements quantified and SLAs: TPS, Latency, Data Size
- ☐ Preserve/expire data
- ☐ Localization of UI

==Proposal==

- ☐ Reinventing the wheel?
- ☐ Evaluated alternatives?
- ☐ Identified error scenarios?
  - ☐ Expected: (e.g. validations)
  - ☐ Unexpected (e.g. timeout, out of memory)

==Context==

- ☐ Who are upstream / downstream dependencies

==Technologies==

- ☐ Follows golden path?
- ☐ Prioritizes serverless to reduce operations?
- ☐ Prefers managed service over DIY?

==Service calls==

- ☐ Caching in your side / dependency side?
- ☐ Safety mechanisms: Throttling, circuit breakers, kill-switch, API tiering
- ☐ Should this be sync or async?
- ☐ (Sync) Retry once immediately? (latency, retry amplification)
- ☐ (Async) Ordering and duplication of events?

==Data==

- ☐ Eventual / strong consistency?
- ☐ Read-after-write problem?
- ☐ Can rollback data or have to issue compensating operations?
- ☐ Andon cord mechanism to stop the damage?

==Operations==

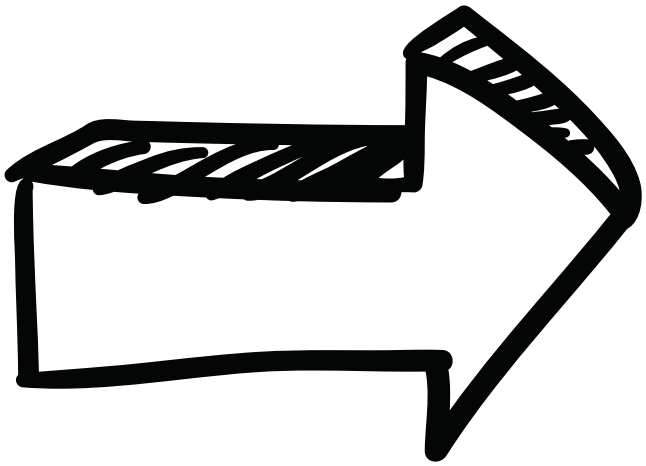
- ☐ State management
- ☐ Infra costs and optimization opportunities
- ☐ Metrics
- ☐ Failure conditions
- ☐ Alarming
- ☐ Deployment design (how to rollout to production?)

- [ ] Service configuration
- [ ] Authentication&Authorization
- [ ] Data Privacy
- [ ] Least privilege principle
- [ ] Error logging
- [ ] Test plan
- [ ] Migrations; How to support both systems? How to calculate mismatches?
- [ ] Implementation plan (dependencies in implementation)

BECOME A

CAREER STRATEGIST

READING THE WEEKLY ARTICLES



strategizeyourcareer.com