Developer's Eisenhower Matrix

Your Code Quality & Velocity Dashboard

Hot Fix	**Real Engineering**
Automate/Delegate	**Delete/Ignore**

How to Use This as a Developer

Your value isn't in closing tickets—it's in solving problems elegantly. This matrix helps you balance urgency with code quality.

The Developer's Decision Tree

if (production_broken) return 'Q1'; if (improves_codebase || builds_skills) return 'Q2'; if (someone_else_can_do) return 'Q3'; else return 'Q4'; // probably delete

Remember

• Technical debt is a Q2 priority, not Q4 • Learning time is work time • Clean code is faster in the long run • Meetings aren't your real work • Ship early, iterate often

The Developer's 3-Question Algorithm

Quick decision tree for any task:

Question 1

Is production broken or will the build fail?

If YES: It's URGENT → Continue to Q2

If NO: It's NOT URGENT → Continue to Q2

Question 2

Will this improve code quality, performance, or my skills long-term?

If YES: It's IMPORTANT → Place based on urgency

If NO: It's NOT IMPORTANT → Place based on urgency

Question 3

Can this be automated, scripted, or handled by someone else?

If YES: Consider DELEGATING/AUTOMATING (especially if in Q3)

If NO: You need to code it yourself

Examples for Each Quadrant

Crisis Mode (Hot Fixes)

Real fires that block users or development

- **Production server is down** - Users can't access the application
- **Security vulnerability in prod** - Data breach risk is immediate
- **Blocking bug for major release** - Can't ship without fixing
- **Data corruption issue** - Actively damaging user data
- **Payment processing broken** - Directly impacts revenue

Quality Zone (Real Engineering)

The work that separates good devs from great ones

- **Refactoring legacy code** - Reduces future bugs and speeds development
- **Writing documentation** - Future you will thank current you
- **Learning new technologies** - Staying relevant in the industry
- **Performance optimization** - Better UX and lower costs
- **Test coverage improvement** - Confidence in your code

Interrupt Zone (Context Switches)

Others' urgencies disrupting your flow

- **Non-critical Slack messages** - Batch check 2-3 times daily
- **Status update requests** - Automate with tools
- **Routine code reviews** - Schedule specific time blocks
- **Meeting note-taking** - Rotate responsibility
- **Environment setup for others** - Document the process

Danger Zone (Time Sinks)

The rabbit holes that eat your day

- **Premature optimization** - Making it 1ms faster doesn't matter yet

- **Bikeshedding in PR reviews** - Tabs vs spaces doesn't matter
- **Endless tool configuration** - Your vim setup is fine already
- **Rewriting working code** - If it ain't broke, don't fix it
- **Framework FOMO chasing** - The new hotness isn't always better

Common Teacher Traps to Avoid

The Shiny Object Syndrome

Constantly switching to the newest framework or tool instead of mastering what you have.

Solution: New tech is Q2 learning time, not Q1 production code. Learn it, evaluate it, then decide.

The Perfection Paralysis

Endlessly refactoring code that already works well enough.

Solution: Ship at 80% and iterate. Perfect code that never ships helps no one.

The Hero Complex

Being the only one who understands critical parts of the codebase.

Solution: Document and share knowledge. Bus factor of 1 is a risk, not a badge of honor.

The Yes Developer

Accepting every feature request, bug fix, and 'quick favor' that comes your way.

Solution: Your job is to deliver value, not to make everyone happy. Learn to say 'not now' or 'here's the trade-off.'

EISENHOWER MATRIX

The Developer's Daily Standup with Your Matrix

10 minutes to optimize your algorithm for the day

Morning (7 minutes)

1. Check build status and error logs
2. Review PR comments and blockers
3. Sort Jira tickets into quadrants
4. Choose ONE Q2 task to complete
5. Block 2+ hours of focus time

Afternoon (3 minutes)

1. Commit and push completed work
2. Update ticket status
3. Note any tech debt created
4. Plan tomorrow's Q2 task
5. Close unnecessary tabs (yes, all 47)