IssuePilot

YouTube

0606

The Problem

New Codebase

Way too many files, zero context. Where do you even start?

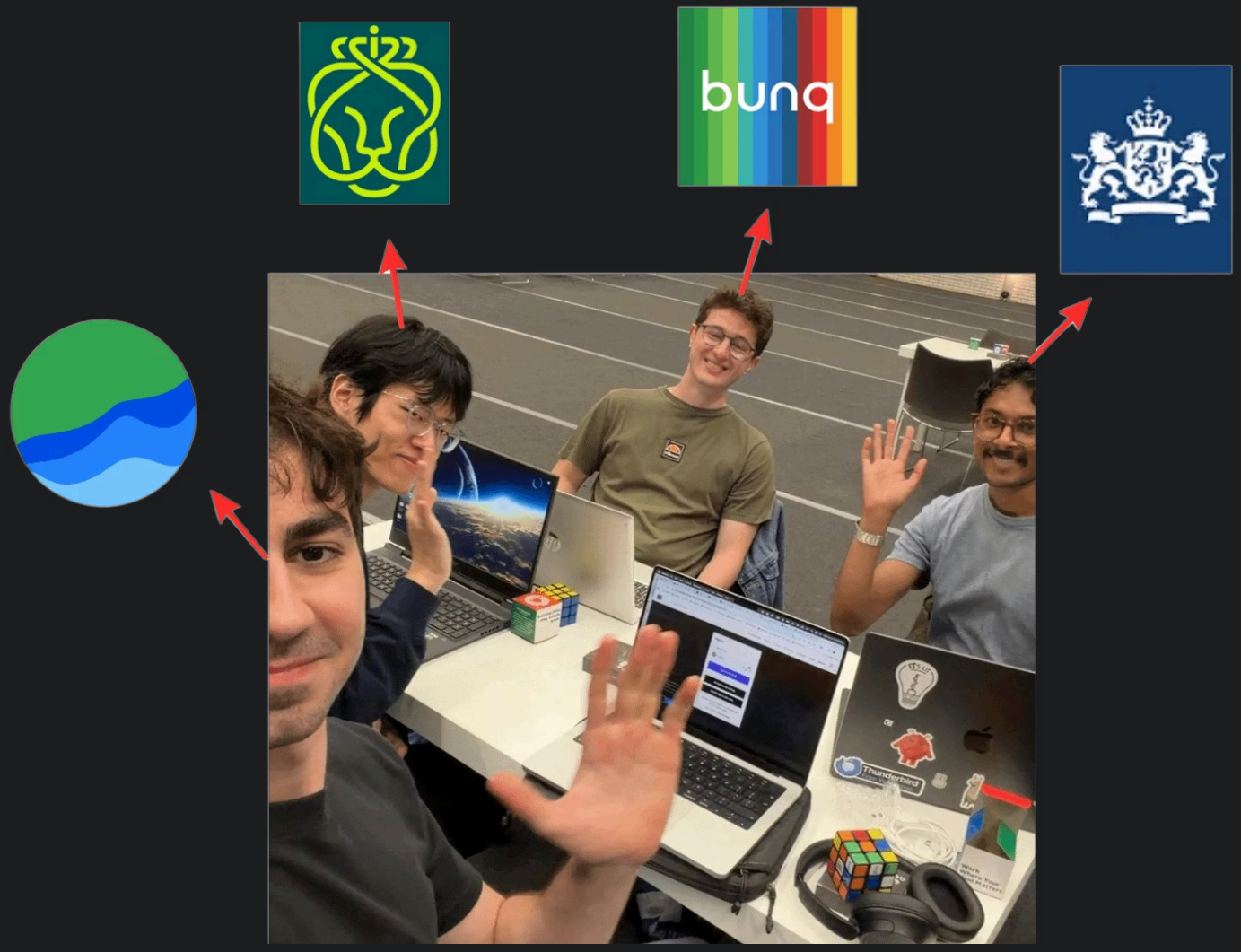
Too Much Context

Ask a senior? They're busy. Ask Slack? Good luck.

58–70% of developer time is spent understanding code. Only around 5% is spent editing it.

Source: Dror G. Feitelson, “From Code Complexity Metrics to Program Comprehension,” *Communications of the ACM*, 2023.

Why are WE trying to solve this problem?



#relatable

Existing onboarding documentation?

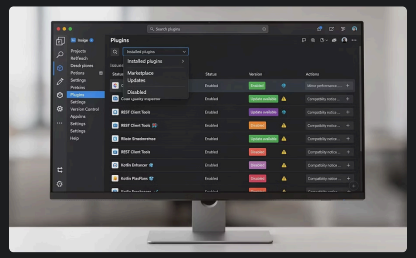
Code keeps updating

Time consuming

Boring!

Do not learn the whole codebase.

The Solution: Issue Pilot

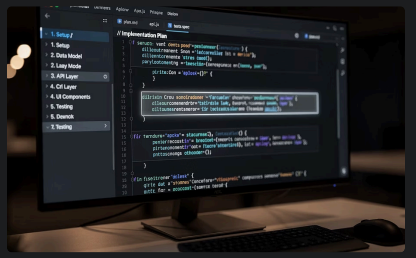


Pick an Issue

Select any GitHub issue directly inside IntelliJ.

Analyze the Issue

IssuePilot scans relevant files, recent commits, and git history.



Resolve issue. Yourself.

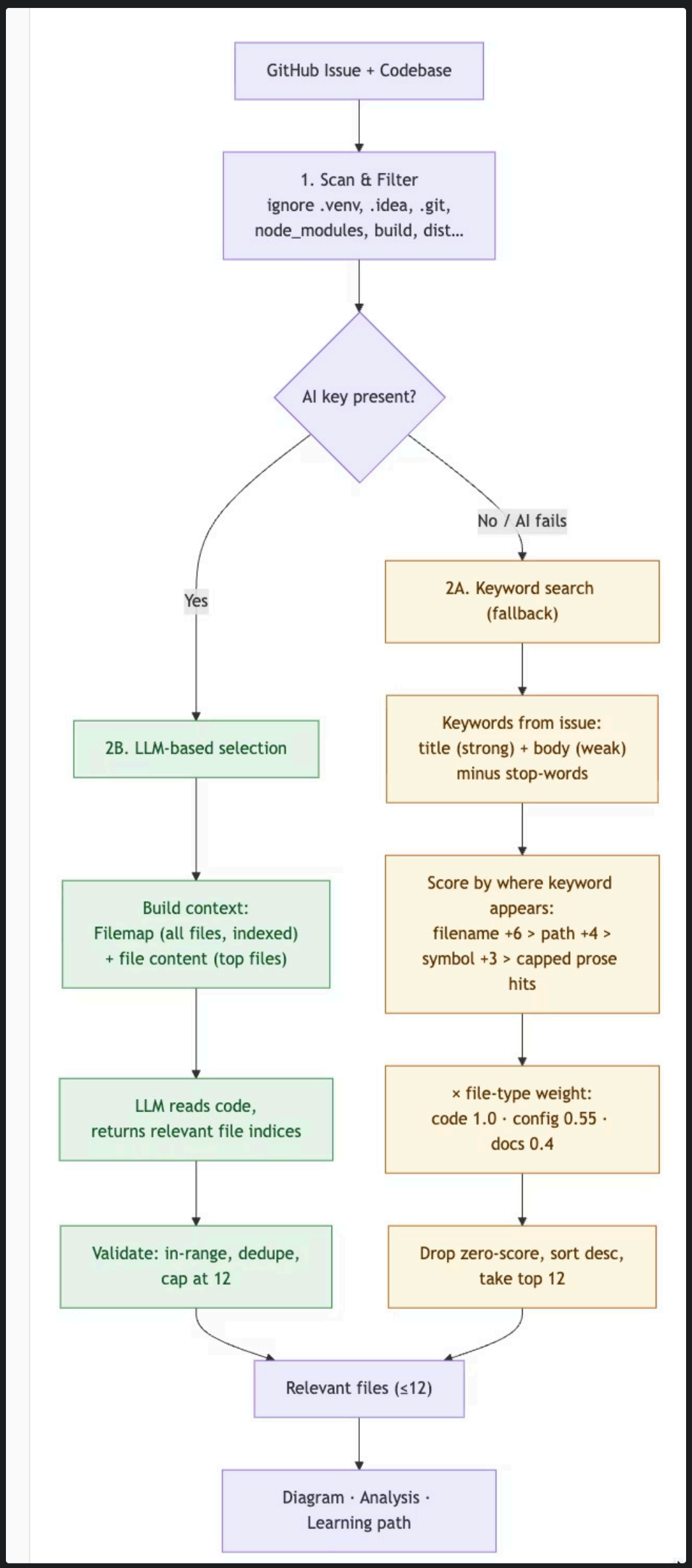
IssuePilot is a tool to help you code, not create infinite code in a second

Over time, DO learn lots of the codebase, one issue at a time

Interns only need to work on one tiny slice.... **finding out everything about that slice is the hard part.**

DEMO time :)

How it works?


GitHub Issue + Codebase
1. Scan & Filter
ignore .venv, .idea, .git,
node_modules, build, dist...
AI key present?
Yes
No / AI fails
2B. LLM-based selection
Build context:
Filemap (all files, indexed)
+ file content (top files)
LLM reads code,
returns relevant file indices
Validate: in-range, dedupe,
cap at 12
2A. Keyword search
(fallback)
Keywords from issue:
title (strong) + body (weak)
minus stop-words
Score by where keyword
appears:
filename +6 > path +4 >
symbol +3 > capped prose
hits
× file-type weight:
code 1.0 · config 0.55 ·
docs 0.4
Drop zero-score, sort desc,
take top 12
Relevant files (≤12)
Diagram · Analysis ·
Learning path

Why Now

AI can write code.

But developers still need to **understand** it.

The fundementals

no JR dev = no SR dev

Did we create something...
people would actually use?

Research & Iteration

23

Users Interviewed

Every person we interviewed was involved in software in some way

5

Prototype Rounds

Iterated based on direct feedback from real users.

2

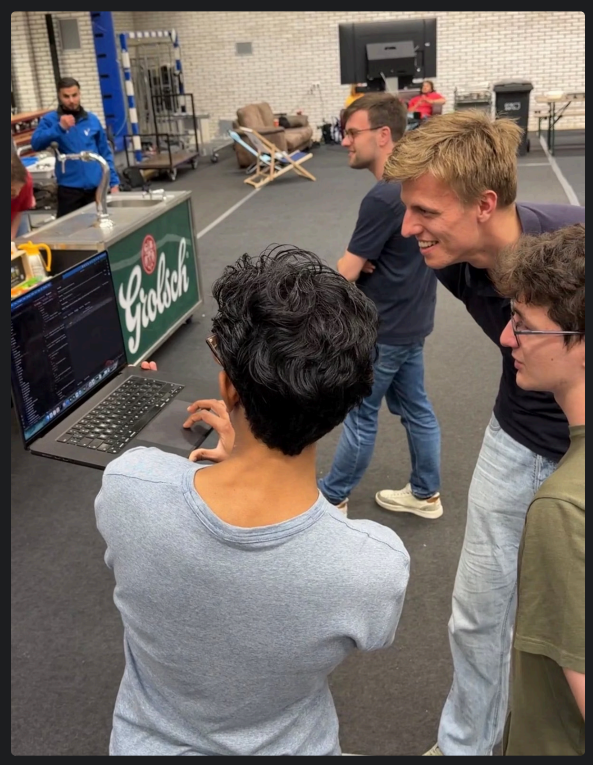
Core Pains

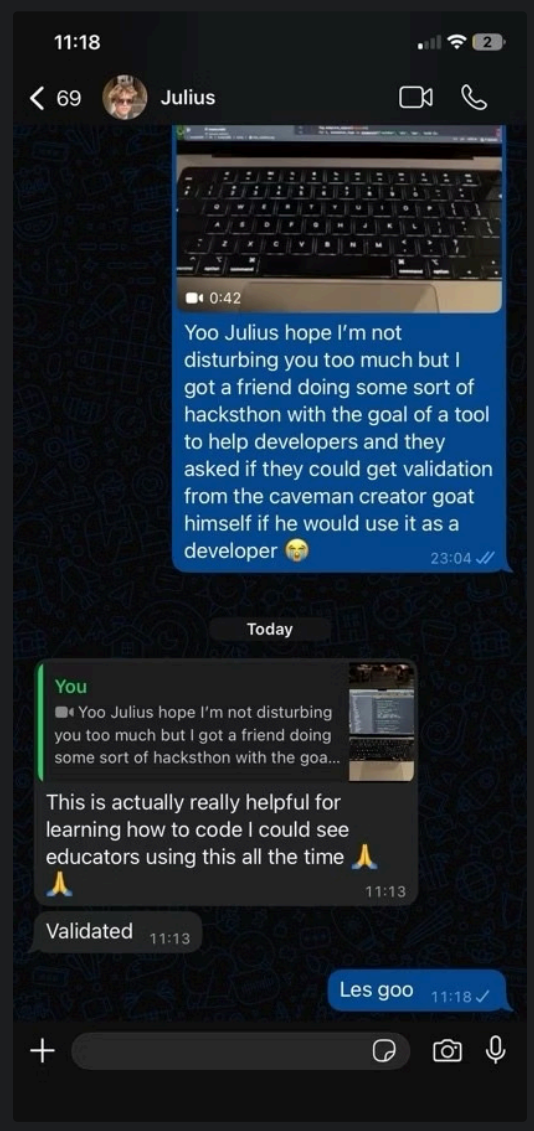
Context overload.

Unfamiliar Codebase

Not just built off assumptions.


TUDelft TUDelft
DE DELFTSE HERUITVENDAGEN


Grolsch


11:18
69
Julius
0:42
Yoo Julius hope I'm not disturbing you too much but I got a friend doing some sort of hacksthon with the goal of a tool to help developers and they asked if they could get validation from the caveman creator goat himself if he would use it as a developer
23:04
Today
You
Yoo Julius hope I'm not disturbing you too much but I got a friend doing some sort of hacksthon with the goa...
This is actually really helpful for learning how to code I could see educators using this all the time 🙏🙏
11:13
Validated 11:13
Les goo 11:18

Expected questions we have the answers to :)

Q: why is the “learn first” section not the same as “code map”?

A: They are calculated differently, “Code map” is fully ranked, call it relevance (keyword in the file name is a very strong signal: ex LoginController.kt for a login related issue.

Simple matching by name at first:

File name: InventoryManager.kt strong match

Folder path: /player/inventory/... strong match

Class/function: loadInventory() medium match

File content: "crash when inventory..." weaker match

Q: Why is “symbol count” important?

A: symbol count is important because of a simple principle: the more classes a file has, the more likely it is to be imporant

(This proved to be surprisingly good)

Q: what is “commit churn”?

A: “commit churn”: how many recent commits are touching the ranked files. Not perfect measure, but we realized it’s VERY indicative

Q: why could there be test files in the code map

A: Looking at tests is an excellent way of figuring out what is going on in the codebase, especially if its related to the issue. Our heuristic actually helped us understand that.

Q: why 60% 25% 15%...?

1. **Relevance (60%)**: the dominant signal, because "you're working on this issue right now" is the most direct, immediate reason to read a file "This is a whole established subfield: treating a bug/issue report as a *query* and source files as a *document corpus*, then ranking files by textual similarity" **http://liqingan.cn/doc/ESE2022_AnEmpiricalStudy.pdf**
2. **Symbol count (25%)**: secondary, because a big API surface suggests a file is structurally foundational, even if it's not specifically about this issue, Source: 2025 paper from Meta on Arxiv
3. **Commit churn (15%)**: the weakest signal, since "recently changed" only loosely correlates with "important to understand" Source: Nagappan & Ball's 2005 ICSE paper "Use of Relative Code Churn Measures to Predict System Defect Density"

Bug Localization