crazyrabbitLTC / **mcp-code-review-server** Public

Code | Issues 2 | Pull requests 1 | Actions | Projects | Security and quality | Insights

main | 1 Branch | 0 Tags | Go to file | Go to file | Code

crazyrabbitLTC rename tools — ac4f281 · last year

build	Fix: Updated tool registration to follo…	last year
code-review	Update code-review submodule wit…	last year
code-review-server	Rename server to CodeQualityAdvi…	last year
docs	Update implementation checklist wit…	last year
src	Fix: Updated tool registration to follo…	last year
test/__tests__	Clean up test files and remove dupli…	last year
.env.example	Fix LLM response parsing and add …	last year
.gitignore	merging	last year
MCP-Typescript-readme.txt	Fix: Updated tool registration to follo…	last year
MCP-docs.txt	Implement LLM integration for code …	last year
README.md	Fix: Updated tool registration to follo…	last year
bun.lockb	Implement direct LLM API integratio…	last year
package-lock.json	Fix: Updated tool registration to follo…	last year
package.json	Fix server connection issues and im…	last year
repomix-output.txt	rename tools	last year
tsconfig.json	Fix Repomix integration and update …	last year

README

Code Review Server

A custom MCP server that performs code reviews using Repomix and LLMs.

Features

- Flatten codebases using Repomix
- Analyze code with Large Language Models
- Get structured code reviews with specific issues and recommendations
- Support for multiple LLM providers (OpenAI, Anthropic, Gemini)
- Handles chunking for large codebases

Installation

```
# Clone the repository
git clone https://github.com/yourusername/code-review-server.git
cd code-review-server

# Install dependencies
npm install

# Build the server
npm run build
```

Configuration

Create a `.env` file in the root directory based on the `.env.example` template:

```
cp .env.example .env
```

Edit the `.env` file to set up your preferred LLM provider and API key:

```
# LLM Provider Configuration
LLM_PROVIDER=OPEN_AI
OPENAI_API_KEY=your_openai_api_key_here
```

Usage

As an MCP Server

The code review server implements the Model Context Protocol (MCP) and can be used with any MCP client:

```
# Start the server
node build/index.js
```

The server exposes two main tools:

1. `analyze_repo`: Flattens a codebase using Repomix
2. `code_review`: Performs a code review using an LLM

When to Use MCP Tools

This server provides two distinct tools for different code analysis needs:

analyze_repo

Use this tool when you need to:

- Get a high-level overview of a codebase's structure and organization
- Flatten a repository into a textual representation for initial analysis
- Understand the directory structure and file contents without detailed review
- Prepare for a more in-depth code review
- Quickly scan a codebase to identify relevant files for further analysis

Example situations:

- "I want to understand the structure of this repository before reviewing it"
- "Show me what files and directories are in this codebase"
- "Give me a flattened view of the code to understand its organization"

code_review

Use this tool when you need to:

- Perform a comprehensive code quality assessment
- Identify specific security vulnerabilities, performance bottlenecks, or code quality issues
- Get actionable recommendations for improving code
- Conduct a detailed review with severity ratings for issues
- Evaluate a codebase against best practices

Example situations:

- "Review this codebase for security vulnerabilities"
- "Analyze the performance of these specific JavaScript files"
- "Give me a detailed code quality assessment of this repository"
- "Review my code and tell me how to improve its maintainability"

When to use parameters:

- `specificFiles`: When you only want to review certain files, not the entire repository
- `fileTypes`: When you want to focus on specific file extensions (e.g., .js, .ts)
- `detailLevel`: Use 'basic' for a quick overview or 'detailed' for in-depth analysis
- `focusAreas`: When you want to prioritize certain aspects (security, performance, etc.)

Using the CLI Tool

For testing purposes, you can use the included CLI tool:

```
node build/cli.js <repo_path> [options]
```

Options:

- `--files <file1,file2>`: Specific files to review
- `--types <.js,.ts>`: File types to include in the review
- `--detail <basic|detailed>`: Level of detail (default: detailed)
- `--focus <areas>`: Areas to focus on (security,performance,quality,maintainability)

Example:

```
node build/cli.js ./my-project --types .js,.ts --detail detailed --focus security,quality
```

Development

```
# Run tests
npm test

# Watch mode for development
npm run watch

# Run the MCP inspector tool
npm run inspector
```

LLM Integration

The code review server integrates directly with multiple LLM provider APIs:

- **OpenAI** (default: gpt-4o)
- **Anthropic** (default: claude-3-opus-20240307)
- **Gemini** (default: gemini-1.5-pro)

Provider Configuration

Configure your preferred LLM provider in the `.env` file:

```
# Set which provider to use
LLM_PROVIDER=OPEN_AI  # Options: OPEN_AI, ANTHROPIC, or GEMINI

# Provider API Keys (add your key for the chosen provider)
OPENAI_API_KEY=your-openai-api-key
ANTHROPIC_API_KEY=your-anthropic-api-key
GEMINI_API_KEY=your-gemini-api-key
```

Model Configuration

You can optionally specify which model to use for each provider:

```
# Optional: Override the default models
OPENAI_MODEL=gpt-4-turbo
ANTHROPIC_MODEL=claude-3-sonnet-20240229
GEMINI_MODEL=gemini-1.5-flash-preview
```

How the LLM Integration Works

1. The `code_review` tool processes code using Repomix to flatten the repository structure
2. The code is formatted and chunked if necessary to fit within LLM context limits
3. A detailed prompt is generated based on the focus areas and detail level
4. The prompt and code are sent directly to the LLM API of your chosen provider
5. The LLM response is parsed into a structured format
6. The review is returned as a JSON object with issues, strengths, and recommendations

The implementation includes retry logic for resilience against API errors and proper formatting to ensure the most relevant code is included in the review.

Code Review Output Format

The code review is returned in a structured JSON format:

```
{
  "summary": "Brief summary of the code and its purpose",
  "issues": [
    {
      "type": "SECURITY|PERFORMANCE|QUALITY|MAINTAINABILITY",
      "severity": "HIGH|MEDIUM|LOW",
      "description": "Description of the issue",
      "line_numbers": [12, 15],
      "recommendation": "Recommended fix"
    }
  ],
  "strengths": ["List of code strengths"],
  "recommendations": ["List of overall recommendations"]
}
```

Releases

No releases published

Packages

No packages published

Contributors 1

crazyrabbitLTC Dennison Bertram

Languages

JavaScript 61.9% **TypeScript** 38.1%