



Spec-Driven Development

What is Spec-Driven Development?

And how can it make code faster?



Spec-Driven Development



Spec-Driven Development (SDD)
is writing structured
specifications first so AI
can automatically build and
test the code.

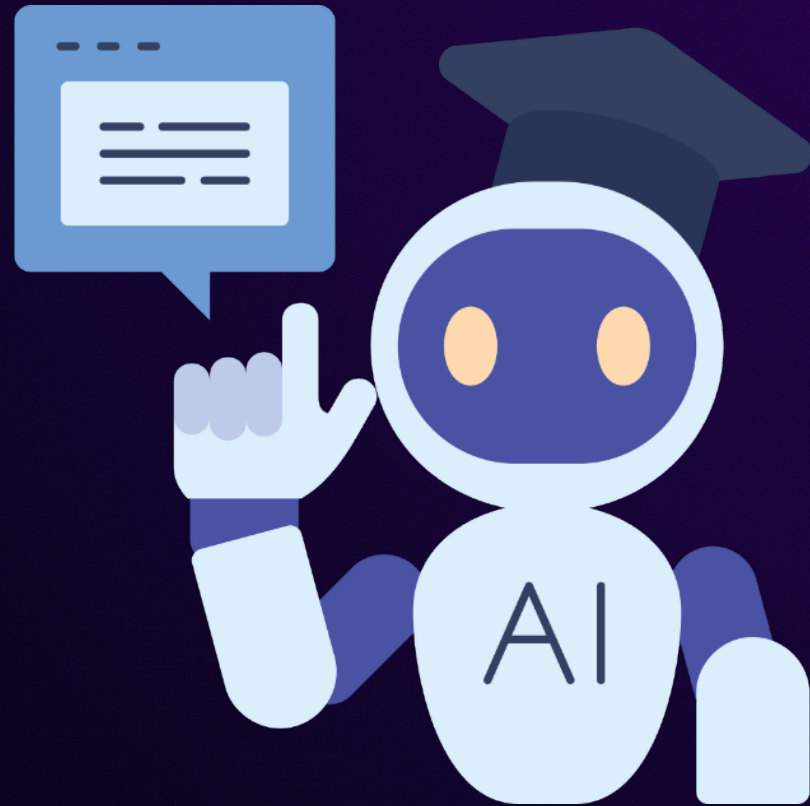
Spec-Driven Development



why?

Spec-Driven Development

AI Intern



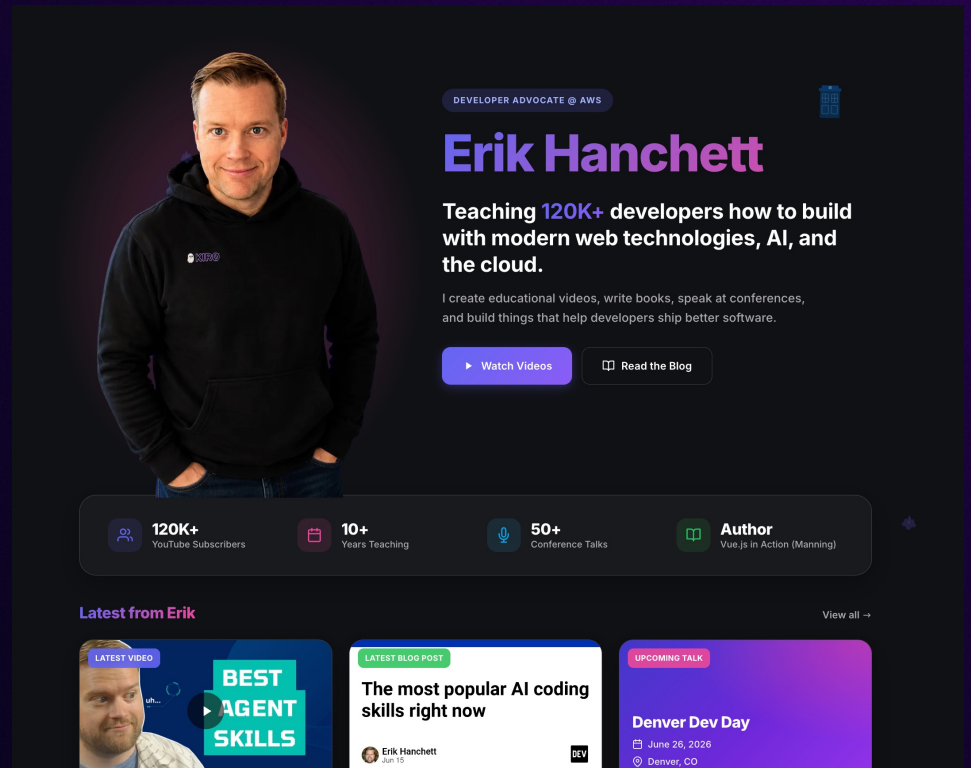


Erik Hanchett

- Senior Developer Advocate
- 15+ Software Development
- Connect with me on LinkedIn!

Where can I get these links?

<https://www.programwitherik.com/>

A screenshot of a developer profile page for Erik Hanchett. The page features a dark background with a portrait of Erik on the left. To the right of the portrait, it identifies him as a 'DEVELOPER ADVOCATE @ AWS' and lists his achievements: 'Teaching 120K+ developers how to build with modern web technologies, AI, and the cloud.' Below this, it states 'I create educational videos, write books, speak at conferences, and build things that help developers ship better software.' There are two buttons: 'Watch Videos' and 'Read the Blog'. A stats bar shows '120K+ YouTube Subscribers', '10+ Years Teaching', '50+ Conference Talks', and 'Author Vue.js in Action (Manning)'. At the bottom, there's a 'Latest from Erik' section with three items: a video titled 'BEST AGENT SKILLS', a blog post titled 'The most popular AI coding skills right now', and an upcoming talk at 'Denver Dev Day' on June 26, 2026.

DEVELOPER ADVOCATE @ AWS

Erik Hanchett

Teaching **120K+** developers how to build with modern web technologies, AI, and the cloud.

I create educational videos, write books, speak at conferences, and build things that help developers ship better software.

[▶ Watch Videos](#) [Read the Blog](#)

120K+ YouTube Subscribers **10+** Years Teaching **50+** Conference Talks **Author** Vue.js in Action (Manning)

Latest from Erik [View all →](#)

- LATEST VIDEO**
BEST AGENT SKILLS
- LATEST BLOG POST**
The most popular AI coding skills right now
Erik Hanchett
- UPCOMING TALK**
Denver Dev Day
June 26, 2026
Denver, CO

Spec-Driven Development



Can't the latest
frontier models
do everything?

Spec-Driven Development



6 Tips Before We Get Started!



© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Spec-Driven Development



1. Too much context!

Models get confused!

Spec-Driven Development



2. AGENTS.md / Steering



Spec-Driven Development



3. Use Skills

Spec-Driven Development



4. Too much Trust!

Are we code
reviewing?



5. Set up AI Code Reviews

Spec-Driven Development



6. Don't sacrifice
Speed over
maintainability!

What patterns are
being created?



History lesson...



© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Alexa+ / AI agents at scale

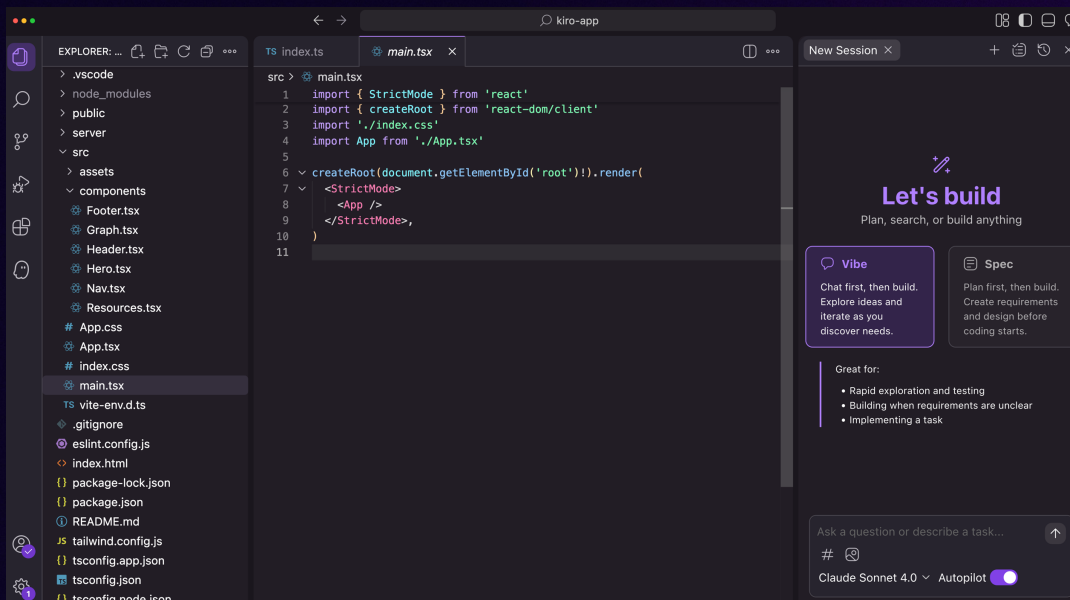
600 million devices

Hundreds of specialized expert systems

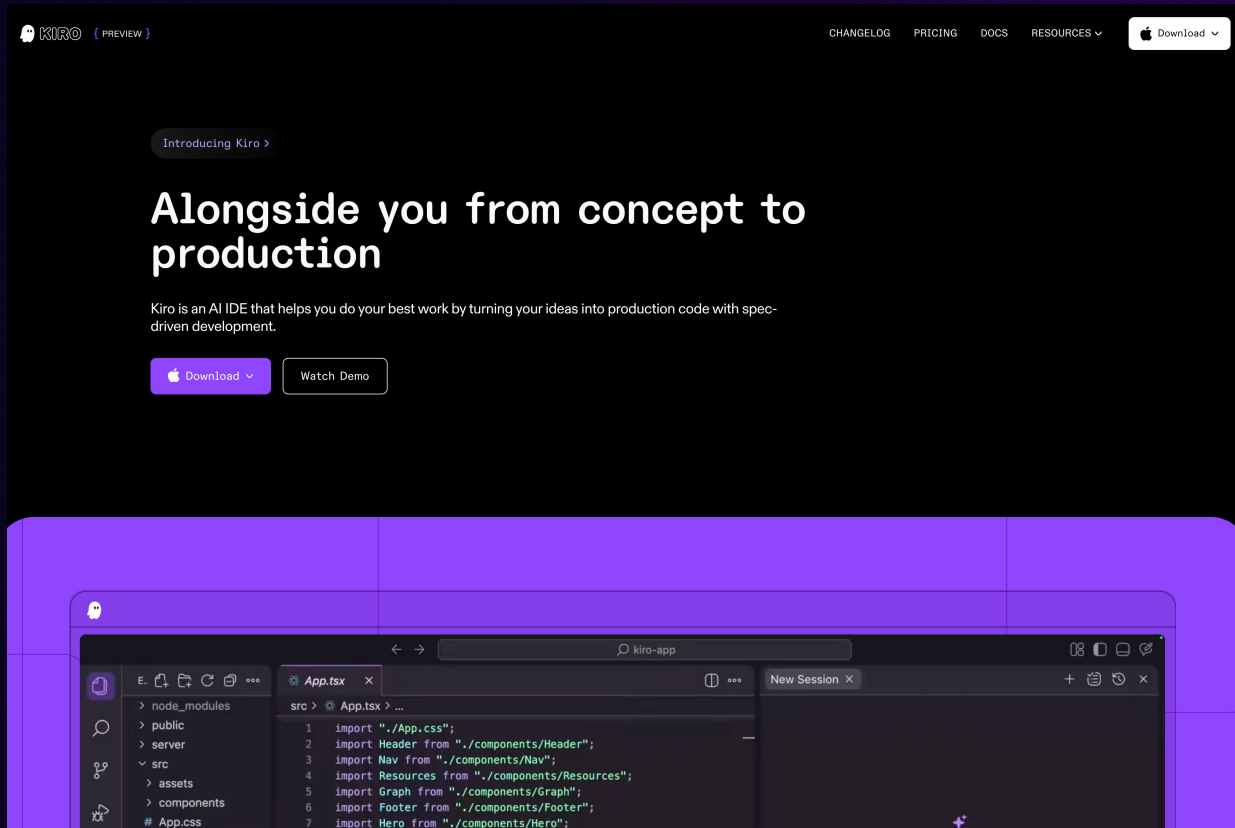
Tens of thousands of partner services and devices

Introducing Kiro

Kiro is a new AI IDE coding assistant,



Website! Kiro.dev



© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

So is the only reason to use Kiro for SDD?



No!



© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.



* Web

* iOS

* CLI

How do you do this without Kiro?



What do you need to tell your IDE?



- Tell the AI IDE what it should do
- Include the following:

What do you need to tell your IDE?



- Tell the AI IDE what it should do
- Include the following:
 - User Requirements

What do you need to tell your IDE?



- Tell the AI IDE what it should do
- Include the following:
 - User Requirements
 - Design Document

What do you need to tell your IDE?



- Tell the AI IDE what it should do
- Include the following:
 - User Requirements
 - Design Document
 - Implementation details

What do you need to tell your IDE?



- Tell the AI IDE what it should do
- Include the following:
 - User Requirements
 - Design Document
 - Implementation details
 - <https://github.com/github/spec-kit>
 - <https://github.com/Fission-AI/openspec>

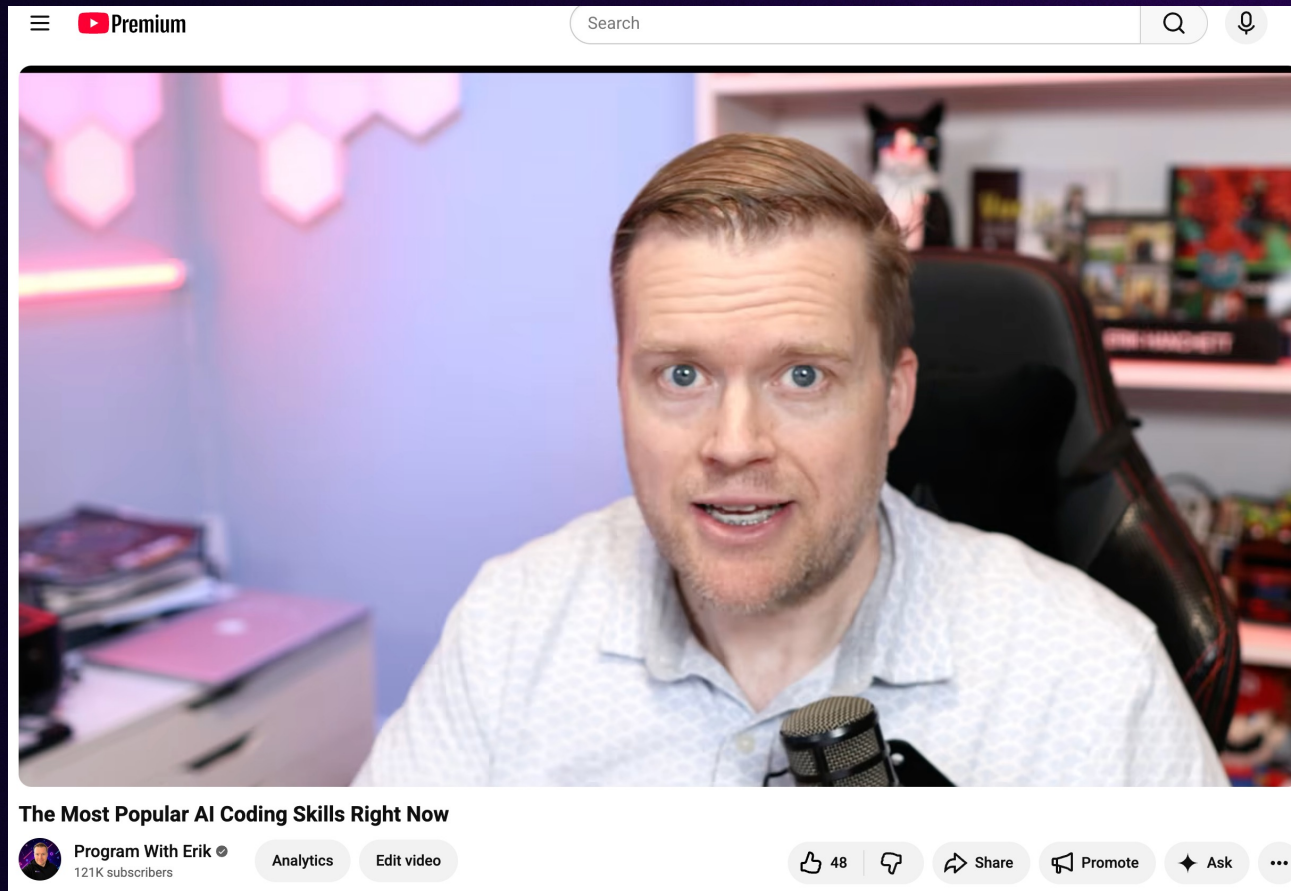
SDD Skill Wrappers

<https://github.com/open-gsd/gsd-core>

<https://github.com/obra/superpowers>



SDD Skill Wrappers



(AWS) AI-DLC

1. Inception (What & Why)
2. Construction (How)



<https://github.com/awslabs/aidlc-workflows>



© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Build More Architect Dreams

<https://docs.bmad-method.org/>



© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Bmad Method

1. Analysis (Optional)



© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Bmad Method

1. Analysis (Optional)
2. Planning (Required)



Bmad Method

1. Analysis (Optional)
2. Planning (Required)
3. Solutioning (Required for BMad Method/Enterprise tracks)

Bmad Method

1. Analysis (Optional)
2. Planning (Required)
3. Solutioning (Required for BMad Method/Enterprise tracks)
4. Implementation



Claude Code

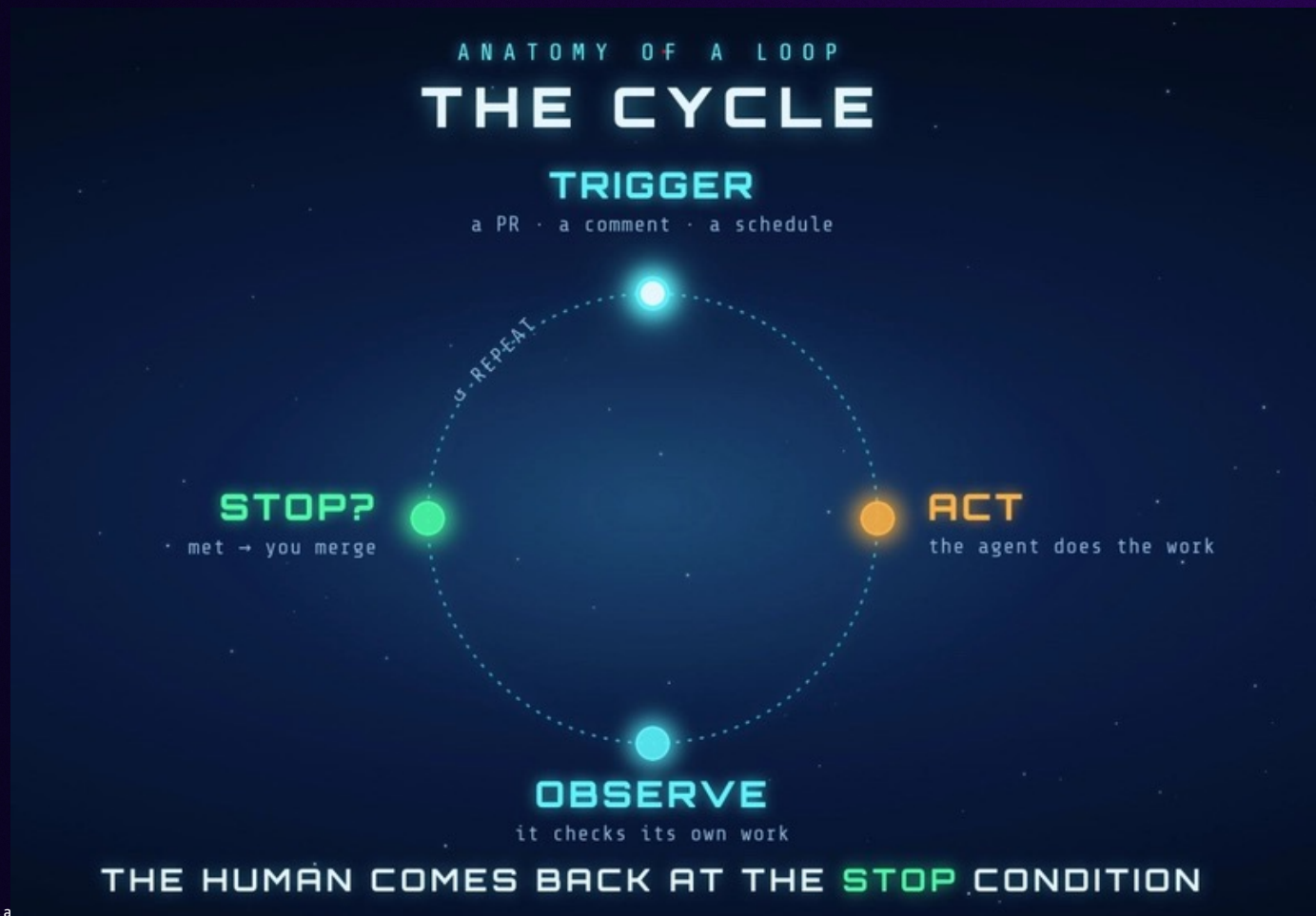
/plan

/goal



© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Loop engineering?



Key Findings

- All emphasize “plan before you code”

Key Findings

- All emphasize “plan before you code”
- **Context Engineering**

Key Findings

- All emphasize “plan before you code”
- **Context Engineering**
- Verification is the bottleneck

Key Findings

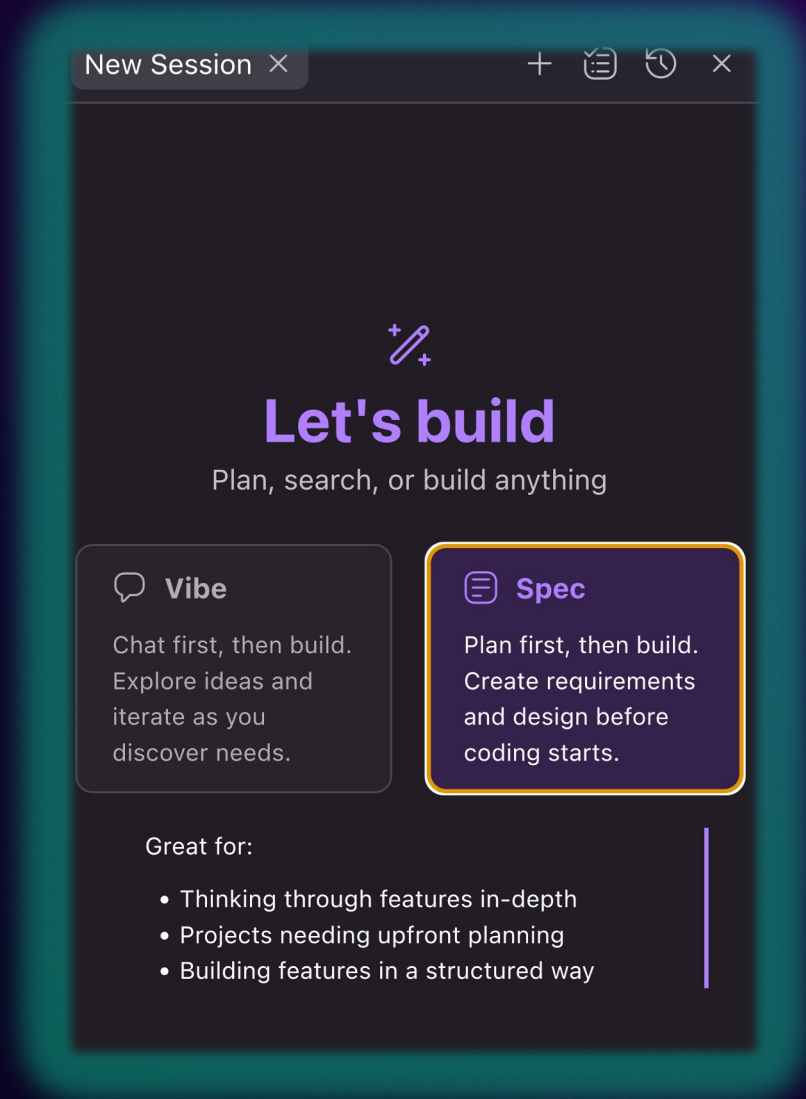
- All emphasize “plan before you code”
- **Context Engineering**
- Verification is the bottleneck
- All have the same weakness – not great for small bug fixes

Spec-kit



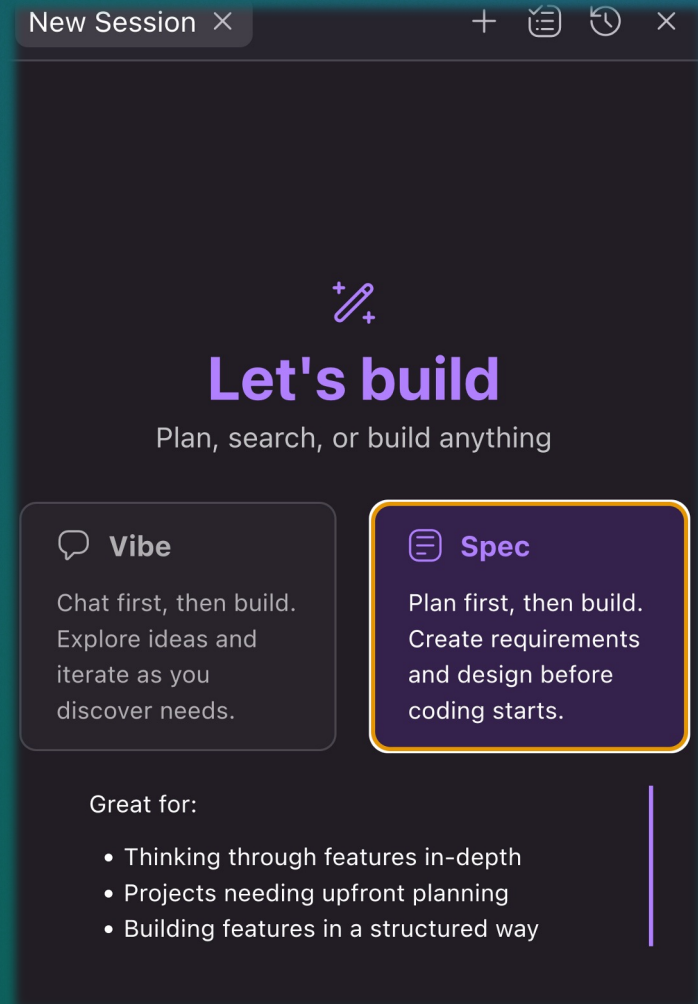
Features

- Spec-Driven Development



Ask it to begin!

“Build a Movie MCP server to keep track of movies I’ve watched”



Requirements Phase



```
requirements.md U X
.kiro > specs > movie-mcp-server > requirements.md > # Requirements Document

Spec: movie-mcp-server 1 Requirements 2 Design 3 Task list Refine

1 # Requirements Document
2
3 ## Introduction
4
5 This feature involves creating a comprehensive step-by-step guide for building a Movie MCP (Model Context
6 Protocol) server using AWS Bedrock AgentCore. The guide will enable developers to create their own movie
7 information service that can be integrated with AI assistants and other applications through the MCP protocol.
8 The server will provide movie data, showtimes, and related information through a standardized interface.
9
10 ## Requirements
11
12 ### Requirement 1
13
14 **User Story:** As a developer, I want a complete step-by-step guide for creating a movie MCP server, so that I
15 can build and deploy my own movie information service using AgentCore.
16
17 #### Acceptance Criteria
18
19 1. WHEN a developer follows the guide THEN they SHALL be able to create a functional MCP server from scratch
20 2. WHEN the guide is completed THEN the server SHALL support standard MCP protocol operations
21 3. WHEN following the setup steps THEN the developer SHALL have all necessary dependencies and configurations in
22 place
23 4. IF the developer has basic Python knowledge THEN they SHALL be able to complete the guide without external
24 assistance
25
26 ### Requirement 2
27
28 **User Story:** As a developer, I want the MCP server to provide movie information and showtimes, so that AI
29 assistants can access current movie data through the MCP protocol.
```

Design Phase



```
design.md U x
.kiro > specs > movie-mcp-server > design.md > # Design Document

Spec: movie-mcp-server 1 Requirements 2 Design 3 Task list Refine

1 # Design Document
2
3 ## Overview
4
5 This design outlines the creation of a comprehensive step-by-step guide for building a Movie MCP (Model Context
6 Protocol) server using AWS Bedrock AgentCore. The guide will transform the existing movie assistant agent into a
7 standardized MCP server that can be consumed by any MCP-compatible client application.
8
9 The design leverages the existing movie scraping functionality while restructuring it to conform to MCP protocol
10 standards, enabling broader integration possibilities with AI assistants, development tools, and other
11 applications that support MCP.
12
13 ## Architecture
14
15 ### High-Level Architecture
16
17 ...
18
19 MCP Client (AI Assistant) ↔ Movie MCP Server (AgentCore) ↔ Galaxy Theatres Website
20
21 |
22 |
23 |
24 |
25 |
26
27 AWS Services
28 - ECR
29 - CodeBuild
30 - IAM
```



Review

- Review the markdown files for inconsistencies, hallucinations, errors

Implementation Phase



```
tasks.md U x
.kiro > specs > movie-mcp-server > tasks.md > # Implementation Plan

Spec: movie-mcp-server  1 Requirements  2 Design  3 Task list  Update tasks

1  # Implementation Plan
2  ↪ Start task
3  - [ ] 1. Set up MCP server project structure and dependencies
4
5    - Create new directory structure for the MCP server implementation
6    - Set up Python virtual environment and install MCP Python SDK
7    - Create requirements.txt with all necessary dependencies (mcp, aiohttp, BeautifulSoup4, pydantic)
8    - Initialize basic project files (README.md, .gitignore, etc.)
9    - Requirements: 1.1, 5.1
10
11 ↪ Start task
12 - [ ] 2. Implement core movie data models and validation
13
14   - Create Pydantic models for MovieInfo with validation (id, title, genre, rating, duration, description,
15   poster_url, trailer_url)
16   - Create ShowtimeInfo model with validation (movie_id, theatre_name, date, times, ticket_url, available_seats)
17   - Create MovieToolResponse model for standardized MCP responses
18   - Write unit tests for all data models and validation logic
19   - Requirements: 2.1, 5.2, 6.1
20
21 ↪ Start task
22 - [ ] 3. Create movie data scraping service
23
24   - Implement MovieDataService class with async methods for web scraping
25   - Create fetch_current_movies() method using aiohttp and BeautifulSoup4
26   - Implement HTML parsing logic to extract movie information from Galaxy Theatres website
27   - Add error handling for network failures and parsing errors
28   - Write unit tests with mock HTTP responses
```

How Does MCP Help?



© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The Integrations Problem

Companies are spending more of their AI development time on plumbing instead of intelligence.

Model providers



Gemini

User applications



Data sources



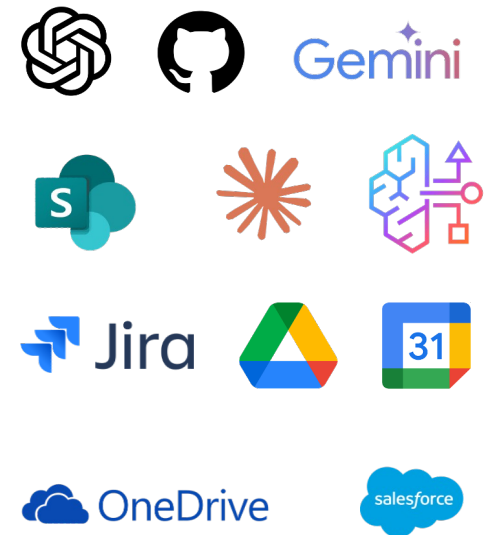
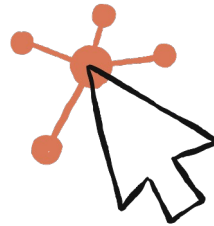
A Solution to the Integrations Problem

MCP is the USB-C port for AI - one standard interface that connects any AI to any tool.

User applications



MCP



Isn't MCP Dead?



© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

MCP <-> Spec Driven Development



© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

MCP <-> Spec Driven Development

- Specs can be pulled from your favorite project management service!

MCP <=> Spec Driven Development

- Specs can be pulled from your favorite project management service
- **Grab technical details for your project**

MCP <=> Spec Driven Development

- Specs can be pulled from your favorite project management service
- Grab technical details for your project
- will follow rules set forth in your steering files

Demo



© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

STIRTREK

> .nuxt

> app

> node_modules

> public

◆ .gitignore

TS nuxt.config.ts

{ } package-lock.json

{ } package.json

① README.md

tsconfig.json

stirtrek

Open cnat

Inline chat

Show All Commands

Go to File

Find in Files

Start Debugging

Toggle Terminal

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

○ → stirtrek git:(main)

New Session

Let's build

Plan, search, or build anything

Vibe

Chat first, then build.
Explore ideas and
iterate as you
discover needs.

Spec

Plan first, then build.
Create requirements
and design before
coding starts.

Great for:

- Rapid exploration and testing
- Building when requirements are unclear
- Implementing a task

Ask a question or describe a task...

Claude Opus 4.6 Autopilot

main

Autocomplete

Report issue

Kiro Power 2157.22 / 10000 updated just now

design.md — stirtrek

app.vue

design.md

.kiro > specs > movie-database > design.md > # Design Document: Movie Database (80s Theme)

movie-database

1 Design

2 Requirements

3 Task list

Sync Files

Continue

1

Design Document: Movie Database (80s Theme)

2

3 ## Overview

4

5 A retro-styled movie database application built with Nuxt 4 and Tailwind CSS v4, featuring a synthwave/80s aesthetic. Users can browse, search, and view details about movies with an immersive neon-lit, retro-futuristic visual experience reminiscent of 1980s pop culture.

6

7 The application uses a client-side data layer with a local JSON dataset of movies, providing fast filtering, searching, and sorting without requiring an external API. The 80s theme is expressed through neon color palettes (hot pink, electric blue, purple), retro typography, grid-line backgrounds, and glowing UI elements inspired by synthwave art.

8

9 ## Architecture

10

11 ```mermaid

12 graph TD

13 subgraph "Nuxt 4 Application"

14 A[app.vue - Layout Shell] --> B[pages/index.vue - Movie Grid]

15 A --> C[pages/movies/_id.vue - Movie Detail]

16 B --> D[components/MovieCard.vue]

17 B --> E[components/SearchBar.vue]

18 B --> F[components/GenreFilter.vue]

19 C --> G[components/MovieDetail.vue]

20 A --> H[components/AppHeader.vue]

21 A --> I[components/AppFooter.vue]

22 end

23

24 subgraph "Data Layer"

Open Preview

New Session

- Component hierarchy with pages for browsing and movie details
- Data models and filtering/sorting algorithms with formal correctness properties
- A synthwave design system with neon pinks, electric blues, and glow effects
- Composables for movies, search, and filtering
- Error handling, testing strategy, and performance considerations

Take a look at the design and let me know if you'd like any changes — different components, adjusted color palette, additional features, or anything else before we move on.

Est. Credits Used: 1.16 Elapsed time: 2m 29s

Est. Credits Used: 0 Elapsed time: 10s

> 2 changes accepted (View all)

Revert

Ask a question or describe a task...

#

Claude Opus 4.6

Autopilot

main*

0

-- NORMAL --

Ln 1, Col 1

Spaces: 4

UTF-8

LF

{}

Markdown

Autocomplete

Report issue

Kiro Power 2158.83 / 10000 updated 1m ago

FileEditSelectionViewGoRunTerminalWindowHelp

Preview requirements.md — stirtrek

design.md Urequirements.md UPreview requirements.md X

Acceptance Criteria

1. THE Application SHALL use lazy loading (loading="lazy") on movie poster images to defer off-screen image loading

2. IF a poster image fails to load, THEN THE Application SHALL swap to a placeholder image using an error handler on the image element

New Session X

I'll remove Requirement 12 and its related correctness property, then move on to creating the implementation tasks.

Read file(s)

requirements.md

design.md 440 — EOF

Now I'll remove Requirement 12 from requirements.md and Property 13 from design.md, then proceed to task creation.

Accepted edits to

requirements.md

Accepted edits to

design.md

Requirement 12 and its corresponding correctness property are removed. Now let me create the implementation tasks.

Creating

tasks.md

> Working

Cancel

Follow

Ask a question or describe a task...

#📎🕒Claude Opus 4.6Autopilot

main*

0 0

-- NORMAL --

Autocomplete

Report issue

Kiro Power 2162.83 / 10000 updated 2m ago

tasks.md — stirtrek

design.md U requirements.md U Preview tasks.md tasks.md U X

.kiri > specs > movie-database > tasks.md > # Implementation Plan: Movie Database (80s Theme) > ## Tasks

movie-database 1 Design > 2 Requirements > 3 Task list Sync Files Run all tasks

1 # Implementation Plan: Movie Database (80s Theme)

7 ## Tasks

45 - Add base styles: dark background, neon text-shadow glow utility classes, bold uppercase heading styles

46 - Add grid-line background pattern using CSS linear-gradient

47 - Requirements: 10.1, 10.2, 10.3, 10.4, 10.6, 10.7_

48

49 ✓ Task completed | View changes | View execution

50 - [x] 2.2 Create glow animation styles in `app/assets/css/animations.css`

51 - Define neon glow box-shadow effects for card hover states

52 - Define hover scale and glow intensity transitions

53 - Use transform and opacity for GPU-accelerated animations

54 - Import this file in `nuxt.config.ts` CSS array

55 - Requirements: 10.5_

56

57 ✓ Task completed | View changes | View execution

58 - [x] 3. Layout and browsing UI components

59

60 ✓ Task completed | View changes | View execution

61 - [x] 3.1 Create `app/components/AppHeader.vue`

62 - Display application title with neon glow text effect

63 - Provide navigation link to home page

64 - Style with gradient border bottom and Neon_Theme colors

65 - Requirements: 9.1, 9.2_

66

67 ✓ Task completed | View changes | View execution

68 - [x] 3.2 Create `app/components/AppFooter.vue`

69 - Display footer with retro-styled content

70 - Consistent Neon_Theme styling

71

New Session X

15 classic 80s movies in the dataset

- **Composables:** useSearch (debounced), useMovies (filtering, sorting, genre extraction)
- **80s theme:** Neon color tokens, glow text utilities, card hover animations, grid-line background
- **Components:** AppHeader ("RETRO FLIX" with neon glow), AppFooter, SearchBar, GenreFilter, SortControls, MovieCard
- **Index page:** Responsive movie grid with search, genre filtering, and sort controls all wired up

Run `npm run dev` to fire it up and take a look. The movie detail page links won't work yet (that's Task 6), but the grid, search, filtering, and sorting should all be functional. Let me know what you think!

Est. Credits Used: 8.58 Elapsed time: 11m 37s

> 17 changes accepted (View all)

Revert X

Ask a question or describe a task...

Claude Opus 4.6 Autopilot

main* 0 -- NORMAL -- Ln 101, Col 1 Spaces: 4 UTF-8 LF Markdown Autocomplete Report issue Kiro Power 2172.56 / 10000 updated 12m ago

tasks.md — stirtrek

design.md U

requirements.md U

Preview tasks.md

tasks.md U X

.kiri > specs > movie-database > tasks.md > # Implementation Plan: Movie Database (80s Theme) > ## Tasks

movie-database 1 Design > 2 Requirements > 3 Task list Sync Files Run all tasks

1 # Implementation Plan: Movie Database (80s Theme)

7 ## Tasks

105 - Render MovieCard components in a responsive grid (1 col → 2 col → 3 col → 4 col)

106 - Show loading indicator while data loads

107 - Show empty state message when no movies match filters

108 - Requirements: 1.1, 1.2, 1.3, 3.1, 4.2, 5.3, 6.1, 9.4

109

Start task

110 [] 5. MVP Checkpoint

111 - Verify the app runs, movie grid displays, search/filter/sort work. Ask the user if questions arise.

112

Task completed View changes View execution

113 [x] 6. Integrate TMDB API for real movie data and posters

Task completed

114 - [x] 6.1 Create Nuxt server API route `server/api/movies.get.ts`

115 - Fetch popular/top-rated 80s movies from TMDB API (discover endpoint filtered to 1980-1989)

116 - Map TMDB response to the existing Movie interface (id, title, year, genre, rating, director, cast, plot, poster, runtime)

117 - Use TMDB image base URL for poster paths (<https://image.tmdb.org/t/p/w500/>)

118 - Read TMDB API key from runtime config (NUXT_TMDB_API_KEY env var)

119 - Return Movie[] array

120 - Cache results server-side to avoid excessive API calls

121

Task completed

122 - [x] 6.2 Create Nuxt server API route `server/api/movies/[id].get.ts`

123 - Fetch a single movie's full details from TMDB by ID

124 - Include credits endpoint for director and cast

125 - Map to Movie interface

New Session X

All filtering/sorting logic unchanged

Config:

nuxt.config.ts has runtimeConfig.tmdbApiKey wired to NUXT_TMDB_API_KEY env var

.env.example documents the required variable

To use real TMDB data, grab a free API key from themoviedb.org/settings/api and add it to a .env file:

NUXT_TMDB_API_KEY=your_key_here

Without the key, the app gracefully falls back to the local 15-movie dataset. Let me know when you're ready to continue with the remaining tasks.

Est. Credits Used: 8.23 Elapsed time: 4m 37s

> 19 changes accepted (View all)

Revert X

Ask a question or describe a task...

Claude Opus 4.6 Autopilot

main* 0 -- NORMAL -- Ln 109, Col 1 Spaces: 4 UTF-8 LF {} Markdown Autocomplete Report issue Kiro Power 2212.78 / 10000 updated 1m ago

movies.get.ts — stirtrek

in.md U

requirements.md U

Preview tasks.md

tasks.md U

TS movies.get.ts 9+, U

server > api > TS movies.get.ts > ...

```
72  * Retrieves multiple pages and enriches each movie with credits.
73  */
74  async function fetchMoviesFromTmdb(apiKey: string): Promise<Movie[]> {
75    const pagesToFetch = 3 // ~60 results (20 per page)
76    const allTmdbMovies: TmdbMovie[] = []
77
78    for (let page = 1; page <= pagesToFetch; page++) {
79      const data = await $fetch<TmdbDiscoverResponse>(`
80        ${TMDB_API_BASE}/discover/movie`,
81        {
82          query: {
83            api_key: apiKey,
84            'primary_release_date.gte': '1980-01-01',
85            'primary_release_date.lte': '1989-12-31',
86            sort_by: 'vote_count.desc',
87            page,
88          },
89        },
90      )
91      allTmdbMovies.push(...data.results)
92    }
93
94    // Fetch credits in parallel (batched to avoid rate-limiting)
95    const movies: Movie[] = []
96    const batchSize = 5
97
98    for (let i = 0; i < allTmdbMovies.length; i += batchSize) {
99      const batch = allTmdbMovies.slice(i, i + batchSize)
100      const creditResults = await Promise.all(
101        batch.map((m) => fetchCredits(m.id, apiKey)),
102      )
103    }
```

New Session

router, search bar, genre filter, sort controls, movie cards)

- ✓ Task 4 — Movie grid page with everything wired together
- ✓ Task 5 — MVP checkpoint
- ✓ Task 6 — TMDB API integration (server routes, caching, fallback to local data)
- ✓ Task 7 — Movie detail page with cinematic 80s layout
- ✓ Task 9 — Final checkpoint (build verified)

Remaining (optional):

- Task 8 — Property-based tests (all sub-tasks marked optional with *)

Run `npm run dev` to see the full app. Add a TMDB API key in `.env` for real movie data and posters, or leave it out to use the built-in 15-movie dataset. Let me know if you'd like any tweaks or want to tackle the property tests.

Est. Credits Used: 5.47 Elapsed time: 2m 38s

> 6 changes accepted (View all)

Revert

Ask a question or describe a task...

0 Claude Opus 4.6 Autopilot

main*

10 0

-- NORMAL --

Ln 1, Col 1

Spaces: 2

UTF-8

LF

{ } TypeScript

✓ Autocomplete

Report issue

Kiro Power 2218.46 / 10000 updated 1m ago

tasks.md — stirtrek

lesign.md U

requirements.md U

Preview tasks.md

TS movies.get.ts 9+, U

tasks.md U

.kiri > specs > movie-database > tasks.md > # Implementation Plan: Movie Database (80s Theme) > ## Tasks

movie-database

1 Design

2 Requirements

3 Task list

Sync Files

Run all tasks

1

Implementation Plan: Movie Database (80s Theme)

7

Tasks

151

- Use useMovies composable to look up movie by route param id

152

- Render MovieDetail component with full movie information

153

- Display "Movie not found" message with link home if id is invalid

154

- Requirements: 7.1, 7.2, 7.3, 7.4

155

156

- [x] 8. Property-based tests

157

- [x] 8.1 Set up Vitest + fast-check testing infrastructure

158

- Install vitest, fast-check, @vue/test-utils as dev dependencies

159

- Create vitest config if needed

160

161

- [x] 8.2 Write property tests for movie data validation

162

- **Property 1: Movie Schema Validation** - For any Movie record, validate id is non-empty unique string, year is 1900-current year, rating is 0-10, genre has at least one element, title is non-empty, runtime is a positive integer

163

- **Validates: Requirements 2.1, 2.2, 2.3, 2.4, 2.5, 2.6**

164

- Test file: `tests/properties/movie-schema.test.ts`

165

166

- [x] 8.3 Write property tests for search precision and completeness

167

- **Property 2: Search Precision** - For any search query and movie list, every movie in filtered results has the query as a case-insensitive substring of title, plot, or director

168

- **Property 3: Search Completeness** - For any search query and movie list, every movie matching the query appears in filtered results

169

- **Validates: Requirements 3.1, 3.5**

170

- Test file: `tests/properties/search.test.ts`

171

172

- [x] 8.4 Write property tests for genre filtering

173

- **Property 4: Genre Extraction Exhaustiveness** - Extracted genres equal the sorted unique set of

New Session x

+ x

detail renders all fields

movie-card Property MovieCard renders required fields

12

The test infrastructure includes:

- Vitest + happy-dom for fast headless testing
- fast-check for property-based test generation
- @vue/test-utils for component mounting
- A shared movieArbitrary helper that generates valid Movie objects
- A setup file that registers Vue auto-imports (ref, computed, etc.)

Run npm test anytime to verify all 12 correctness properties hold. The entire spec is now complete.

Est. Credits Used: 5.2 Elapsed time: 5m 6s

> 14 changes accepted (View all)

Revert x

Ask a question or describe a task...

Claude Opus 4.6 Autopilot

main*

10 0

-- NORMAL --

Ln 197, Col 35

Spaces: 2

UTF-8

LF

{}

Markdown

Autocomplete

Report issue

Kiri Power 2222.17 / 10000 updated 5m ago

How to learn more!



- <https://kiro.dev>

<https://discord.gg/kirodotdev>

Thanks!

- LinkedIn: Erik Hanchett
- X/Bluesky: Erikch