Developing Applications With An AI Code Assistant

Practical Insights From Real-World Projects

Video generated using Google Gemini

Richard Moulton

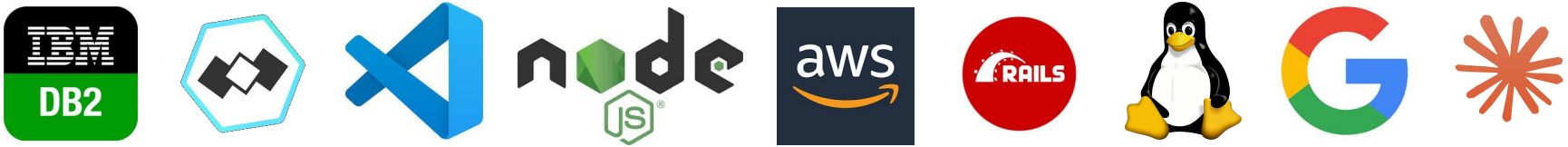
RM Software Services Ltd
richard@rmsoftwareservices.co.uk
https://rmsoftwareservices.co.uk/

RM SOFTWARE SERVICES

v1.0

Richard Moulton

- IBM i developer for 37 years
- Started working as a trainee RPG developer in 1989
- Now working as a freelance IBM i developer since 2019
- I provide development services to a handful of IBM i clients
- I specialise in IBM i RPGLE and of course DB2 for i
- Working more and more with Open Source Software
    - I've contributed to quite a few projects!
    - I've even created a few projects!



I work mostly from my home office in Weare Giffard, North Devon, UK

[illegible] miles from Milton Keynes (by road)

What are we going to discuss?

Setting The Scene

- My Experience
- What is a code assistant?
- Model vs Code Assistant

The Main Topic

- Project #1: RM ACS Launcher
- Project #2: RM Console
- Project #3: RM Connector

Extra Topics

- Context Is Everything
- The Agentic Loop

My Code Assistant Story

My Experience With Code Assistants

1. Started using ChatGPT (for free)
   - Copy and paste code to and from the web interface
2. Moved to Claude and Claude Code
   - Started with the Individual Pro plan
   - Now on the Individual Max plan
3. The evaluation version of IBM Bob
   - I soon ran out of tokens
   - Timing wasn't right for these projects

About 18 months ago

Around 8 months ago

Towards the end of 2025

I have only used code assistants with local source (on my laptop)

I use Claude Code via the VS Code extension

I haven't done any significant work with RPG code yet

I will reference Claude Code a lot during the talk but this is not specific to Claude Code.

I would expect a similar experience with other code assistants.

What is a code assistant?

What is a code assistant?

A **code assistant** is a software tool powered by large language **model**s (LLMs) that helps developers write, understand, and modify code through natural language interaction.

Key Capabilities

- Generates and completes code from plain-English descriptions
- Explains unfamiliar code, errors, and concepts
- Refactors, debugs, and writes tests on existing code
- Operates inside your IDE or terminal

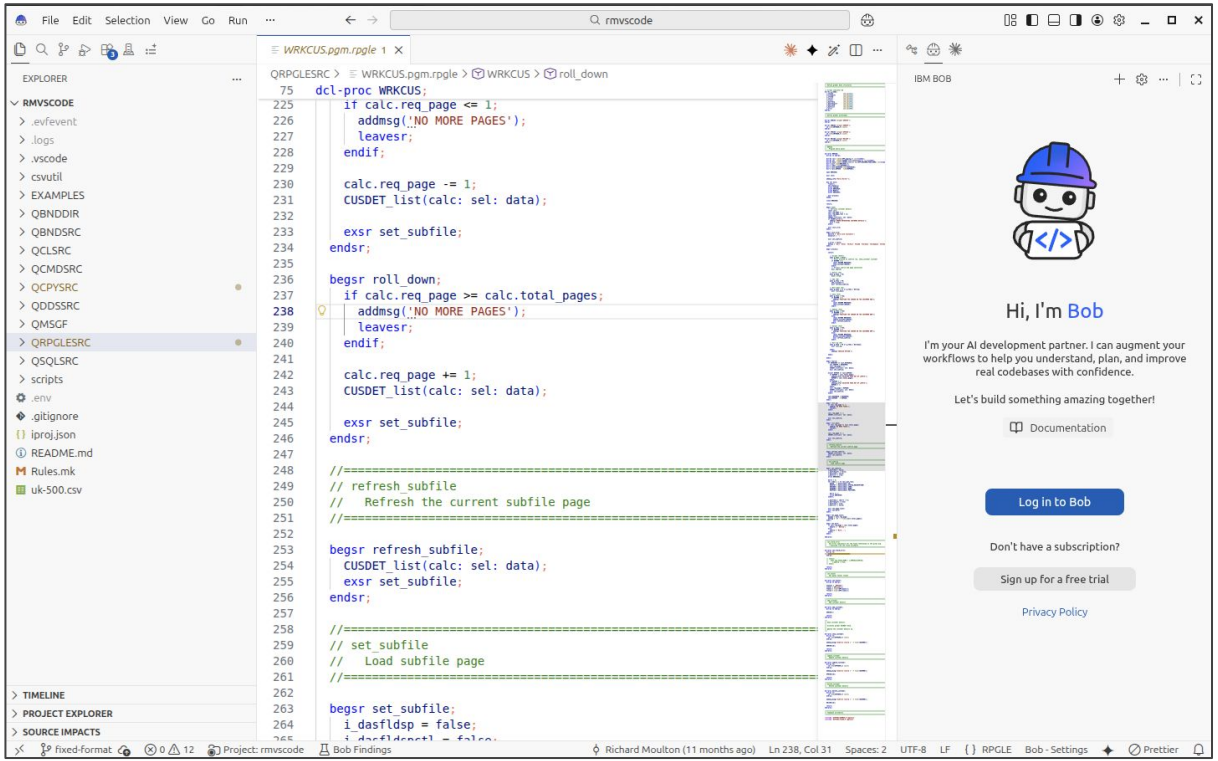
Some Examples

- IBM Bob
- Claude Code
- Codex
- GitHub Copilot
- Cursor
- Google Gemini

Where do you start? Today, I would start with Bob.

I currently use the Claude Code extension in VS Code

What is a code assistant?



The IBM Bob IDE

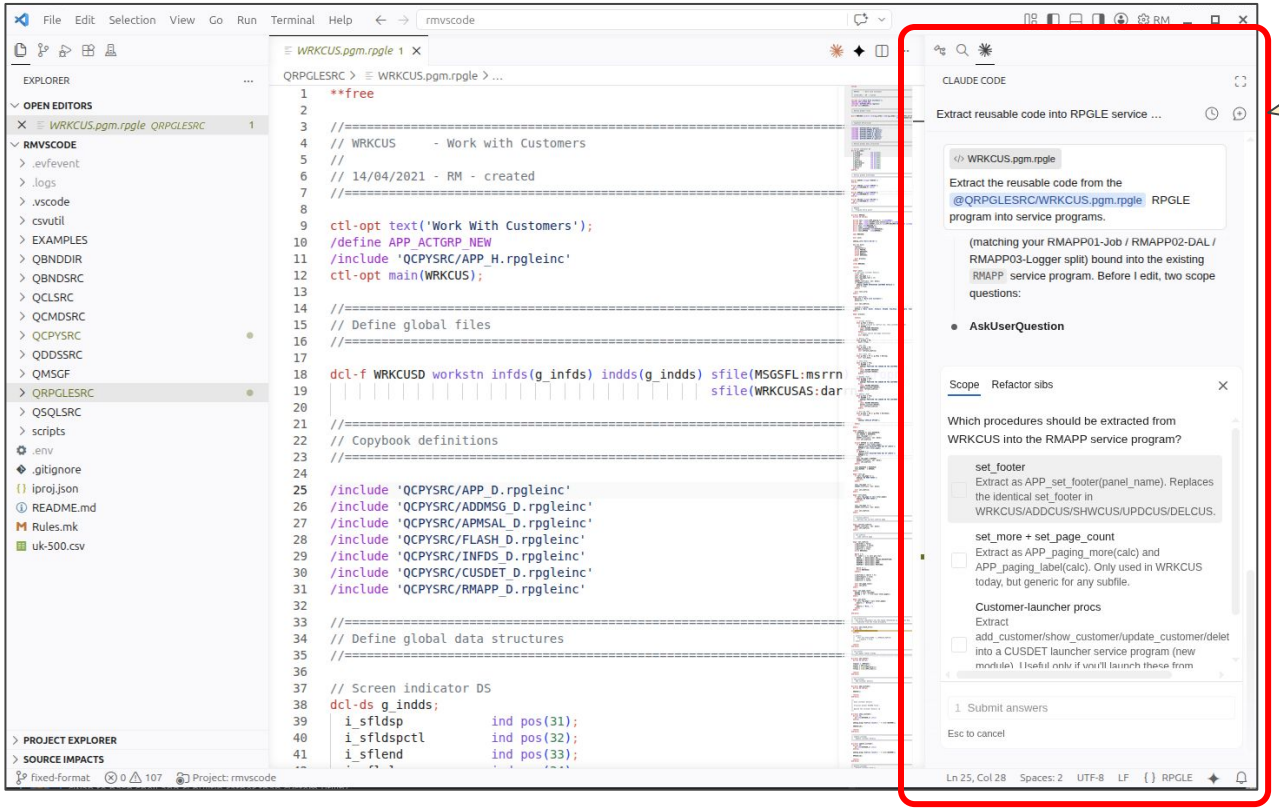
A full IDE - a fork of VS Code.

Also available in your terminal as Bob Shell.

Great documentation:

https://bob.ibm.com/docs/ide

What is a code assistant?

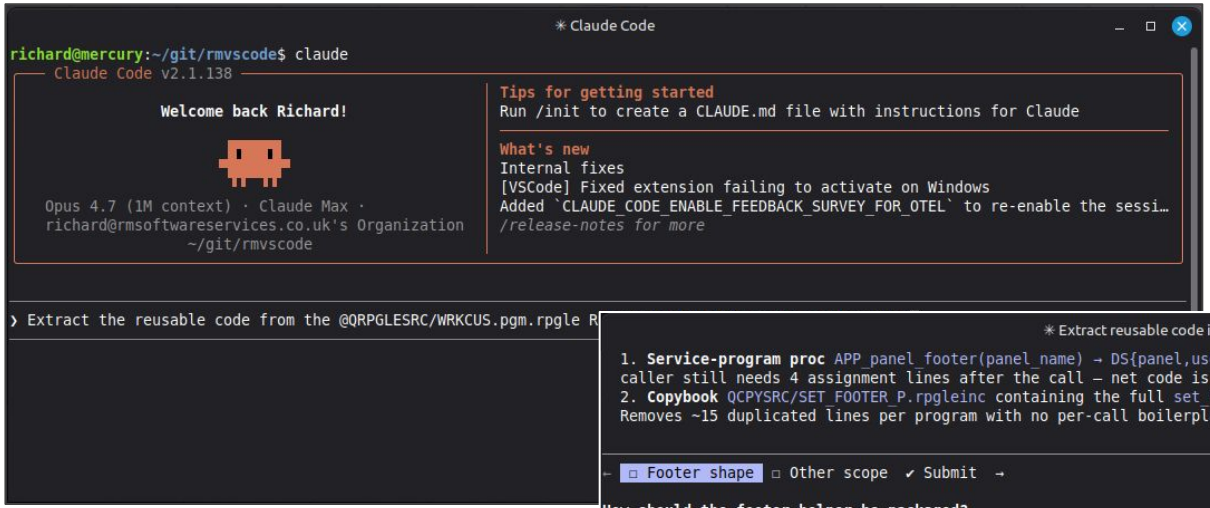


The Claude Code extension running in VS Code

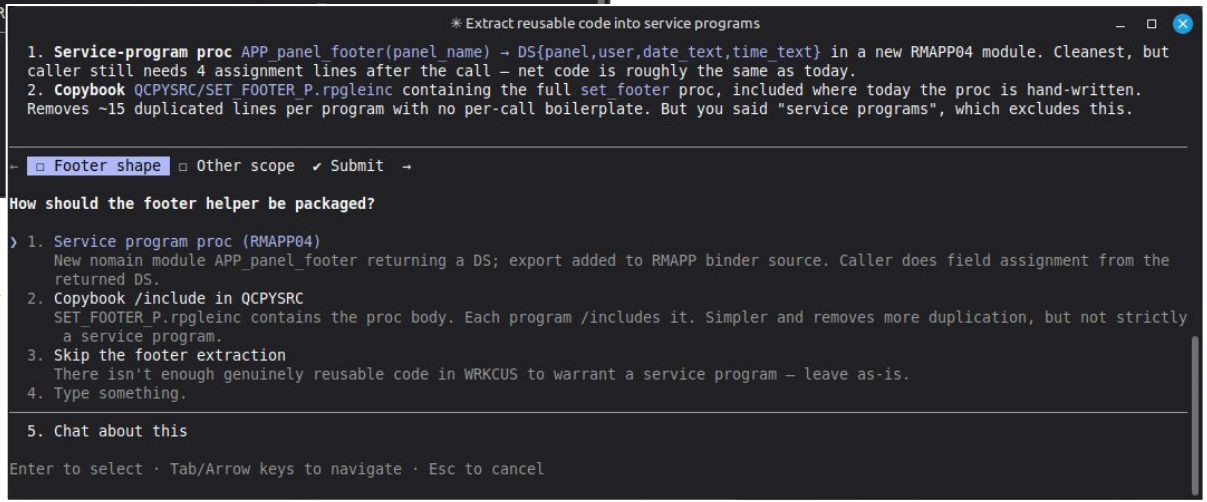
Claude Code options:

- CLI
- On the web
- On the desktop
- VS Code
- JetBrains IDE

What is a code assistant?



The Claude Code CLI



Model vs Code Assistant

Model vs Code Assistant

Context = Your prompt, conversation history, file contents, tool outputs, project instruction files and skills - all supplied by the code assistant.

The Model

- Reasons over the full **context**
- Decides which tool to call, with what arguments
- Reads tool output and reasons about it
- Generates the actual code
- Plans, replans, course-corrects

The Code Assistant

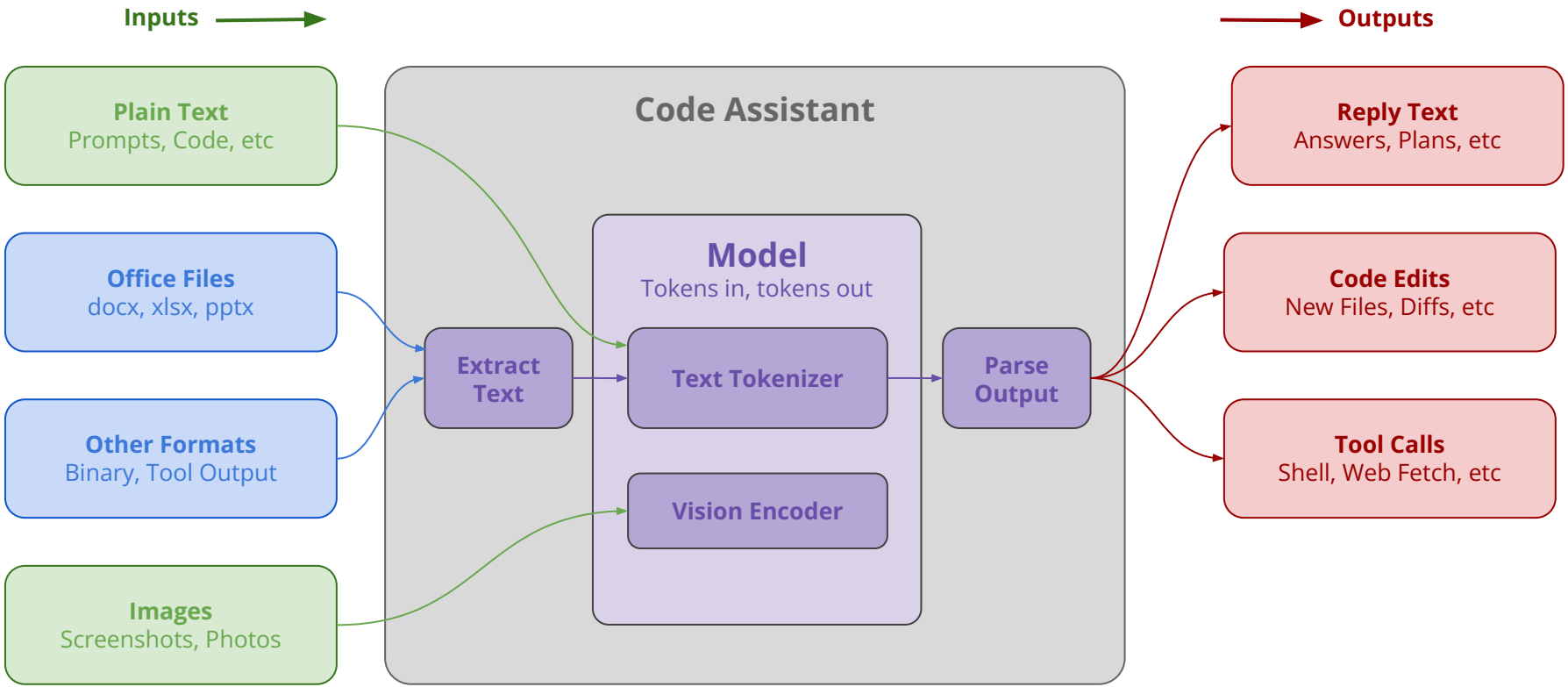
- Executes the tools (file edits, shell, web fetch)
- Manages the context window
- Loads project instruction files and skills on demand
- Provides safety: permissions, checkpoints, plan mode
- Saves and resumes sessions
- The UI you actually look at

The model thinks. The code assistant does everything else.

Model vs Code Assistant

Tokens are what AI platforms charge!

Models Only Work In Tokens



Model vs Code Assistant

Tokens are what AI platforms charge!

Model Tokens

- Models only work in tokens.
- Tokens are numerical representations of text fragments.
- All models can tokenize text. Modern models can also process images natively via a vision encoder.
- Tokens are counted, both input and output tokens.

Tokens & Pricing

- Typically charged per million tokens.
- Input and output are priced differently.
- Output is usually more expensive - often 3-5 x more expensive.
- Image inputs cost tokens too. How big of an image are you sending?
- **Context** management matters financially, not just technically!

“Memory” is an illusion created by the assistant resending everything each turn.

Context = Your prompt, conversation history, file contents, tool outputs, project instruction files and skills

Model vs Code Assistant

“Memory” is an illusion created by the assistant resending everything each turn.

The Model

The model has no memory, it’s stateless between calls.

Project Instruction Files
Markdown files that give the assistant persistent, project specific context: naming standards, coding conventions, architecture notes, build/test commands, gotchas.

Skills
Packaged, reusable capabilities (instructions + scripts + reference files) that the assistant can invoke on demand.

The Code Assistant

- Every single turn, the assistant sends the entire conversation - the **context**
- This includes your prompt, conversation history, file contents, tool outputs, project instruction files, skills, etc
- Long sessions on large files can get expensive faster
- Most providers now provide **prompt caching** - unchanging parts of the context get cached on their side and charged at a lower rate
- Most assistants don’t keep every file indefinitely - once a file isn't actively being discussed, it may get dropped from the **context**
- Most assistants provide a mechanism to perform a **context** compaction - older turns are summarised

Start fresh sessions for new tasks!

Project #1: RM ACS Launcher

Project #1: RM ACS Launcher

Why create a new ACS Launcher?

This solves a very specific problem for me.

I need to connect as different users to multiple systems and some systems require different users with different access rights.

The IBM ACS Launcher doesn't allow you to easily swap between different users.

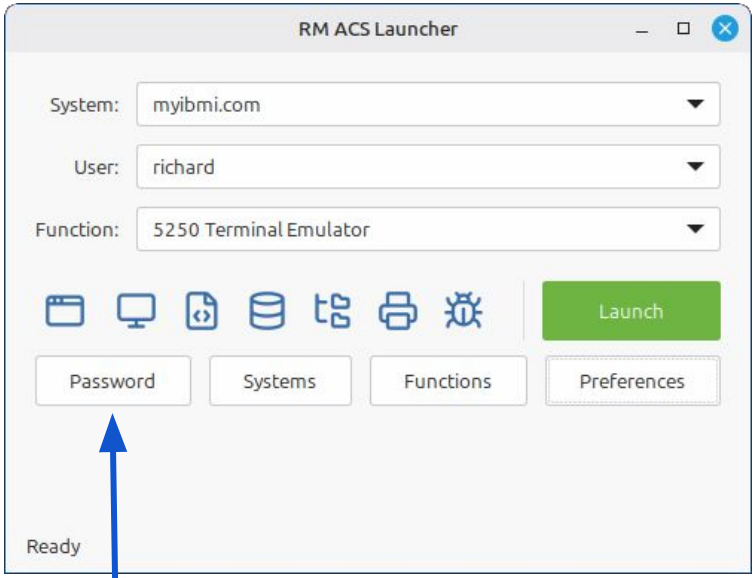
Why mention this pet project?

Written in Python using GTK3 (Linux GUI library).

I know very little Python and very little about GTK3.

Designed with Claude, coded by Claude.

Uses the ACS CLI



Is it secure?
I asked Claude and Gemini to perform a full security review.

Probably!

Am I the only person in the world that will ever use this application?

https://github.com/richardm90/rm-acs-launcher

Project #1: RM ACS Launcher

Summary from the Claude Code security review.

Summary

Item	Status
Password storage (libsecret/keyring)	✅ Secure — correct use of `Secret.COLLECTION_DEFAULT` + typed schema
Password transmission to ACS	⚠️ MEDIUM — leaks via `/proc/<pid>/cmdline` for the lifetime of the logon process
Config file handling	✅ No secrets persisted to disk by the launcher
Subprocess invocation	✅ argv-style, no shell injection
Other categories (deserialization, eval, etc.)	✅ None present

The single concrete finding is the argv-leak of the password during logon. The keyring-side mechanism the user asked about is implemented correctly; the weakness is downstream of it, where the cleartext value is handed to ACS.

As a result of this security review I needed to understand this particular problem and design a solution to address the vulnerability.

Project #1: Lessons Learned #1

- I don’t know how to create a Linux desktop application though I do bring a lot of developer experience to the party - just not in Linux desktop applications.
- I therefore put a lot of time into the design, using Plan mode. I kept iterating over the design until I fully understood how it was going to be implemented. Ensuring the password was secure was a particular focus during the planning phase.
    - **Plan first, then execute**
- Put as much into the plan as possible, I did miss some stuff.
    - State your dependency preference up front, for me it was standard libraries and already installed system packages only.
    - Ask the question “What happens on upgrade?”. The assistant can easily skip this step.
    - The assistant decided where the app should be installed and it wasn’t the best practice location. I missed this during planning phase and had to patch it later.
    - Decide on your commit message style, code style, and naming once, write it down, point the assistant at it. I wasn’t clear on this and had to patch it later.

Project #1: Lessons Learned #2

- The assistant generated nearly 1,500 lines of code as one initial commit. It would have been better to **break this into separate tasks** and commits e.g. config layer, keyring handling, etc. I missed this during planning phase.
- Include release process documentation, 6 months down the road I will have forgotten!
- Was this a successful project?
    - Patch pace slowed: v0.1.0 → 0.1.1 (next day) → 0.1.2 (a week) → 0.1.3 (three months)
- This is a 1,500-line desktop app with installer, config, keyring integration, and packaging - built and shipped to v0.1.0 in 4 hours. Without an assistant, it's a weekend project at best - and that’s with the skills in the technologies involved.
    - That changes what's worth building
    - **Tools that were "not quite worth the time" might now be**

Project #2: RM Console

Project #2: RM Console

A modern web application for IBM i built with Vue.js and Node.js, connecting to DB2 and all running on IBM i.

A project I've been thinking about for years and now possible with the help of a coding assistant.

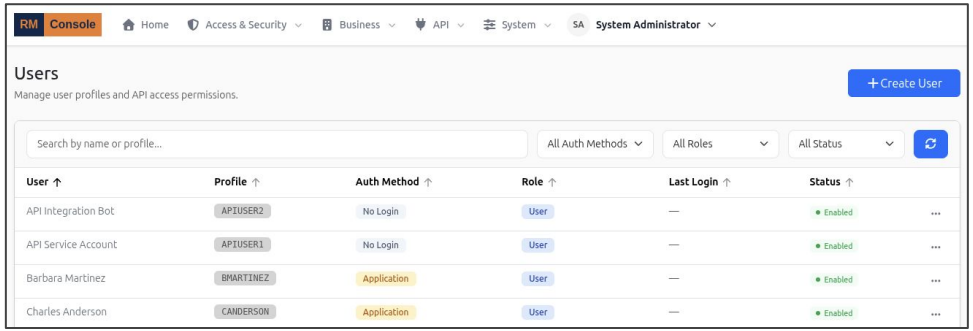
Utilising technologies I'm already familiar with.

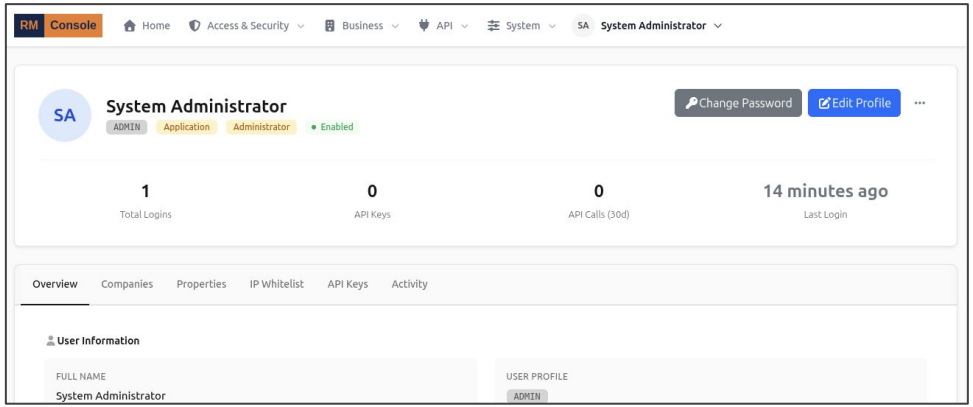
- Node.js
- Vue.js
- DB2 for i

An early decision in the planning phase.

I wanted to develop the main application locally and use standard Node.js testing tools with a full test suite on the backend.

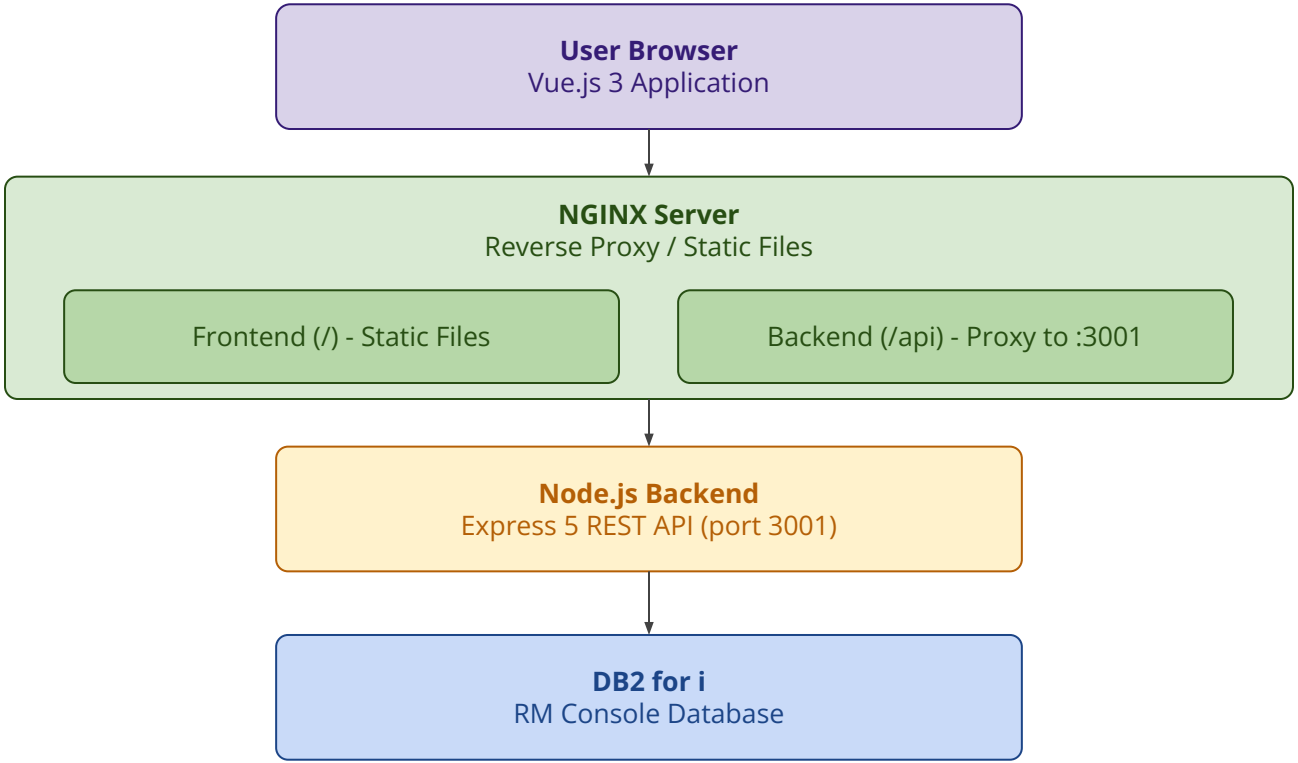
Another early decision.





https://github.com/richardm90/rm-console

Project #2: RM Console Architecture

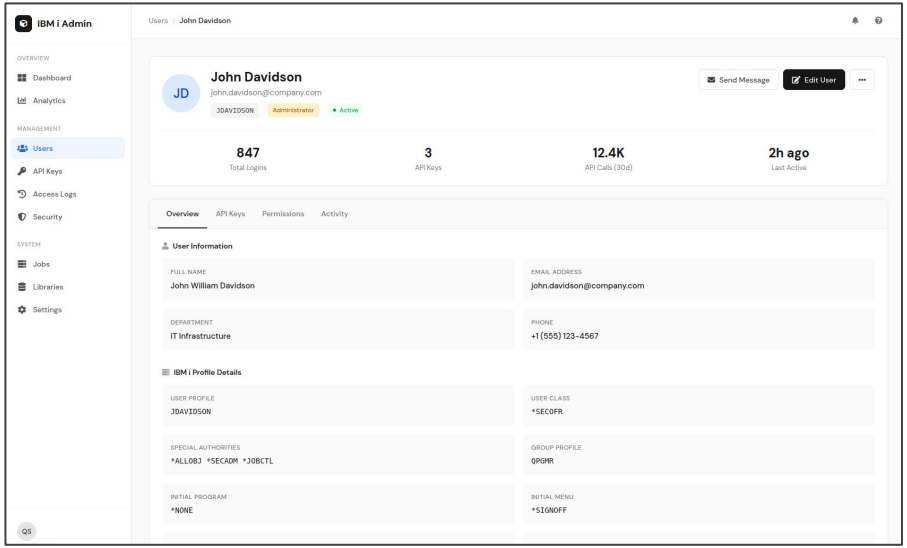


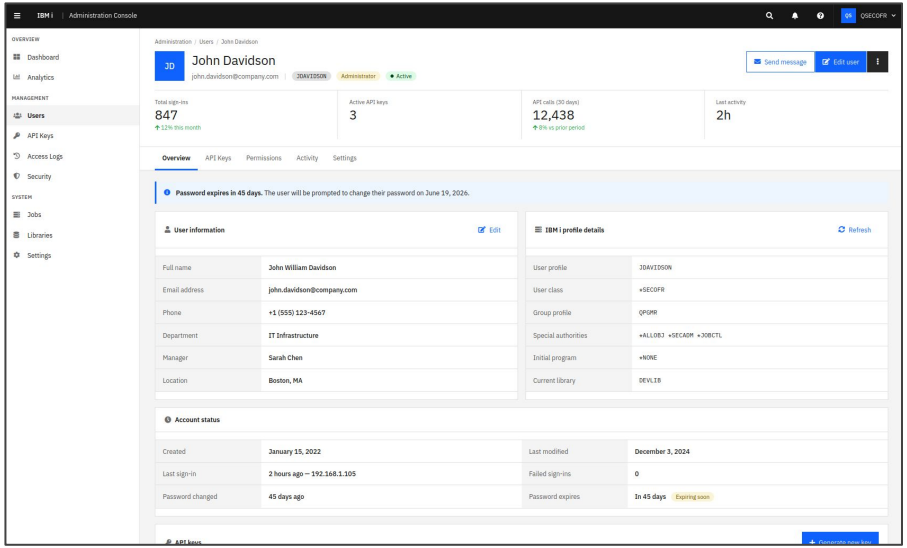
Project #2: Benefit Highlights #1

Not one of my core strengths

I get a lot of value from using code assistants to design UIs.

They can come up with different variants - corporate, modern, dark, etc - and quickly!





Project #2: Benefit Highlights #2

Architectural refactoring from Fat Controllers to Repository Pattern.

Using a coding assistant lowers the cost of doing architectural changes properly.

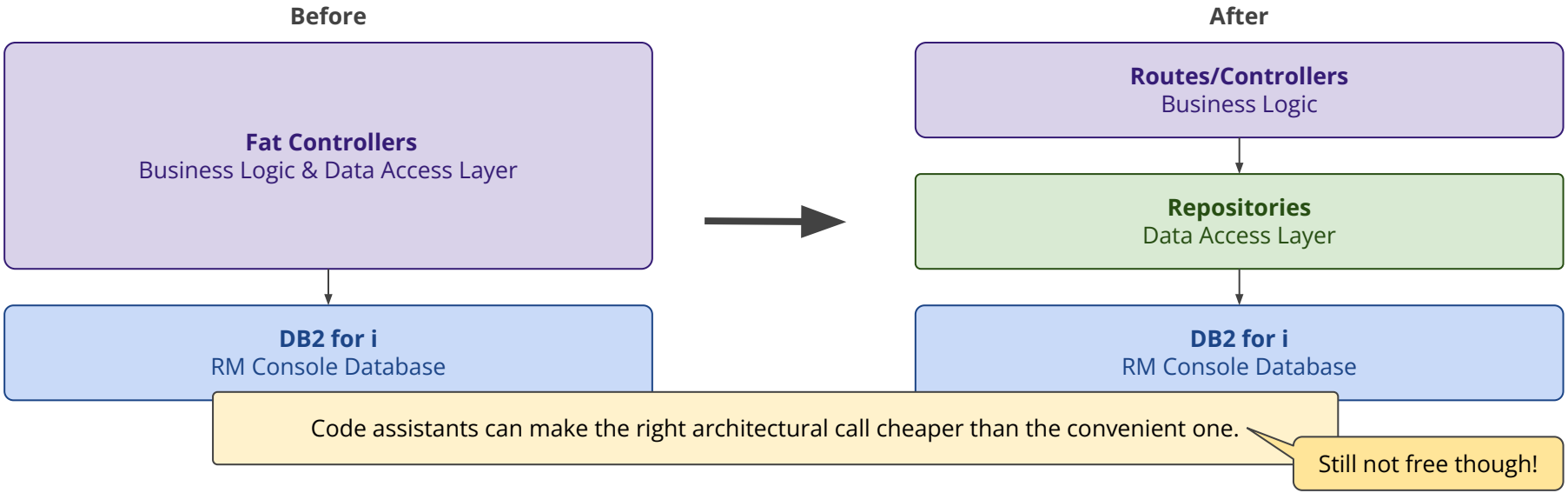
The disciplines that normally get skipped (tests, documentation, simultaneous refactor of consumers) can more easily be made at the point of change.

The "I'll document it later" debt can make legacy systems harder to maintain a decade later.

I already had full test coverage in place, which gives you further confidence to make architectural changes.

Why make the change?

- Separation of concerns
- Mock repositories easily in tests
- Test business logic without database
- Easy migration to stored procedures



Project #2: Benefit Highlights #3

A full database migration system and database deploy script.

The coding assistant helped with the research of best practices for database deployment and migration.

With the correct directions the coding assistant will create migration files, maintain deploy script, etc as the database evolves.

Deploy Script

Handles fresh installs and an upgrade from a specific point - via the schema version table.

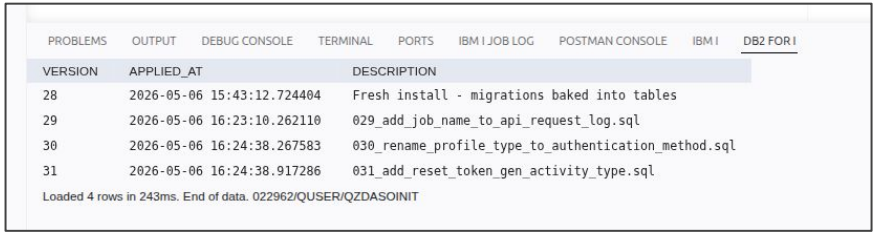
Options to run the script interactively or non-interactively.

An option to create test data with a new database.

Migration System

Numbered migration files - each containing the SQL statements to move the database forward.

Tracks the applied migration files - via a schema version table.



Project #2: Lessons Learned #1

This is the big one!

Fail To Plan - Plan To Fail

- Less of a plan and more of a set of goals
- I asked the assistant to go, and it did
- I didn't review enough of the code it generated
- It's really tough to anticipate the decisions the assistant will make

You can start with a brief plan but break it up into functional parts. As you start on each part then drill into the detail.

A lot of time was spent on refactoring. Like a lot!

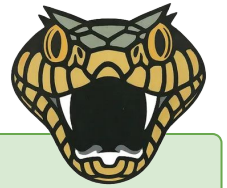
I very rarely give it permission to edit automatically.

This comes back to reviewing the changes. Catch poor decisions early.

Code assistants are most effective working from a plan, rather than when you ask it to "dive in".

The "dive in" approach can work for Proof of Concept projects!

Project #2: Lessons Learned #2



To Mapepire Or Not To Mapepire?

I wanted to utilise Mapepire as the DB2 for i connector.

There is a standard Node.js mapepire-js client library, cleaner interface than nodejs-idb-pconnector and it doesn't need to run on an IBM i server.

It was going to allow me to achieve my goal of developing the main application locally.

The Problem

It performs well enough when running remotely from your laptop but it does not perform well when compared to nodejs-idb-pconnector when running directly on an IBM i.

I had to come up with a solution pretty darn quickly!

I'd already got a lot of the application up and running.

See Project #3!

The goal for local development vs running on IBM i for production should have been clearer in my plan.

Project #2: Lessons Learned #3

IBM i & Node.js Package Compatibility

I ran afoul of utilising several Node.js packages that aren't compatible with the IBM i platform.

Vite & Vitest

I originally used the Vite build tool /dev server and Vitest unit test framework in the project.

These utilise the esbuild package to build the project and esbuild won't install on IBM i.

This required a significant rewrite of 130+ tests utilising WebPack & Node test runner.

bcrypt

I originally used the bcrypt package to hash passwords and API keys. This package won't build on IBM i.

I had to swap to use the bcryptjs package.

Fortunately this required very little rework as the package implementation was broadly the same.

Make it clear in the plan that any new packages should be checked for platform compatibility.

Project #2: Lessons Learned #4

API Design & Database Connections

Numerous web pages included data from more than one resource endpoint.

Each piece of resource data was being retrieved separately as an asynchronous request - running in parallel.

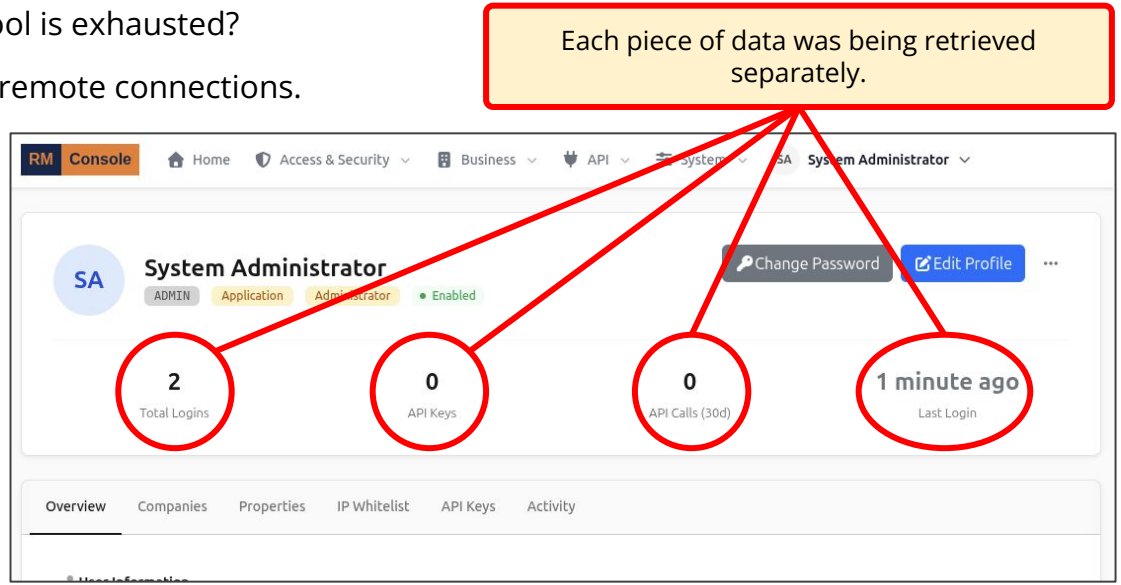
How long before my database connection pool is exhausted?

Especially obvious during development with remote connections.

I ended up creating composite APIs rather than use the pure resource based APIs.

In this example I created a summary endpoint (GET /users/:userId/summary), and 5 queries were consolidated to 2 queries.

I updated the assistant project instruction file to include an API Design section.



Project #2: Lessons Learned #5

Making The Same Mistake

Vue.js allows you to create UI components, which can be used as different elements to build up a page. These components include the styling to maintain a consistent look across your app.

Each time I asked the assistant to create a new list view, etc based on an existing list view it would do just that but it always wanted to introduce custom styling.

The assistant kept undoing the major benefit of creating and using Vue.js components.

Skills

A `vue-frontend` skill could help here.

You could prepare a clear set of rules of how you would like a specific task to be performed.

- An HTML reference design enforces visual consistency but not structural consistency - look for opportunities to create a reusable component.
- Decide the patterns (where state lives, who calls APIs, how validation works) before generating the second dialog, not the seventh.
- If a view feels wrong, throw it away and regenerate against the new shared components - but only after you have shared components.
- Invest in vocabulary up-front. Each rename sweep is cheap individually but you'll do many of them.
- The assistant will generate visually consistent CSS by inlining the same literal everywhere. It doesn't generate structurally consistent CSS unless you give it the structure first. A CSS constants file could save many sweeps later.

Project #3: RM Connector

Project #3: RM Connector

https://github.com/richardm90/rm-connector-js

An IBM i DB2 connector for Node.js, supporting both remote (mapepire/WebSocket) and native (idb-pconnector/DB2 CLI) backends with connection pooling and management.

This project originally started as a thin layer over the Mapepire JS client to allow more flexibility in creating and managing connection pools but it soon grew into something different as I realised the performance difference between the two Node.js DB2 for i connectors.

Mapepire JS Client

https://github.com/Mapepire-IBMi/mapepire-js

Pure JS client (written in TypeScript) for connecting to Db2 for IBM i.

- A consistent client SDK across languages
- Minimal dependencies
- No native dependencies (for instance, drivers) needed on the client machine
- Communication to the server is done through a single port
- Data is always encrypted
- Can be installed on client machines and on IBM i

If you're connecting from a remote client use this connector.

I wanted to develop from my local client, testing against an IBM i from my local client so I needed to use the Mapepire JS Client.

But I didn't want app performance to suffer on production!

idb-pconnector

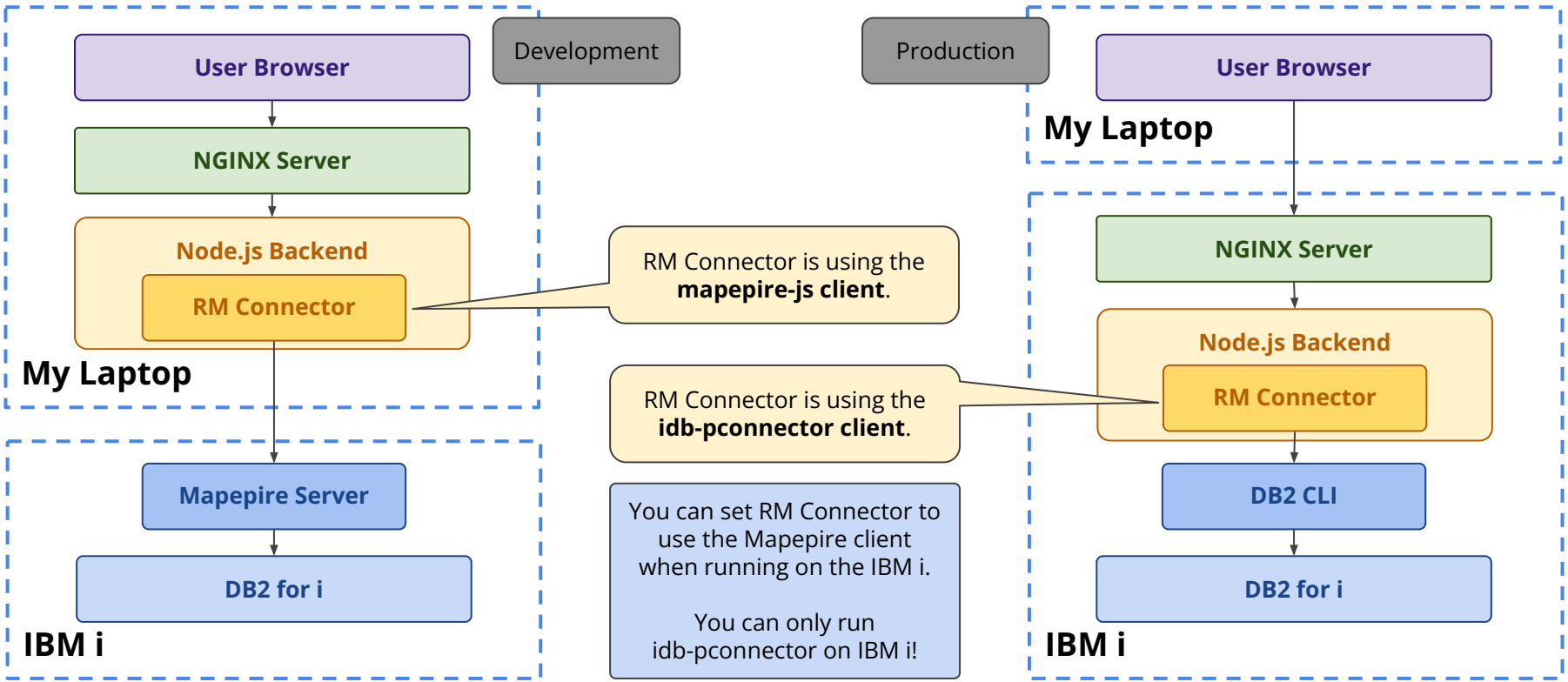
https://github.com/IBM/nodejs-idb-pconnector

Promise-based DB2 Connector for IBM i.

- Connection pooling
- Uses DB2 CLI on the IBM i
- Can only be installed on the IBM i

If you're connecting from an IBM i use this connector.

Project #3: RM Connector Architecture



Project #3: Benefit Highlights #1

This is a project I probably wouldn't have started without a coding assistant - it's just too big of a project for me to undertake on languages and technologies that I wasn't fluent.

Parity testing against two backend connectors!

My Technology Challenges

`idb-pconnector, idb-connector`

- Native DB2 CLI C API

`mapepire-js, mapepire-server`

- Websockets over TCP
- Java server

I cover the basics and have documented the rest.

I could also do the research. How long would it take me?

I now have a suite of parity and performance tests.

I could do all of the things I defer to the assistant but it would take a lot longer.

My Scope Challenges

Identify differences between the two connectors

- Numeric precision
- Date & time format
- Error messages

What do I do about the differences?

- Assistants are great at researching in multiple code languages

You must challenge the assistants findings, especially IBM i specifics.

How do I confidently make changes to my layer?

- Assistants are great at creating test suites

What overhead does my wrapper add?

Project #3: Benefit Highlights #2

Discovered as part of creating the parity test suite.

I discovered some bugs in `idb-connector` and have submitted some fixes!

The assistant helped identify the bug, proposed a fix, helped to write the GitHub issue and PR.

Bug #1 - Fix Boolean Buffer Overflow

Fix a buffer overread in the native addon that caused garbage characters to be appended to string values returned from queries. The bug is most easily reproduced with BOOLEAN columns, where querying a `FALSE` value returns `"FALSEډ"` or similar corrupted data.

https://github.com/IBM/nodejs-idb-connector/pull/191

Bug #2 - Inserting a BLOB Crashes Node.js

When inserting a `Buffer` into a `BLOB`, `BINARY`, or `VARBINARY` column via `bindParameters` (1-D), `bindParametersSync`, or the legacy 2-D `bindParam`, the driver silently overwrites the caller's input `Buffer` after `SQLExecute`. For larger payloads the overwrite also overruns the caller's allocation and the Node.js process dies with SIGSEGV / SIGILL.

https://github.com/IBM/nodejs-idb-connector/issues/202

Project #3: Lessons Learned #1

The assistant can produce some great documentation.

It will, if asked, keep the project document up to date with your changes.

Performance Comparisons

The performance comparison scripts have definitely evolved over recent months and the coding assistant applied changes to the documentation.

Documentation is just as important as your code and requires the same level of attention when reviewing changes suggested by the assistant.

The performance comparison document was a good example of this. It had changed many times over a period of several weeks with sections added out of context with their position in the document.

It required a major restructure!

Mapepire Architecture

When describing how Mapepire handles attaching to a database pool connection the assistant got it wrong. It claimed:

> *"The native mapepire pool also multiplexes, but it dispatches each new query to the first job that's ready. On fast loopback this funnels most of the work onto the earliest connection and leaves the rest underused."*

This isn't true and I don't know how it came to this conclusion.

If something doesn't sound right then challenge the assistant, ask it to double-check the findings.

Vital if you're making decisions based on the findings!

Project #3: Lessons Learned #2

IBM i specifics. Assistants are becoming more “IBM i” aware but they still reach for the more prolific platform terms and ways of working.

Missed During Planning

Not something the assistant (or me) considered up front so didn’t get picked up as part of the design but soon got picked up when testing against an IBM i.

- SQL Naming vs System Naming
- Date and Time Formats
- UTF-8 →CCSID 1208

Not major issues in my case but still extra work after the event.

I requested tests were included to cover UTF-8 data.

Platform Vocabulary

Whenever the assistant uses an industry-generic name (ODBC, JDBC, "the driver"), check whether the IBM i version of that thing actually has a different name

- idb-pconnector was referred to as an ODBC connector
- DB2 Connection = IBM i Job - it can be expensive

Something I didn’t pick up on and proliferated in the docs and code.

Creating new DB connections can be expensive on IBM i.

Q&A

Code Assistants: Broader Implications - The Honest Middle

1. Skill Erosion / Skill Growth

The genuine concern that juniors won't learn fundamentals, balanced against how it actually accelerates learning.

2. Knowledge Transfer and Onboarding

Using AI to ramp up on unfamiliar codebases (huge for IBM i shops with retiring senior devs)

3. The IBM i Training Data Problem

RPGLE, CL, DDS, and DDL aren't well-represented in model training data the way JavaScript or Python are.

4. Where It Shines

Writing tests, documentation, refactoring, explaining unfamiliar code, scaffolding, regex, SQL.

5. Where It Struggles

Hallucinated APIs, confidently wrong RPGLE, over-engineering simple things, the times you spent longer fixing its output than writing it yourself.

6. The Review Discipline

You still own the code. What will your review process look like?

Useful Resources

Useful Resources

My Projects

- RM ACS Launcher
- RM Console
- RM Connector

DB2 for i Connectors

- Mapepire JS client
- Mapepire docs
- idb-pconnector

Code Assistants Mentioned

- IBM Bob
- IBM Bob docs
- Claude Code
- Codex
- GitHub Copilot
- Cursor
- Google Gemini Code Assist

Claude Code Docs

- Memory / CLAUDE.md
- CLAUDE.md vs auto memory
- Skills

Extras: Context Is Everything

Context: CLAUDE.md

Instructions You Write

- Loaded at the start of every conversation
- Markdown file
- Write instructions you'd otherwise re-explain
- Add an instruction here if Claude makes the same mistake twice
- Can exist at multiple scopes
- All discovered files are concatenated into context rather than overriding
- Ordering runs from the filesystem root down to your working directory
- Keep it small, target under 200 lines per file
- `/init` generates a starting `CLAUDE.md`

1. Managed Policy

Organisation wide standards; cannot be overridden.

Location: `C:\Program Files\ClaudeCode\CLAUDE.md` (Windows)

2. User Level

Personal preferences across all projects.

Location: `~/.claude/CLAUDE.md`

3. Project Level

Team shared, committed to source control.

Location: `./CLAUDE.md` or `./.claude/CLAUDE.md`

4. Local Level

Personal, project specific; gitignore this.

Location: `./CLAUDE.local.md`

Context: MEMORY.md

Claude Writes This (auto memory)

- Loaded at the start of every conversation
- Markdown file
- One file per project
- Machine local, not shared
- Lets Claude accumulate knowledge across sessions
- It saves notes for itself as it works: build commands, debugging insights, architecture notes, code style preferences, and workflow habits
- It's selective, it decides what's worth remembering
- You can audit, edit or delete!

Directory Location

Each project gets a directory, derived from the git repository.

Location: `~/.claude/projects/<project>/memory/`

Directory Contents

Contains `MEMORY.md` plus topic files, `MEMORY.md` acts like an index file to the topic files.

- `MEMORY.md`
- `feedback_yajl_addutf16_for_utf8_vars.md`
- `feedback_yajl_gen_function_names.md`

Context: CLAUDE.md vs MEMORY.md

CLAUDE.md

- You write these
- Coding standards
- Workflows
- Architecture

MEMORY.md

- Claude writes these
- Build commands
- Debugging insights
- Discovered preferences

Use CLAUDE.md to guide behavior; let auto memory learn from your corrections without manual effort.

Context: Path Scoped Instructions

.claude/rules/

- You write these
- Project specific is common
- You can have directory specific rules, but this can become tangled - best to keep rules in one place
- Support symlinks for sharing a rules set across projects
- User-level `~/.claude/rules/` for personal cross-project preferences

For task-specific instructions that don't need to be in context all the time, use skills instead, which only load when you invoke them or when Claude determines they're relevant to your prompt.

```
your-repo/
├── .claude/
│   ├── CLAUDE.md
│   └── rules/
│       └── sqlrpgle.md
└── QRPGLESRC/
    └── ...source members...
```

```
---
paths:
  - "**/*.sqlrpgle"
---

# SQLRPGLE Conventions

- Use fully free-format (**free) for new source members
- Prefix host variables with `:` inside SQL e.g. `:url`
- After SQL errors, capture `SQLCODE`, `SQLSTATE`
- Put reusable logic in subprocedures
- ...your rules...
```

Context: SKILL.md (1/2)

On-Demand Packaged Workflows

- You write these
- A skill is a `SKILL.md` file, optionally with supporting files
- The `description` field used by the model during planning to decide whether the skill matches the current task
- Can exist at multiple scopes
- When skills share the same name across levels, organisation overrides personal, and personal overrides project.

Skills only load when you invoke them or when Claude determines they're relevant to your prompt.

1. Organisation Level

Organisation wide.

Location: via managed settings

2. Plugin Level

Plugins are distributable bundles that can be packaged and shared.

Where the plugin is enabled.

Location: `<plugin>/skills/<skill-name>/SKILL.md`

3. Project Level

Applies to this project only.

Location: `.claude/skills/<skill-name>/SKILL.md`

4. Personal Level

Applies to all of your projects.

Location: `~/.claude/skills/<skill-name>/SKILL.md`

Context: SKILL.md (2/2)

On-Demand Packaged Workflows

- Supporting files live in the same skill directory, alongside `SKILL.md`
- This keeps `SKILL.md` focused whilst allowing Claude access detailed reference material only when needed - these don't need to be loaded into context every time the skill runs
- Claude only knows about supporting files if `SKILL.md` references them
- Keep `SKILL.md` under 500 lines, move detailed reference material to separate files
- Once a skill loads it remains in context

Skills only load when you invoke them or when Claude determines they're relevant to your prompt.

```
rpg-fixed-to-free/
├── SKILL.md      # required — overview and navigation
├── reference.md  # detailed docs — loaded only when needed
├── examples.md   # usage examples — loaded only when needed
└── scripts/
    └── helper.py # utility script — executed, not loaded
```

```
---
name: rpg-fixed-to-free
description: "Convert fixed-format RPG IV (and RPG III / RPG400)
source to modern fully free-format **FREE RPGLE. Use this skill
whenever ...
---

# Fixed-format to free-format RPG conversion

## Overview

This skill converts legacy fixed-format RPG IV (and where relevant RPG
III / RPG400) into modern fully free-format `**FREE` RPGLE ...

## Workflow

1. **Identify the source vintage.**
...
```

Context: Decision Tree

Should be in context every session, you write it → CLAUDE.md (scoped by directory level)

Same as above but only for certain file types/paths → `.claude/rules/` with `paths` frontmatter

A procedure Claude pulls in only when relevant → skill

Let Claude remember its own learnings → auto memory (just leave it on)

Code Assistant Terms: Conversation vs Session

Broadly the same but not interchangeable.

Conversation

- A conversation is the dialogue itself
- The back-and-forth turns between you and the assistant.
- It's what you'd see if you printed out the chat transcript.

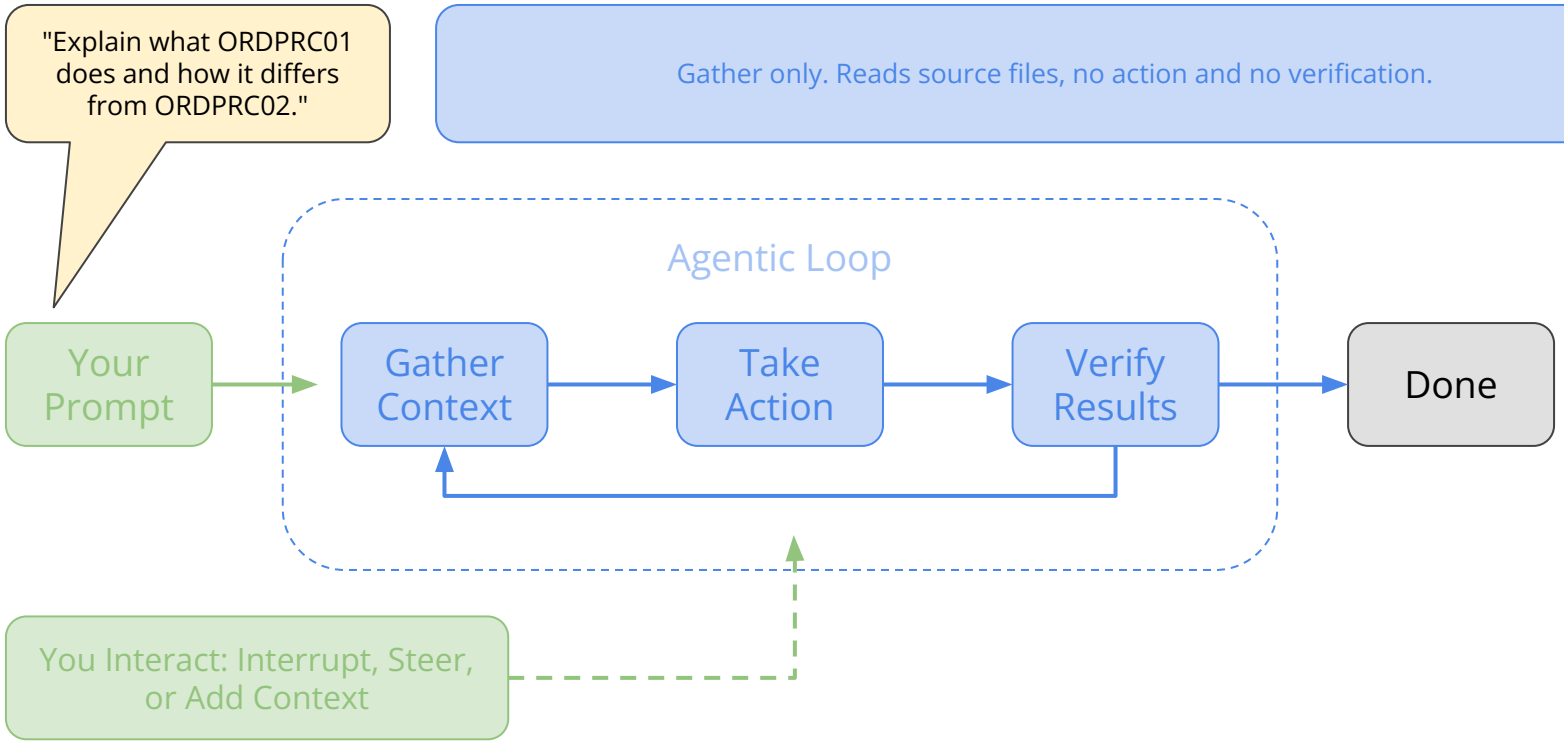
Session

- A session is the broader unit of work the code assistant manages.
- It contains the conversation, but also ...
  - the working directory
  - the loaded project instructions
  - the active skills
  - the cumulative context window
  - the tool-call history
  - and the resumable state on disk

Extras: The Agentic Loop

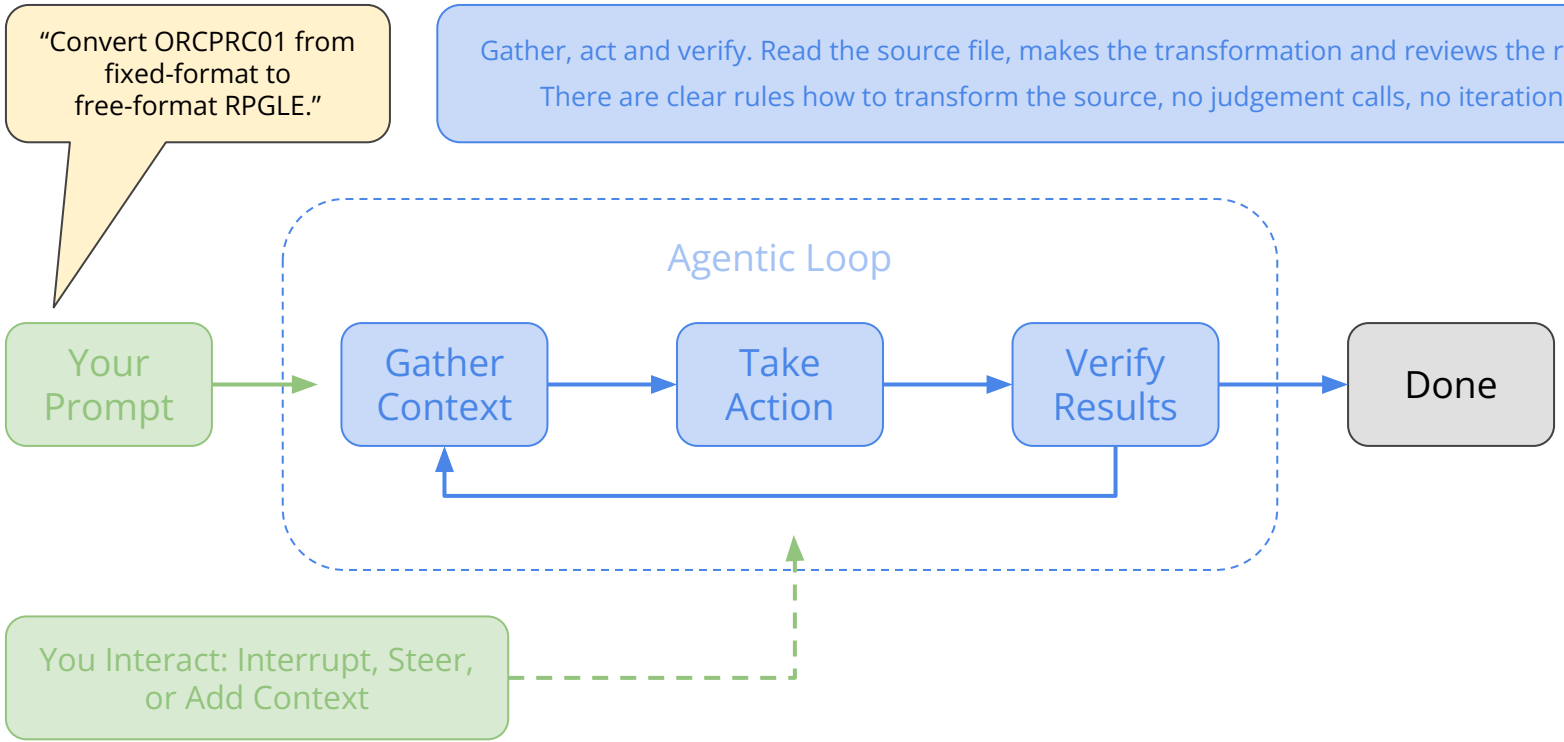
How Code Assistants Work: The Agentic Loop

Read Only Question Example



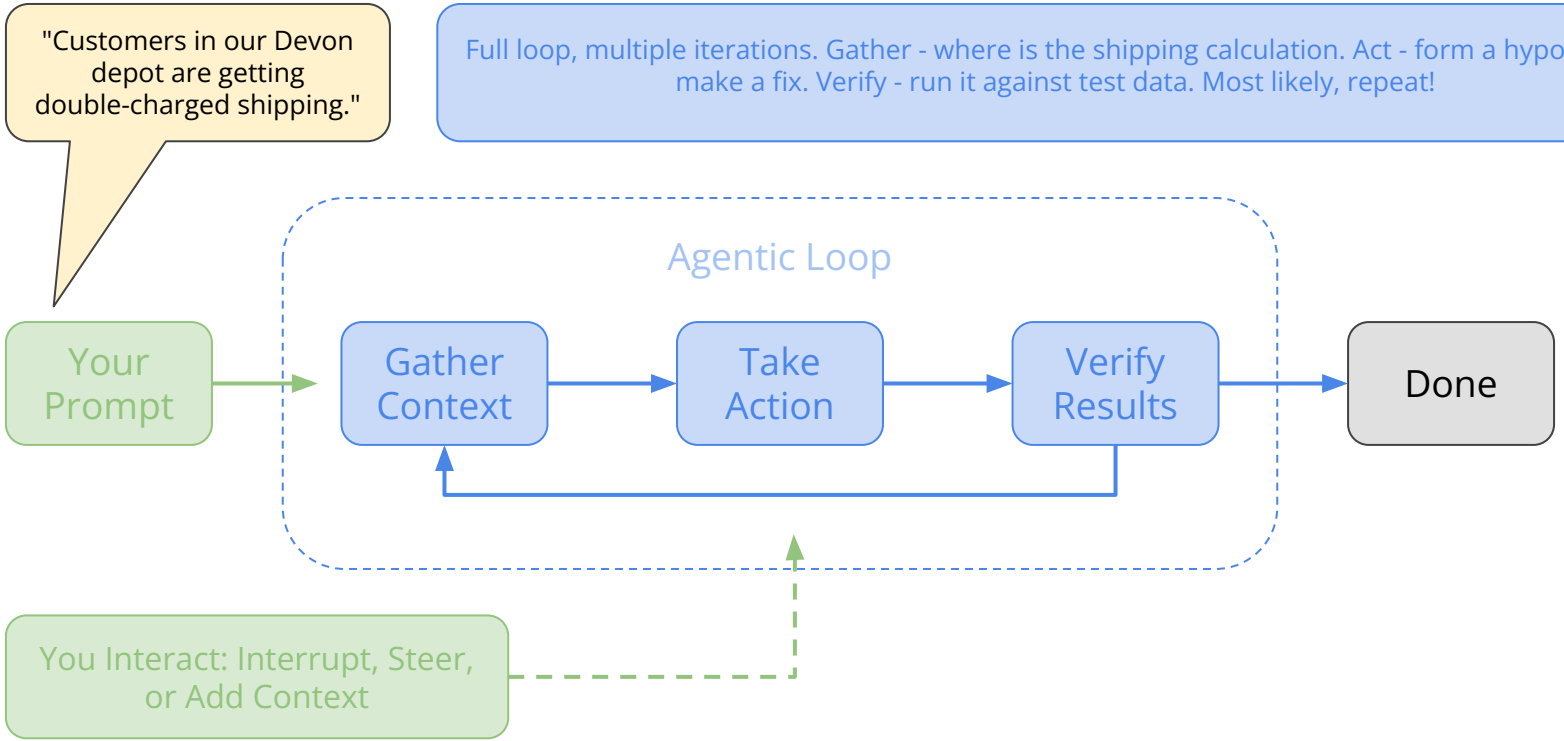
How Code Assistants Work: The Agentic Loop

Mechanical Edit



How Code Assistants Work: The Agentic Loop

Bug Fix



How Code Assistants Work: The Agentic Loop

Refactor

"Extract the reusable code from the monolithic ORDPRC01 RPGLE program into service programs."

Verification heavy. Each individual change might be small but every change risks breaking something so verification dominates. Heavily leans into regression tests and reading callers!

Do you have automated tests?