ryanjoachim / **mcp-rtfm** Public

Code | Issues 1 | Pull requests 1 | Actions | Projects | Security and quality | Insights

main | 2 Branches | 0 Tags | Go to file | Go to file | Code

**ryanjoachim** Merge pull request #3 from RobertCoop/patch-1 — 054fe51 · last year

| | | |
|---|---|---|
| src | bug: MCP crashes on init | last year |
| .gitignore | Initial commit: MCP server for intelligent do… | 2 years ago |
| README.md | Update README.md | 2 years ago |
| package-lock.json | Initial commit: MCP server for intelligent do… | 2 years ago |
| package.json | Cheeky description+introduction | 2 years ago |
| tsconfig.json | Initial commit: MCP server for intelligent do… | 2 years ago |

README

# MCP-RTFM

TypeScript 5.0 | MCP 0.1.0 | License MIT

> "RTFM!" they say, but what if there's no FM to R? 🤔 Enter MCP-RTFM: an MCP server that helps you *create* the F*ing Manual everyone keeps telling people to read! Using advanced content analysis, metadata generation, and intelligent search capabilities, it transforms your non-existent or unreadable docs into an interconnected knowledge base that actually answers those "basic questions" before they're asked.

> **Plot twist**: Instead of just telling people to RTFM, now you can actually give them an FM worth R-ing! Because the best response to "read the f*ing manual" is having a manual that's actually worth reading. 📚✨

## 📚 Table of Contents



## 🚀 Quick Start

```
# Install dependencies
npm install

# Build the server
npm run build

# Add to your MCP settings and start using
await use_mcp_tool({
  server: "mcp-rtfm",
  tool: "analyze_project_with_metadata", // Enhanced initialization
  args: { projectPath: "/path/to/project" }
});
```

```
// This will:
// 1. Create documentation structure
// 2. Analyze content with unified/remark
// 3. Generate intelligent metadata
// 4. Build search index with minisearch
// 5. Add structured front matter
// 6. Make your docs actually readable!
```

## ✨ Features

### Documentation Management Tools

- `analyze_existing_docs` - Analyze and enhance existing documentation with content analysis and metadata
- `analyze_project_with_metadata` - Initialize documentation structure with enhanced content analysis and metadata generation
- `analyze_project` - Basic initialization of documentation structure
- `read_doc` - Read a documentation file (required before updating)
- `update_doc` - Update documentation using diff-based changes
- `get_doc_content` - Get current content of a documentation file
- `get_project_info` - Get project structure and documentation status
- `search_docs` - Search across documentation files with highlighted results
- `update_metadata` - Update documentation metadata
- `get_related_docs` - Find related documentation based on metadata and content links
- `customize_template` - Create or update documentation templates

### Default Documentation Files

The server automatically creates and manages these core documentation files:

- `techStack.md` - Detailed inventory of tools, libraries, and configurations
- `codebaseDetails.md` - Low-level explanations of code structure and logic
- `workflowDetails.md` - Step-by-step workflows for key processes
- `integrationGuides.md` - Instructions for external system connections
- `errorHandling.md` - Troubleshooting strategies and practices
- `handoff_notes.md` - Summary of key themes and next steps

### Documentation Templates

Built-in templates for different documentation types:

- Standard Documentation Template
- API Documentation Template
- Workflow Documentation Template

Custom templates can be created using the `customize_template` tool.

## 📝 Example Workflows

### 1. Analyzing Existing Documentation

```
// Enhance existing documentation with advanced analysis
await use_mcp_tool({
  server: "mcp-rtfm",
  tool: "analyze_existing_docs",
  args: { projectPath: "/path/to/project" }
});

// This will:
// - Find all markdown files in .handoff_docs
// - Analyze content structure with unified/remark
```

```
// - Generate intelligent metadata
// - Build search index
// - Add front matter if not present
// - Establish document relationships
// - Preserve existing content

// The results include:
// - Enhanced metadata for all docs
// - Search index population
// - Content relationship mapping
// - Git context if available
```

### 2. Enhanced Project Documentation Setup

```
// Initialize documentation with advanced content analysis
await use_mcp_tool({
  server: "mcp-rtfm",
  tool: "analyze_project_with_metadata",
  args: { projectPath: "/path/to/project" }
});

// Results include:
// - Initialized documentation files
// - Generated metadata from content analysis
// - Established document relationships
// - Populated search index
// - Added structured front matter
// - Git repository context

// Get enhanced project information
const projectInfo = await use_mcp_tool({
  server: "mcp-rtfm",
  tool: "get_project_info",
  args: { projectPath: "/path/to/project" }
});

// Search across documentation with intelligent results
const searchResults = await use_mcp_tool({
  server: "mcp-rtfm",
  tool: "search_docs",
  args: {
    projectPath: "/path/to/project",
    query: "authentication"
  }
});

// Results include:
// - Weighted matches (title matches prioritized)
// - Fuzzy search results
// - Full content context
// - Related document suggestions
```

### 3. Updating Documentation with Content Links

```
// First read the document
await use_mcp_tool({
  server: "mcp-rtfm",
  tool: "read_doc",
  args: {
    projectPath: "/path/to/project",
    docFile: "techStack.md"
  }
});

// Update with content that links to other docs
await use_mcp_tool({
  server: "mcp-rtfm",
  tool: "update_doc",
  args: {
```

```
    projectPath: "/path/to/project",
    docFile: "techStack.md",
    searchContent: "[Why this domain is critical to the project]",
    replaceContent: "The tech stack documentation provides essential context for development. See [[workflowDetails]]
    continueToNext: true // Automatically move to next document
  }
});
```

### 4. Managing Documentation Metadata

```
// Update metadata for better organization
await use_mcp_tool({
  server: "mcp-rtfm",
  tool: "update_metadata",
  args: {
    projectPath: "/path/to/project",
    docFile: "techStack.md",
    metadata: {
      title: "Technology Stack Overview",
      category: "architecture",
      tags: ["infrastructure", "dependencies", "configuration"]
    }
  }
});

// Find related documentation
const related = await use_mcp_tool({
  server: "mcp-rtfm",
  tool: "get_related_docs",
  args: {
    projectPath: "/path/to/project",
    docFile: "techStack.md"
  }
});
```

### 5. Searching Documentation with Context

```
// Search with highlighted results
const results = await use_mcp_tool({
  server: "mcp-rtfm",
  tool: "search_docs",
  args: {
    projectPath: "/path/to/project",
    query: "authentication"
  }
});

// Results include:
// - File name
// - Line numbers
// - Highlighted matches
// - Context around matches
```

### 6. Creating Custom Templates

```
// Create a custom template for architecture decisions
await use_mcp_tool({
  server: "mcp-rtfm",
  tool: "customize_template",
  args: {
    templateName: "architecture-decision",
    content: `# {title}

## Context
[Background and context for the decision]

## Decision
```

```
[The architecture decision made]

## Consequences
[Impact and trade-offs of the decision]

## Related Decisions
[Links to related architecture decisions]`,
    metadata: {
      category: "architecture",
      tags: ["decision-record", "design"]
    }
  }
});
```

## 🔧 Installation

### VSCode (Roo Cline)

Add to settings file at: Add to settings file at:

- Windows: `%APPDATA%\Code\User\globalStorage\rooveterinaryinc.roo-cline\settings\cline_mcp_settings.json`
- MacOS: `~/Library/Application Support/Code/User/globalStorage/rooveterinaryinc.roo-cline/settings/cline_mcp_settings.json`
- Linux: `~/.config/Code/User/globalStorage/rooveterinaryinc.roo-cline/settings/cline_mcp_settings.json`

```
{
  "mcpServers": {
    "mcp-rtfm": {
      "command": "node",
      "args": ["<path-to-mcp-rtfm>/build/index.js"],
      "disabled": false,
      "alwaysAllow": []
    }
  }
}
```

### Claude Desktop

Add to config file at:

- Windows: `%APPDATA%\Claude\claude_desktop_config.json`
- MacOS: `~/Library/Application Support/Claude/claude_desktop_config.json`
- Linux: `~/.config/Claude/claude_desktop_config.json`

```
{
  "mcpServers": {
    "mcp-rtfm": {
      "command": "node",
      "args": ["<path-to-mcp-rtfm>/build/index.js"],
      "disabled": false,
      "alwaysAllow": []
    }
  }
}
```

## 🎯 Advanced Features

### Content Linking

Use `[[document-name]]` syntax to create links between documents. The server automatically tracks these relationships and includes them when finding related documentation.

### Metadata-Driven Organization

Documents are organized using:

- Categories (e.g., "architecture", "api", "workflow")
- Tags for flexible grouping
- Automatic relationship discovery based on shared metadata
- Content link analysis

### Enhanced Content Analysis

The server uses advanced libraries for better documentation management:

- **unified/remark** for Markdown processing:
  - AST-based content analysis
  - Accurate heading structure detection
  - Code block and link extraction
  - Proper Markdown parsing and manipulation
- **minisearch** for powerful search capabilities:
  - Fast fuzzy searching across all documentation
  - Field-weighted search (titles given higher priority)
  - Full content and metadata indexing
  - Efficient caching with TTL management
  - Real-time search index updates

### Intelligent Metadata Generation

- Automatic content analysis for categorization
- Smart tag generation based on content patterns
- Structured front matter in documents
- AST-based title and section detection
- Code snippet identification and tagging
- Context-aware result presentation

### Template System

- Built-in templates for common documentation types
- Custom template support with metadata defaults
- Template inheritance and override capabilities
- Placeholder system for consistent formatting

## 🛠️ Development

```
# Install dependencies
npm install

# Build the server
npm run build

# Development with auto-rebuild
npm run watch
```

## 🐛 Debugging

Since MCP servers communicate over stdio, debugging can be challenging. Use the MCP Inspector:

```
npm run inspector
```

The Inspector will provide a URL to access debugging tools in your browser.

## License

MIT © Model Context Protocol

### Releases

No releases published

### Packages

No packages published

### Contributors 2


**ryanjoachim** Ryan Joachim


**RobertCoop** Robert Coop

### Languages

**JavaScript** 100.0%