

AI-DRIVEN DEVELOPMENT

Creating an 'AI Developer' who works as part of your Team

An implementation that integrates with your processes, tools and practices.

February, 2026

TL;DR

AI tools for development have evolved hugely in the past year. From prompting and getting it to write code, the introduction of agents, MCP, skills, increased context windows, and generally more capable LLMs has changed the game completely.

From playing with AI and creating a few complete systems and tools, a year ago, I wanted to see if I can create an ‘AI Developer’ that has moved up the development stack, handling the process of delivery from requirement to deployment, while adhering to a defined development, or SDLC, process. The human-in-the-loop is there for review and approval only and not coding at all. To reflect the current reality in most development teams, using GitHub, Jira and Confluence, by our ‘AI buddy’ is essential.

To do this, Using Claude Code, I created a suite of agents and skills, with a SDLC coordinator agent which provides the workflow, delegates to other expert agents/skills, defines the quality gates to be passed, controls on what AI is allowed to do, where humans are needed and what external tools in the environment can be interacted with and with what permissions. With an existing full-stack codebase, providing an invoicing system, with basic login capabilities, I decided to throw the high-level request to ‘add configurable MFA support’. This, I figured, has a nice mix of moving parts on the front-end and back-end, while having security considerations to consider.

With my environment set up, my AI development colleague, took my request, turned it into a specification, broke it into clear tasks, did the implementation, with full reviewing, testing and debugging, and pushed out a working MFA feature, including configuration and user login. My part was being asked to review and approve things at various stages, typically in Jira or GitHub, which I happily did.

AI worked with Jira and GitHub as a developer might do, creating epics and stories, updating them with comments, adding estimates, committing code, triggering PRs, creating feature branches, and more. The hope was that the result of this addition, to an existing codebase, would include sufficient history, and documentation, in the expected places, to allow any developer to update or support the feature later.

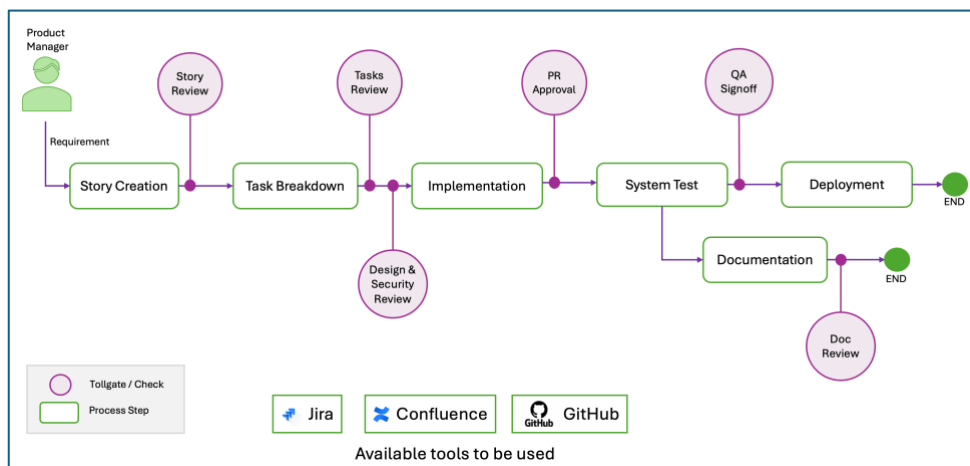


Figure 1- Development Process

While I did need to get involved once or twice along the way, this was most likely due to my definition of the skills/agents, and some misconfiguration of integrations. However, from the high-level request, it did essentially deliver as requested.

To me, it showed that with basic flows, skills and configuration, with integrations and tight controls on approvals, it is possible to introduce an AI developer into the full workflow as a contributor, or collaborator, working alongside the existing team. My request was not very complex, but it did show my AI engineer involved, in a controlled manner, at all stages, and producing really good quality, from specs to stories to code to tests. The delivered feature was good too, fitting into the overall tone of the existing system.

It did require that the overseeing developer (me in this case) had a decent understanding of the system, domain, programming language, good practices, AI capabilities and challenges, to be able to review, assess and accept its work. I am always suspicious about what the LLMs deliver, but as they prove themselves, I will likely apply more trust and be less picky on the reviews, however, ... not sure this is good or bad!

The document provides more of the details, some examples on skills and agents, and some outputs and conclusions

Introduction

Depending on where you read, the AI capabilities today are going to make all developers unemployed, however, I think it will likely negatively impact developers who don't embrace AI's capabilities and 'move up the stack'.

In a previous article, I wanted to see what was possible to create, using AI as a chat/tool in an IDE. This, at the time was effective, and I developed both native and web software, which I am still using today. I realised back then, it did require a fair level of technical awareness, understanding, guidance and interaction on my part – so no free lunches!

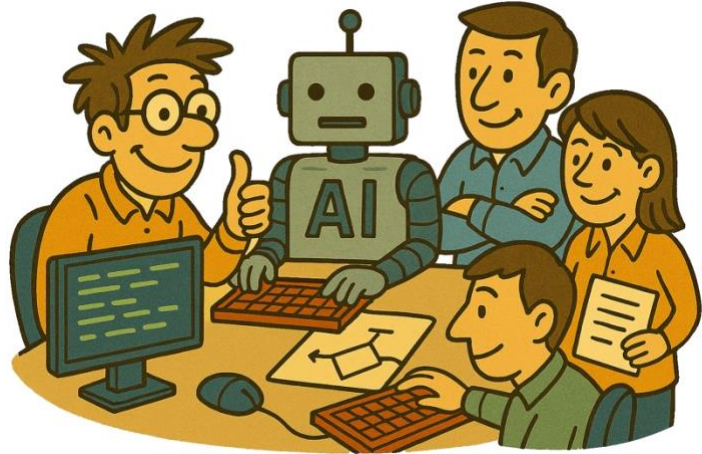
Since then, LLM/AI capabilities have increased massively. The evolution of AI for development has moved from using AI as a tool to AI becoming a system, where a controlled, structured process can be built around it. With the support for MCP, plugins, skills, agents and a variety of other additions which have appeared, the limitations associated with the tools 9-12 months ago have evaporated quite substantially.

Effective software engineering employs a development process, or SDLC (software development lifecycle), that provides structure and control over all aspects of the delivery, from requirements definition to the deployment to production. It defines the steps, approvals, use of tools, documentation, etc. The development process is something that has evolved over time and is embedded in the organisation, proven and adhered to by all involved, so if AI is to be introduced in a more advanced capacity, it needs to be done without breaking what is already there. If changes are needed to the process, they should have clear benefits and rationale behind them.

I have been working with multiple development organisations, using AI tools on an individual basis to increase effectiveness, coding, testing, apply checks, etc. The lever of usage is often dependent on the interest and skill of the individual developer, so few consistencies in general. The challenge of moving up the value chain, and letting AI loose from requirement to deployment, as an autonomous 'developer, as part of an existing team, is another matter completely. Design and creativity, amongst other skills come into the picture.

So, having said that ...

... my latest adventure is to see how I can deploy the latest LLM/AI development tools to become more like and 'AI team member' of an existing development team. It shall follow a defined process, work according to rules, security requirements, approval demands and more and do this autonomously driving itself, with the human-in-the-loop being there to review and approve actions only.



The Goal

So, my goal was clear:

To see if an AI system, behaving like a senior developer, could take a high-level requirement, for an update to an existing software system, and fully implement it, effectively and with quality, using best industry practices and in accordance with a defined development process workflow.

The Playing Field

To make things a little more realistic, I am adding my 'AI team member' to work on a new feature in an existing code base. The system is a 'React-FastAPI-Python-Postgres' application that supports a receipt and invoice registration function for a small company. It incorporates a multi-role login approach, support for uploading and reporting on invoices, with AI used to extract the details from each invoice or receipt.

I have a clearly defined development process in place, with GitHub, Jira and Confluence being used by the team who developed the system originally. Any development that is to be done will follow this process strictly.

I am going to ask my new team member to take a high-level requirement, specify, design, implement, test, review and deploy all in adherence with the defined development process.

The System Being Updated

Dealing with invoices and receipts, the key aspect of interest in this project is its basic username and password login mechanism.

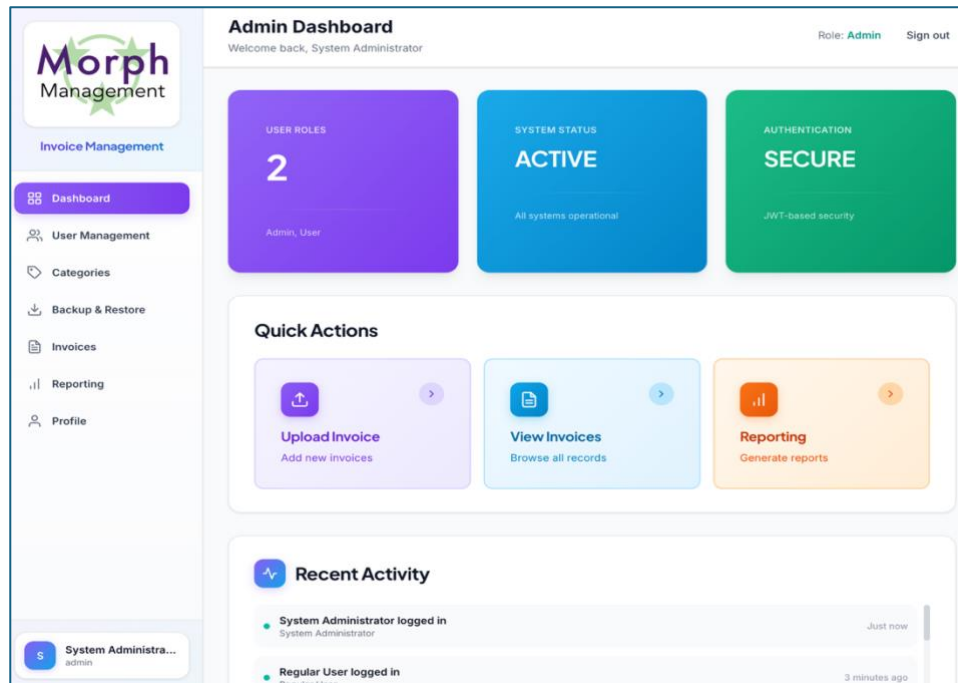


Figure 2- The Dashboard for an Administrator

User management is done by an admin. Currently the password is set for the user when they are created, and I have not provided the capability to change it yet ☹️. I wanted to provide the user with an additional personalized approach to enhanced security and be able to use their phone to provide a one-time token in addition to their username/password authentication. This should be done using a typical authenticator app, but I'll let my AI friend figure out how to do this effectively and propose a solution to me, as part of its job.

The Environment

I am using Claude Code, with Opus 4.5 for the initial specification work, going from the initial requirement to a set of stories that implement the requirement. Opus 4.5 is good at the planning and creatively arriving at the solution. However, it is expensive when you get into the coding phase, and I have run out of tokens too often, so adopted a dual approach.

For the implementation phase, I switched to GLM 4.7 by Z.AI, a model which is fast and effective at coding, testing and the down-and-dirty development parts. They claim it is comparable on SWE metrics with Opus 4.5 so it should be good. I chose this as they offer 15x the token limit versus Opus 4.5 at a lower price than Claude Pro, and it integrates seamlessly into the Claude Code environment, with only a handful of environment variables needing to be changed.

As part of the development process, all code is stored in GitHub, all work is carried out from Jira work items, with epics, stories, bugs, etc. all documented there. Confluence is used for documentation, when more is needed than Jira can effectively provide.

Claude Code has ‘easy-to-integrate’ MCP Servers to support interaction with these systems, so I was ready to go. To allow my AI friend to be able to do full testing, I installed the ‘chrome-devtools’ plugin, so it could control the browser too, to test the UI components.

The Development Process

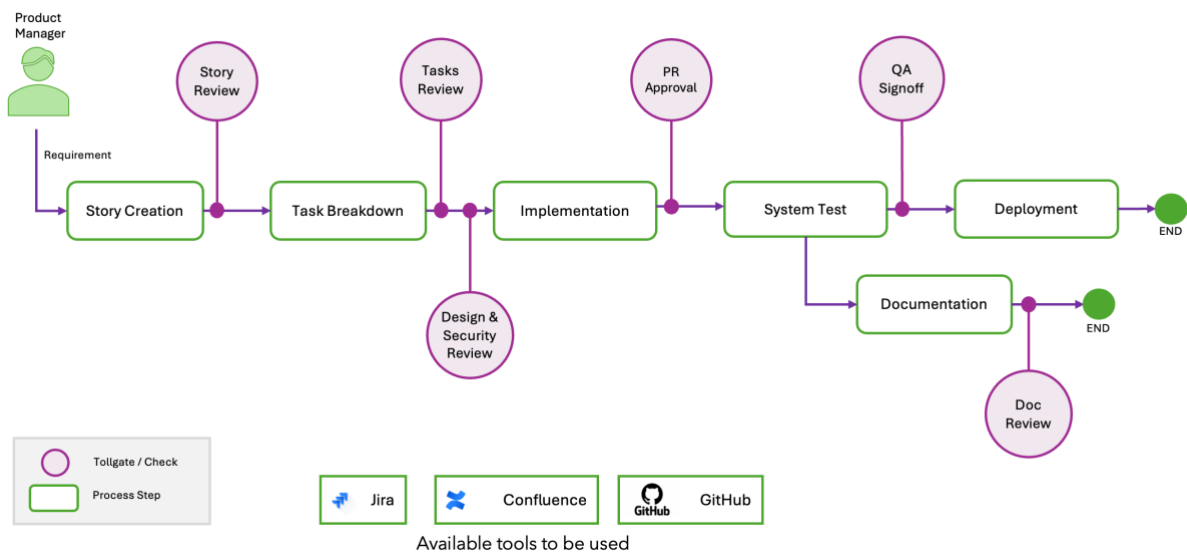


Figure 3 - The Development Process

The development process to be supported has clear steps to be carried out, and checkpoints along the way where reviews and approvals are needed. In the human only workflow, these are done with updates to Jira tickets, or approval steps in GitHub flows.

I want to replicate the same approach for my AI developer, setting the same requirements on it, and ensuring that the human review and approvals are enforced in the same manner.

To implement the development process, with its steps and controls, that the AI can understand, I created an agent called **sdlc-coordinator**. This has, a supporting crew of agents which deal with the various jobs needing to be carried out when moving from the initial requirement to the delivered, tested and in deployed update.

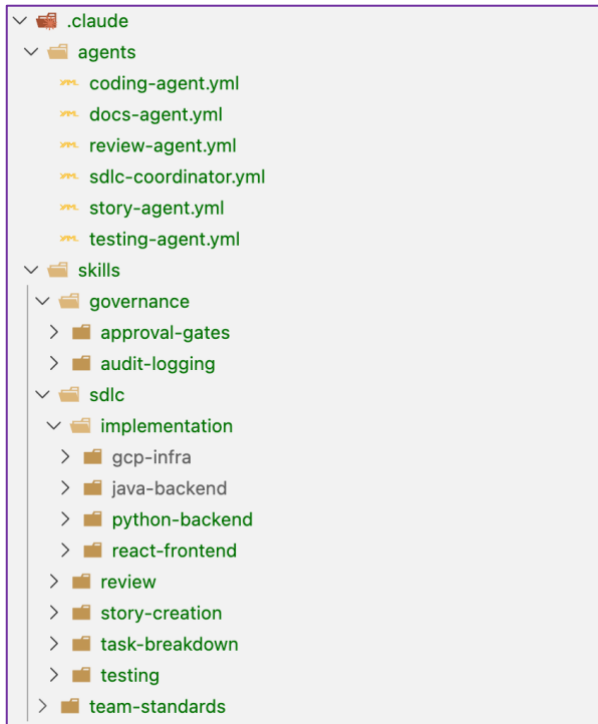


Figure 4 - Claude files representing with agents and skills doing the heavy lifting

As you can see in the **sdlc-coordinator** agent definition below, the parameters for the effective workflow execution are stated in a structured manner. The details include:

- The description of what the coordinator is to do
- The skills that are available for the different steps that it is coordinating
- How delegation is carried out to the other agents
- The tools to be used – e.g. GitHub, Jira, Confluence
- The approval gates and steps that are to be followed, including how approval is done, if needed.
- As the flow will interact with external tools where there is precious information, I indicate what operations it can carry out itself, and what needs approval.
- Similarly, it can send external notifications based on success, or other events arising during the process.
- Additional settings deal with how long the context is available, for the workflow, as the individual steps may be paused, fail, etc. I also include additional relevant content that should be considered and general operational issues like logging.

```

# sdlc-coordinator.yml
name: sdlc-coordinator
version: "1.0"
description: |
  Orchestrates the complete SDLC workflow from story creation to deployment.
  Enforces approval gates and maintains context across all stages.

# Skills this agent can use directly (without delegation)
skills:
  - governance/approval-gates
  - governance/audit-logging
  - team-standards/coding-standards
  - team-standards/documentation

# Delegation configuration - when to delegate to specialized agents
delegation: ...

# MCP servers required
mcp_servers:
  required:
    - jira
    - github
  optional:
    - confluence
    - slack

# Approval gates this agent enforces
approval_gates:
  G-010:
    name: STORY_DRAFT_REVIEW
    auto_approve: false
  G-020:
    name: TASK_BREAKDOWN_REVIEW
    auto_approve: false
  G-030:
    name: DESIGN_REVIEW
    auto_approve: false
    condition: "changes.architecture_impact == true"
  G-035:
    name: SECURITY_REVIEW
    auto_approve: false
    condition: "changes.security_sensitive == true"
  G-050:
    name: PR_APPROVAL
    auto_approve: false
  G-060:
    name: QA_SIGNOFF
    auto_approve: false
  G-090:
    name: DOCUMENTATION_REVIEW
    auto_approve: false

# Behavior configuration
behavior: ...

# Notification configuration
notifications: ...

# Context preservation
context: ...

# Logging configuration
logging:
  level: INFO
  audit_all_actions: true
  include_prompts: false # Privacy: don't log full prompts
  retention_days: 90

```

Figure 5 - sdlc coordinator agent with the section headings (lots more detail under!)

The flow moves from step to step, based on approval by the human. You can see the gates indicated in Figure 5 above. The details of what is done at each step is included in the skill definition and this includes updating Jira and GitHub appropriately.

The skills, as they are defined, result in a very interactive experience for the user, with options being offered in terms of moving through the flow placing blockers in situations if an official approval is not executed.

The story creation flow, as shown below, provides very clear details, constraints and instructions which drive the behaviour we want.

```

# Story Creation Skill

## Purpose
Assists in creating well-structured user stories with comprehensive acceptance criteria, proper context from existing documentation, and alignment with team standards.

## When This Skill Activates
- User mentions creating a new story, feature, or requirement
- Commands like `create story`, `new feature`, `write user story`
- Working with product requirements or feature requests
- References to Jira story creation

## Required MCP Servers
- **Atlassian MCP**: For Jira story creation and Confluence document access
- **GitHub MCP**: For linking to existing code patterns (optional)

## Workflow Overview
...

```

```

graph LR
    PM[Product Manager] --> CD[Claude Drafts]
    CD --> ER[Engineer Reviews]
    ER --> JC[Jira Created]
    
```

```

...

## Role Involvement
> ### Product Manager (PM) ...
> ### Tech Lead / Architect ...
> ### Engineer ...
> ### QA Engineer ...
> ## Process Steps ...
> ## Output Format ...
> ## Trigger Behaviors ...
> ## Integration Points ...
> ## Examples ...

## Quality Checklist
Before presenting draft, verify:
- [ ] User story follows "As a... I want... So that..." format
- [ ] At least 3 acceptance criteria defined
- [ ] Each AC uses Given/When/Then format
- [ ] Technical considerations documented
- [ ] Dependencies identified and linked
- [ ] Appropriate labels and components set
- [ ] Story point suggestion provided with rationale

```

Figure 6 - Story Creation Skill with main sections (lots of details on the roles, steps, behaviours, and some example output formats)

Under each of the headings in this skill there is substantially more information provided to ensure that the agent implementing the skill knows exactly how to behave.

The last section in the skill includes quality controls which force the model to include sufficient clarity and rationale for the proposal and highlight any dependencies. This is key, both for the review process to let it proceed, but also useful as documentation for anyone needing to work on the code in future.

With the code cloned and VS Code up and running, and Claude Code ready to serve, I kicked off the process, by running `@sdlc-coordinator` which got the agent going.

It presented a clear view of what it is proposing to do, and the workflow, and then I kicked it off.

```

Current Setup Summary

Agents Available:
- sdlc-coordinator - Full workflow orchestration
- coding-agent - Implementation assistance
- docs-agent - Documentation management
- review-agent - PR/code review
- story-agent - Story creation and refinement
- testing-agent - Test generation

Skills by Phase:

```

Phase	Skill
Story Creation	<code>sdlc/story-creation</code>
Task Breakdown	<code>sdlc/task-breakdown</code>
Implementation	<code>sdlc/implementation/*</code> (React, Java, Python, GCP)
Testing	<code>sdlc/testing</code>
Code Review	<code>sdlc/code-review</code>

```

Approval Gates:
- G-010 STORY_DRAFT_REVIEW
- G-020 TASK_BREAKDOWN_REVIEW
- G-030 DESIGN_REVIEW (conditional)
- G-035 SECURITY_REVIEW (conditional)
- G-050 PR_APPROVAL
- G-060 QA_SIGNOFF
- G-090 DOCUMENTATION_REVIEW

---
What would you like to do?

1. Start a new feature - Create a story, break down tasks, begin implementation
2. Review current status - Check what's in progress, see pending gates
3. Configure MCP servers - Set up Jira/GitHub/Confluence integrations
4. Customize the workflow - Adjust gates, skills, or behaviors for the calculator project
5. Run a specific SDLC phase - Execute story-creation, task-breakdown, testing, etc.

Let me know what you'd like to focus on!

> Start a new calculator feature

```

At the prompt, I entered the following one and only prompt, and the game was on:

- Can you extend the login mechanism in the system, with an MFA capability. The user shall be able to enable or disable this themselves when they are logged in.

Over the subsequent 75 minutes it progressed through the various steps.

Initially it created an Epic in Jira, describing the overall requirement well.

It asked me to review it, and when I was OK with it, it proceeded to creating 8 stories which were equally clear and detailed, supporting the implementation.

Implement Configurable Multi-Factor Authentication (MFA)

Description

Goal
Add configurable multi-factor authentication (MFA) to the Morph Invoice platform, allowing users to secure their accounts with a second authentication factor. Admins should be able to enforce MFA policies per company.

Background
Morph Invoice currently uses credential-based authentication via NextAuth.js. As the platform handles sensitive financial data (invoices, supplier details, monetary amounts), adding MFA strengthens security posture and supports compliance requirements. The system already has multi-company support with role-based access (ADMIN/USER), which MFA policies should integrate with.

Scope

In Scope

- **TOTP-based MFA** (Time-based One-Time Password) - compatible with Google Authenticator, Authy, Microsoft Authenticator
- **MFA setup wizard** - QR code generation, secret key display, verification step
- **Recovery codes** - One-time backup codes for account recovery when authenticator is unavailable
- **Admin MFA enforcement** - Company-level policy to require/optional MFA for all users
- **User self-service** - Enable/disable MFA from user profile (when not enforced)
- **Remember device option** - Skip MFA on trusted devices for configurable duration
- **MFA status visibility** - Admin dashboard showing MFA adoption per company
- **Audit logging** - Track MFA events (enabled, disabled, verified, failed, recovery code used)

Out of Scope (Future Consideration)

- SMS-based MFA
- WebAuthn/FIDO2 hardware keys
- Email-based OTP
- SSO/SAML integration

Success Metrics

- All admin users can enable TOTP-based MFA on their accounts
- Company admins can enforce MFA for all users in their company
- MFA verification adds less than 3 seconds to the login flow
- Recovery codes allow account access when authenticator is unavailable
- All MFA events are captured in the activity log

Technical Considerations

- Integrate with existing NextAuth.js authentication flow

Figure 7 - Epic created in Jira for the overall project

MFA: Database schema and backend foundation

Description

User Story
As a **developer**, I need the database schema and core backend services for MFA so that subsequent stories have a solid foundation to build on.

Acceptance Criteria

- New Prisma models created:
 - `MfaSettings` - `user_id`, `is_enabled`, `totp_secret` (encrypted), `totp_verified`, `enabled_at`, `updated_at`
 - `RecoveryCode` - `user_id`, `code_hash`, `is_used`, `used_at`, `created_at`
 - `TrustedDevice` - `user_id`, `device_token_hash`, `device_name`, `user_agent`, `expires_at`, `created_at`
- Database migration runs cleanly against existing schema
- TOTP service implemented:
 - Generate TOTP secret
 - Generate provisioning URI (for QR codes)
 - Verify TOTP token with time window tolerance
- Encryption utility for storing TOTP secrets at rest (AES-256-GCM)
- Rate limiting utility for MFA verification attempts (max 5 attempts per 5 minutes)
- All new models have appropriate indexes for query performance
- Unit tests for TOTP generation/verification and encryption utilities

Technical Notes

- Use a well-established TOTP library (e.g., `otplib` or `speakeasy`)
- TOTP secrets must never be stored in plaintext - use AES-256-GCM with a server-side key from environment variable
- Recovery codes should be hashed with `bcrypt` before storage
- Consider adding `mfa_enabled` boolean flag to User model for quick lookups
- Prisma migration should be backward-compatible (all new fields optional or with defaults)

Dependencies

- None (foundational story)

Story Points: 5

Figure 8 - One of the 8 stories created in Jira

With the list of stories created, and reviewed and approved by me, it proposed to attack the implementation. It didn't break these down further into further tasks, though I am could have required this too, but didn't. The project was in Jira and set up for Kanban, as opposed to SCRUM, and the 8 stories were the direct basis for implementation.

With the 8 stories, it proposed the order in which they should be implemented, and which could be run in parallel. With each story it broke it down for me into the various actions it would take, allowing me to review and approve.

During the process, it interacted extensively with Jira and GitHub. In GitHub it created branches on which it built the stories, with each branch covering one or more stories, if they could be handled in parallel.

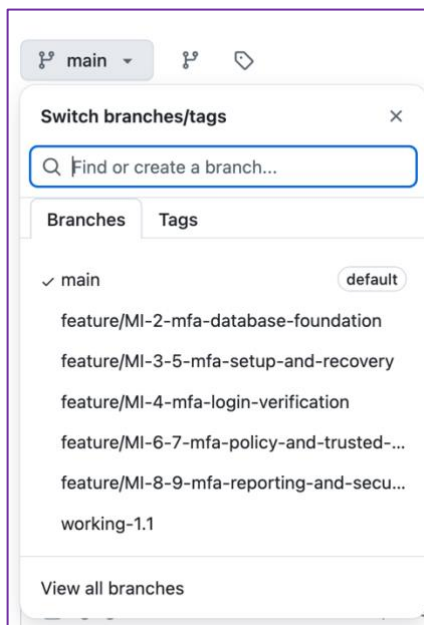


Figure 9 - The feature branches created in GitHub to isolate them until fully tested & approved.

After implementing each story, it created a substantial set of tests. It worked repeatedly testing/debugging until they all worked and then presented details of the outcome for that feature implementation.

With the story complete and passing all of the tests, the workflow:

- committed the code to GitHub in the feature branch.
- triggered a pull request (PR) indicating that it needed to review & approved.
- approved the PR after I reviewed it and approved it,
- added a comment to the story in Jira with the review details on the PR, taken from GitHub,
- added an additional comment which included a very detailed description to the indicating what was implemented, tests run, and relevant useful information.
- with my approval, merged the feature branch into the master branch.

This was repeated 5 or 6 times, with each feature group (if they could be implemented in parallel) implemented, tested and run through the process, above.

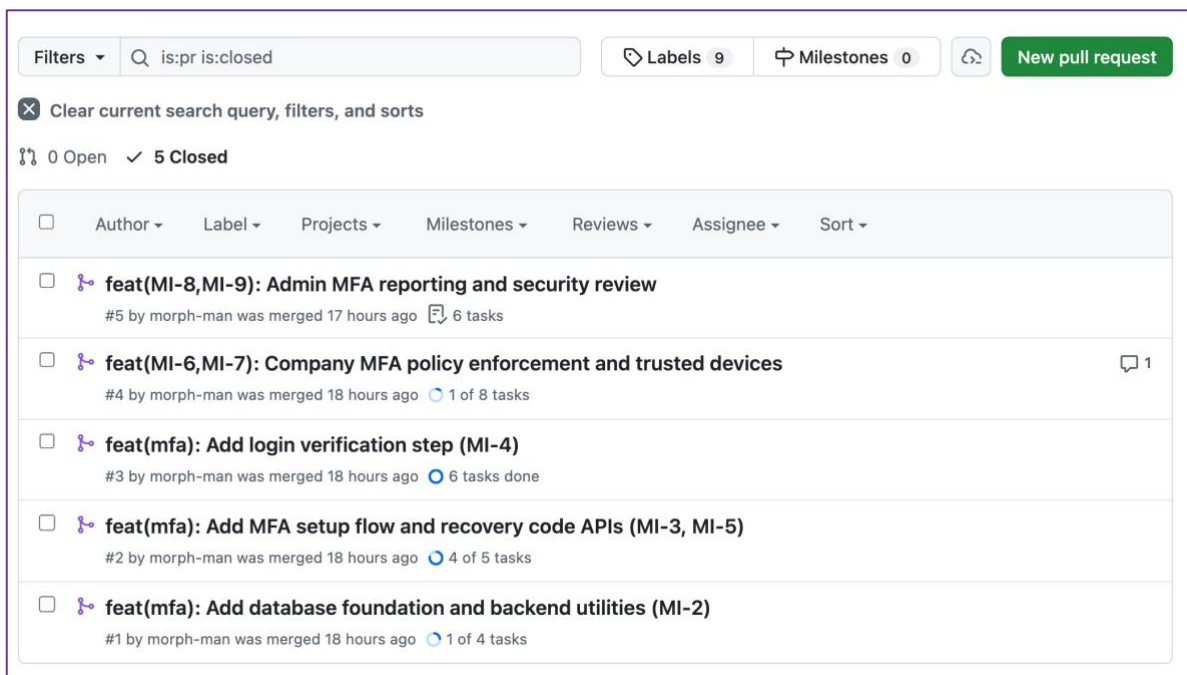


Figure 10 - The PRs in GitHub that were created, which I reviewed and approved during the session.

As it had planned a system testing phase itself, it created a large suite of automated tests, which it worked on until all executed successfully.

Finally, after I had reviewed and approved when asked, it declared itself finished, giving me a very detailed report on the implementation, testing, etc.

I realised that there was also no automatic code review of the code executed as part of the flow – I need to check this - so I decided that I wanted one, and I triggered the code-review skill, where it found a list of issues it believed should be addressed. See some details below.

It added these to Jira as issues to be done, and after checking through them, I indicated that I wanted just the critical & high severity ones addressed.

It proceeded again with its implementation + testing phases and updated GitHub and Jira as I had expected. After all this, Jira had only the less severe work items to do, with the more critical ones resolved.

When it completed testing and updating, it was ready to deploy.

I realised that I had not included a step to deploy the total solution to Docker, after it was QA approved, so I asked it to do this. As the system had previously been deployed in Docker, all the various scripts and flows were in place for this, so a simple request to Claude to deploy it, and worked without any further interaction.

Reviewing the MFA implementation changes made to replace middleware-based enforcement with page-level checks.

```

---
CRITICAL Issues

1. Security: Race Condition in MfaProtection Component
File: app/src/components/auth/MfaProtection.tsx:69-74

Problem: The isChecking state is set to false BEFORE the redirect happens. This creates a window where the protected content is briefly rendered before the redirect occurs.

setIsChecking(false); // Line 69 - Content renders immediately after this
// If MFA is enabled but not verified, redirect to MFA verification
if (!mfaVerified) {
  router.push('/auth/mfa-verify'); // Redirect happens AFTER content renders
}

Impact: Protected content may flash briefly before redirect, potentially exposing sensitive data.
Fix: Move setIsChecking(false) inside the if (!mfaVerified) block or keep it true until after redirect logic.

---

2. Security: Inconsistent Cookie Secure Flag Logic
Files:
- app/src/app/api/auth/mfa/verify/route.ts:114-131
- app/src/app/api/auth/mfa/trust-device/route.ts:93-99

Problem: Two different approaches to determining HTTPS:
- verify/route.ts checks x-forwarded-proto, req.url, and req.nextUrl.protocol
- trust-device/route.ts only checks process.env.NODE_ENV === 'production'

The trust-device endpoint will set secure: true in production even if accessed over HTTP, causing the cookie to fail.

Impact: Trusted device cookie won't work correctly, users won't be able to use "remember device" feature.
Fix: Apply the same HTTPS detection logic to trust-device/route.ts:

const isHttps = req.headers.get('x-forwarded-proto') === 'https' ||
  req.url.startsWith('https://') ||
  req.nextUrl.protocol === 'https:';

---

3. Security: Router.pathname Check May Not Work in App Router
File: app/src/components/auth/MfaProtection.tsx:32

Problem: In Next.js App Router, router.pathname doesn't exist. It should be usePathname() hook instead.

if (router.pathname === '/auth/mfa-verify' || router.pathname === '/auth/signin') {

Impact: The check to skip MFA verification on those pages likely fails, causing unnecessary checks or potential issues.
Fix: Use usePathname() from next/navigation:

import { usePathname } from 'next/navigation';
const pathname = usePathname();
// Then use pathname instead of router.pathname

---

HIGH Issues

4. Performance: Duplicate Database Query in Session Callback
File: app/src/lib/auth.ts:144-155

Problem: The session callback makes an additional database query to fetch mfa_enabled, but this data was already available in the JWT callback at line 84-93.

// JWT callback already fetches this (line 88)
include: { mfaSettings: true }

// Session callback fetches it again (line 146)
const dbUser = await prisma.user.findUnique({
  where: { id: token.sub },
  select: { mfa_enabled: true },
});

Impact: Every session refresh (which happens frequently) triggers an unnecessary database query.
Fix: Add mfaEnabled to the JWT token so it's available in the session callback without an extra query.

---

```

Figure 11 - Part of the output from the code-review

So, what did it deliver?

The requirement was to provide MFA for authentication to the system.

It delivered, the functionality, although there were a few issues that did crop up that needed a little interaction and it needed to debug and do more testing to resolve them, which it did.

However, my awareness, with occasional feedback, being able to follow what was happening, checking error logs, and understanding technical choices being made helped it proceed effectively. It had a problem with the browser controlling MCP, however, to be honest, I think it was a configuration issues on my side that was the source, so I cannot really blame it. When I resolved this, it was able to run its additional tests.

In terms of what was delivered, check out the screenshots below.

It provided the full capability, with management of the user access to MFA, enabling and configuration of MFA for a user, allowing them to scan the QR code with one of a choice of mobile apps and successfully be logged in.

The logic and integration of the feature into the codebase is good and worked pretty much as intended. The details it added in Jira and GitHub were top quality and helpful for a developer needing to work with the code later.

I was offered the option of generating documentation, but declined, this would have created appropriate content and added it to Confluence. I had not this explicitly included in the workflow, so another improvement I could have made.

The following are the screen shots and popups which enable and onboard the user for MFA.

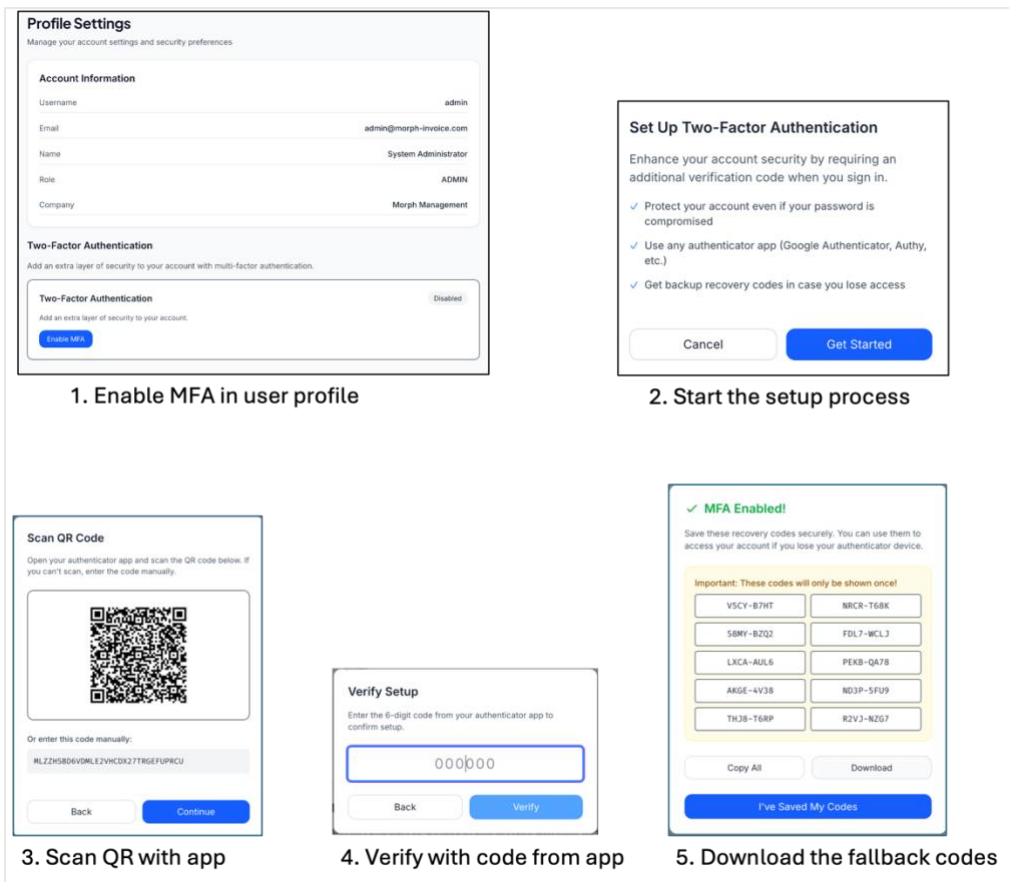


Figure 12 - Setting up the MFA

Then, when they subsequently log in, they are prompted with the following popup, after they enter their username and password.

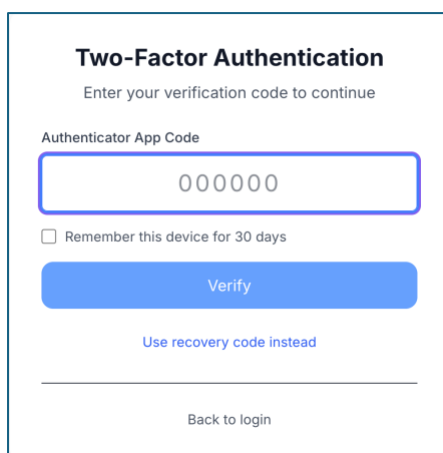


Figure 13 - The MFA login dialog

Conclusions

While there were some occasional challenges and things I needed to look at a little more closely to resolve, I was hugely impressed with how this worked.

Do I think that my AI colleague did its job, taking the high-level requirement and delivering a working feature, with quality? And did I achieve my goal?

I think I have to say yes, especially when considering it was his first day on the job!

Some immediate thoughts after the experience:

- The whole area is new, and getting the pieces to work as intended is not always straight forward. It requires trial and error, creating the playbook as we go. There are clearly improvements that are possible to the agents and skills to refine this.
- The human-in-the-loop, who understands the codebase, the requirement, the domain and how the AI system works is essential, as it needs supervision, reviewing, and the controls need to be in place. While this development ‘did not go off the rails’ at any point, it is essential to really scope the capabilities the AI is allowed tightly and still check, validate and check again.
- While using the ‘plan’ mode in AI for development is the norm nowadays, the use of the sdlc coordinator approach, brought it up a level and allowed me to offload much more of the work to AI, handling it in a structured, effective manner. Breaking things into steps, even if the AI does it for itself too, or later, really help the accuracy.
- Having AI interacting with Jira and GitHub, so effectively, and being controlled according to the policies I had defined, made it easier to fit into a non-AI process flow. Bringing in Confluence, Figma, Slack or other tools would further enhance the capability, and does not seem difficult to include.
- While it was not a complex codebase, I was impressed with how the project integrated into the existing environment and did not seem to have any problems. I’m sure it is not that easy in general, with larger or older systems.
- This project showed me that it is possible, with a 75-minute effort, and tiny cost, to add a reasonably complex feature, in full, in a controlled manner, adopting best security practices, with extensive testing, successfully.
- There were aspects I did not include, or ignored, in my agent/skills files, and I am sure that these could have introduced risks. While this was a demonstration, it was accessing a code repository, Jira, and Confluence, so live systems. It could have done serious damage if it went awry. Understanding how it works, and testing various sub-flows, behaviours, etc. in isolation and adopting a zero-trust model with AI in this case is essential.

- Is this the future? A competent software engineer, who drives a team of AI developers like my little guy, steering them to the goal seems to seem like an obvious progression. While it requires checking, controls, reviews and a human driving them, it opens possibilities to scale development hugely. As with regular development, ensuring the team members work collaboratively, without impacting each other's work negatively, while jointly creating solutions is the goal. There are already approaches with collaborative agents emerging to address this, so ... all problems are being addressed.

What Next?

There are so many ways to improve this going forward:

- Refining the workflow, and what is done at each step would be good. The logic needs to be tightened up, and I can see various behaviours along the way which could be improved; however, this is just updates to skills or agents. I would like to gather metrics to be able to assess how good it is, quality, productivity, etc. It would be good to understand the context status for each of the agents, to see if there are optimisations possible there.
- While the flow uses the agents & skills to provide the expertise and controls, I would introduce more automated checks along the way. More agents could be employed and trigger on different events, so dealing with issues as soon as they arise – for example, scanning and analysing code automatically on local commit.
- Integration with more tools to provide a complete development would be useful, with notifications, CI/CD flow creation and update, introducing phased rollout and support facilities. Marketing and sales material could be automatically generated.
- A larger greenfield project would be good to attack, where the flow may be run in parallel for multiple AI developers. Here the architecture and design of the system and understanding dependencies would be key. I see that this is where the human is really needed, though I can imagine AI can provide real support here, doing the heavy lifting. The approach I used is effective, but understanding how it can scale is important.
- While this was driven from Claude Code, I would like to see how it would be deployed in some of the other LLM tools, such as Cursor or GitHub Copilot. More of the ideas are becoming standardised, like skills or MCP, however with yaml or markdown being the standard way of configuring AI behaviour in all tools, it should not be too difficult.

Realistically the list of potential directions to take is endless. The capabilities of LLMs, the frameworks being provided, the supporting tools, including many new AI services for design, graphics, testing, etc. all up the game massively.

I can't wait to see what another year of LLM and AI evolution will come with.