Raise Your Hack Hackathon

MULTI-AGENT DEVELOPER TOOLS

Transforming your coding workflow with intelligent, specialized AI assistance.

TRACK - Blackbox Ai

Agenda

- Introduction to the Multi-Agent AI Extension
- Tech Stack
- Introduction
- Core Agents: Coding, Security, and Documentation
- High Level Architecture
- Demo Images
- Key Features & Technical Integration
- Setup & Requirements

TECH STACK

CORE TECHNOLOGIES

- **TypeScript** – Primary programming language
- **Node.js** – Runtime environment
- **VS Code Extension API** – Extension development framework

AI SERVICES

- **Groq API** – Primary AI provider (supports LLaMA, Mixtral models)
- **DeepSeek R1** – Secondary AI provider (via BlackBox AI integration)

VS CODE INTEGRATION APIS

- **Completion Provider API** – Inline code suggestions
- **Diagnostic API** – Static analysis and security issue highlighting
- **Webview API** – Custom UI panels inside VS Code
- **Command API** – Handles custom commands and interactions
- **Status Bar API** – Adds quick-access controls in the status bar

FRONTEND (WEBVIEW PANEL)

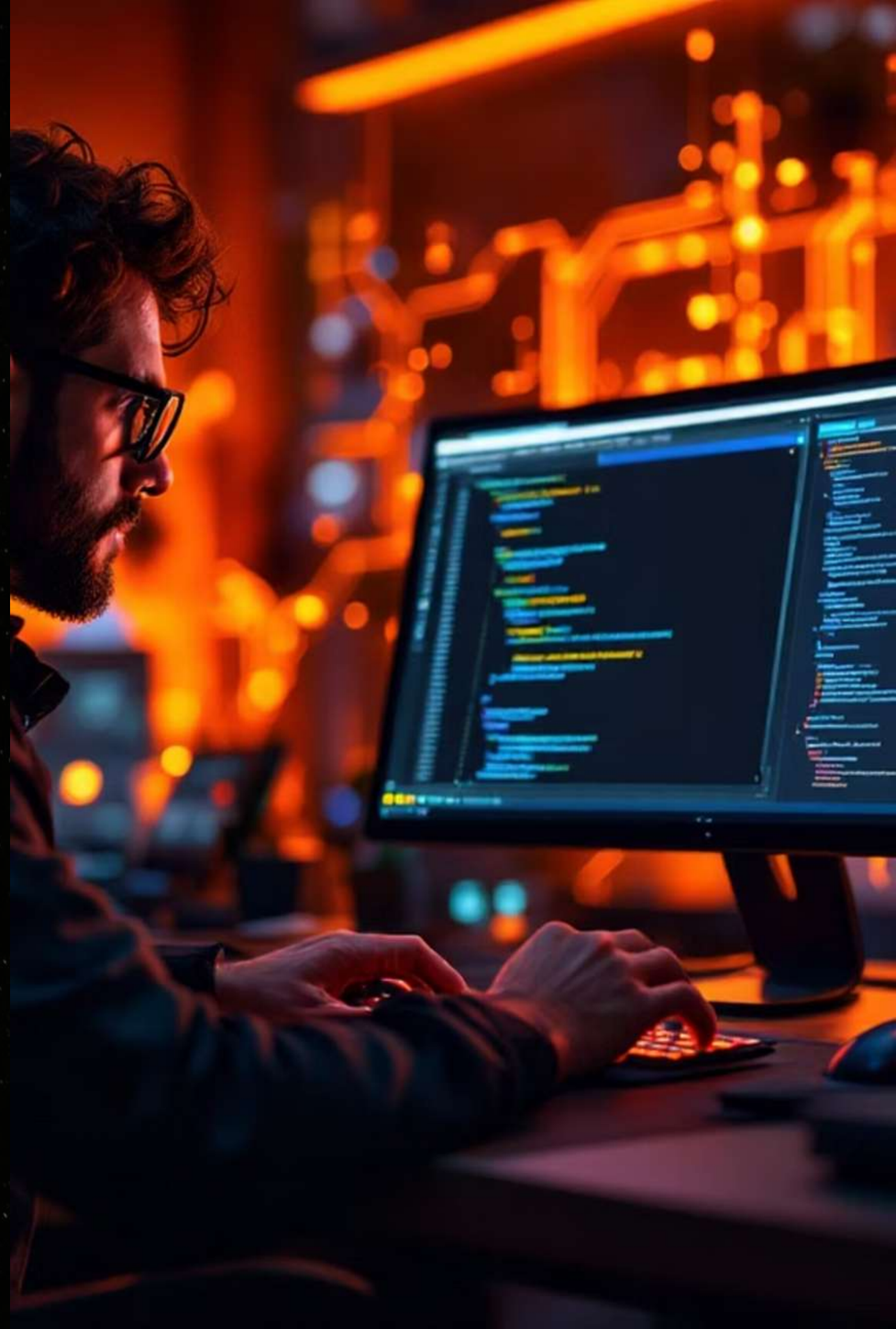
- **HTML/CSS/JavaScript** – Custom UI component development
- **VS Code Theme Integration** – Ensures consistent look with user’s current theme

KEY LIBRARIES

- **HTTP Client** – Handles API communication
- **Streaming Support** – Enables real-time AI responses
- **Caching System** – Optimizes performance and reduces latency
- **Retry Logic** – Manages request failures with intelligent retries
- **Regular Expressions** – Used for pattern matching in security scanning

A Comprehensive AI Development Assistant

The Multi-Agent AI VS Code Extension is an all-in-one tool designed to streamline your development process directly within VS Code. It leverages advanced AI to provide real-time assistance, ensuring code quality, security, and comprehensive documentation.



Core Agents

Three specialized AI agents work in synergy to cover every aspect of your coding journey:

Coding Agent

Intelligent code completions, quality analysis, optimization suggestions, and automated unit test generation.

Security Agent

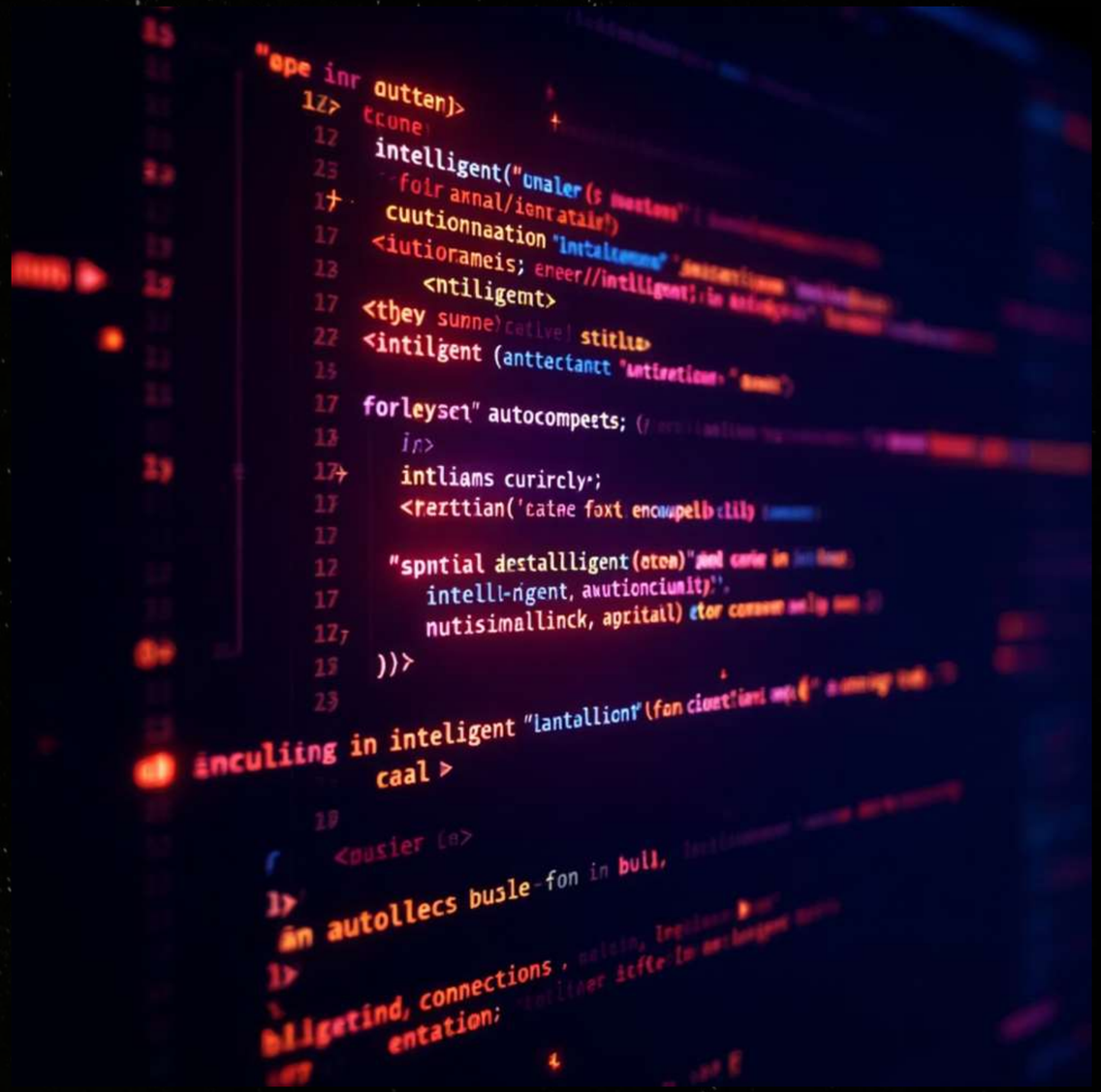
Comprehensive security scans, static code analysis, dependency scanning, secrets detection, and compliance checks.

Documentation Agent

Generates API docs, architectural documentation, troubleshooting guides, and code explanations.

Coding Agent: Your Intelligent Co-pilot

- **Intelligent Code Completions:** Context-aware suggestions as you type, significantly speeding up development.
- **Quality & Optimization:** Identifies code smells, suggests refactorings, and optimizes performance.
- **Automated Unit Tests:** Generates boiler-plate unit tests for functions and modules across various languages.
- **Language Agnostic:** Supports TypeScript, JavaScript, Python, Java, C++, C#, Go, and Rust.



Security Agent: Fortifying Your Codebase

- **Static Code Analysis:** Proactive identification of potential vulnerabilities without code execution.
- **Dependency Scanning:** Flags known vulnerabilities in third-party libraries and packages.
- **Secrets Detection:** Prevents accidental exposure of API keys, passwords, and sensitive information.
- **Compliance Checks:** Ensures adherence to industry standards like OWASP Top 10 and PCI DSS.

Documentation Agent: Clarity at Your Fingertips

Effortlessly generate comprehensive documentation, tailored to your needs.

API Documentation

Generates clear and concise API specifications, complete with parameters, return types, and examples.

Architectural Overviews

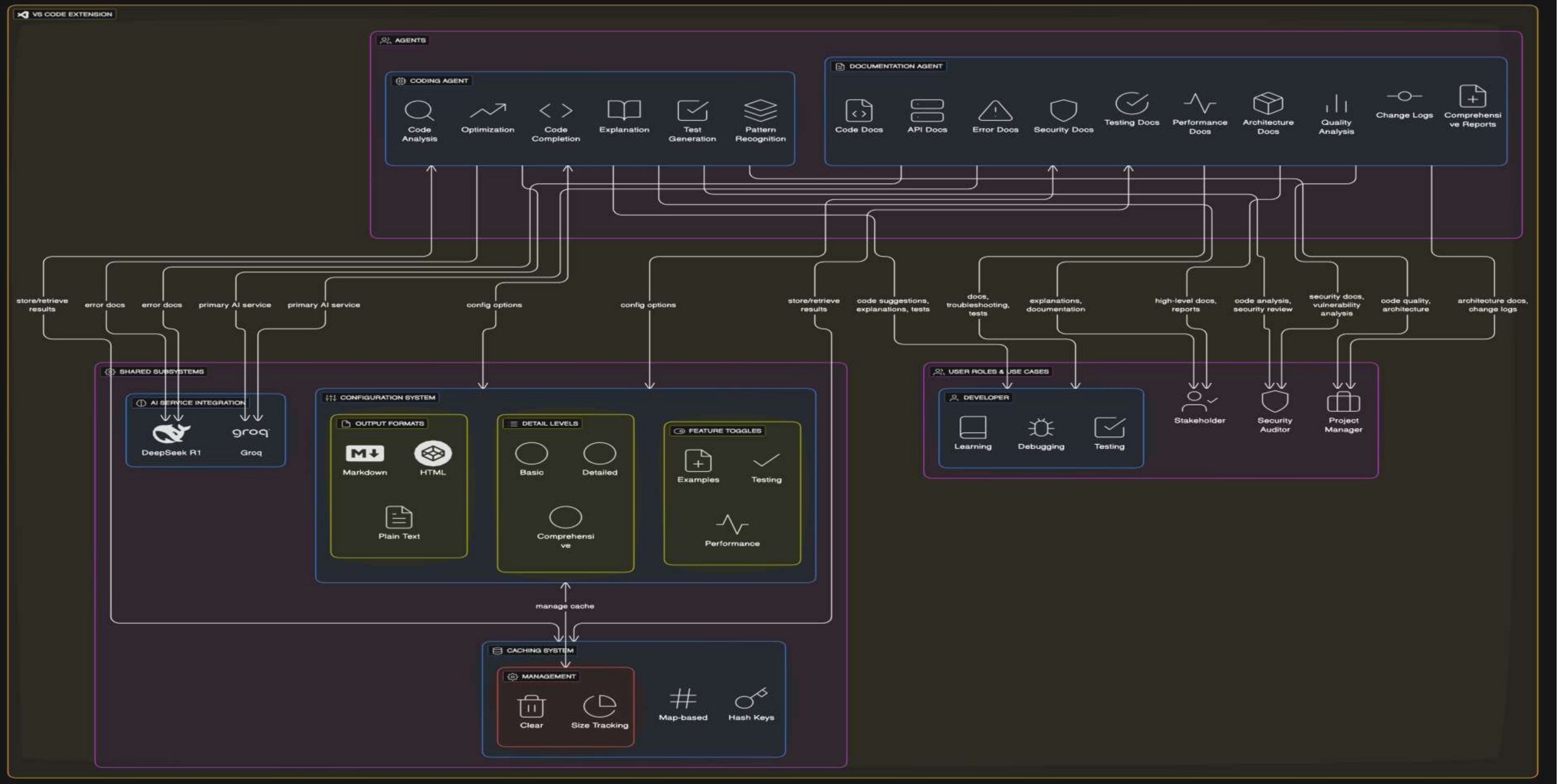
Maps out system components and their interactions, aiding in onboarding and system understanding.

Troubleshooting Guides

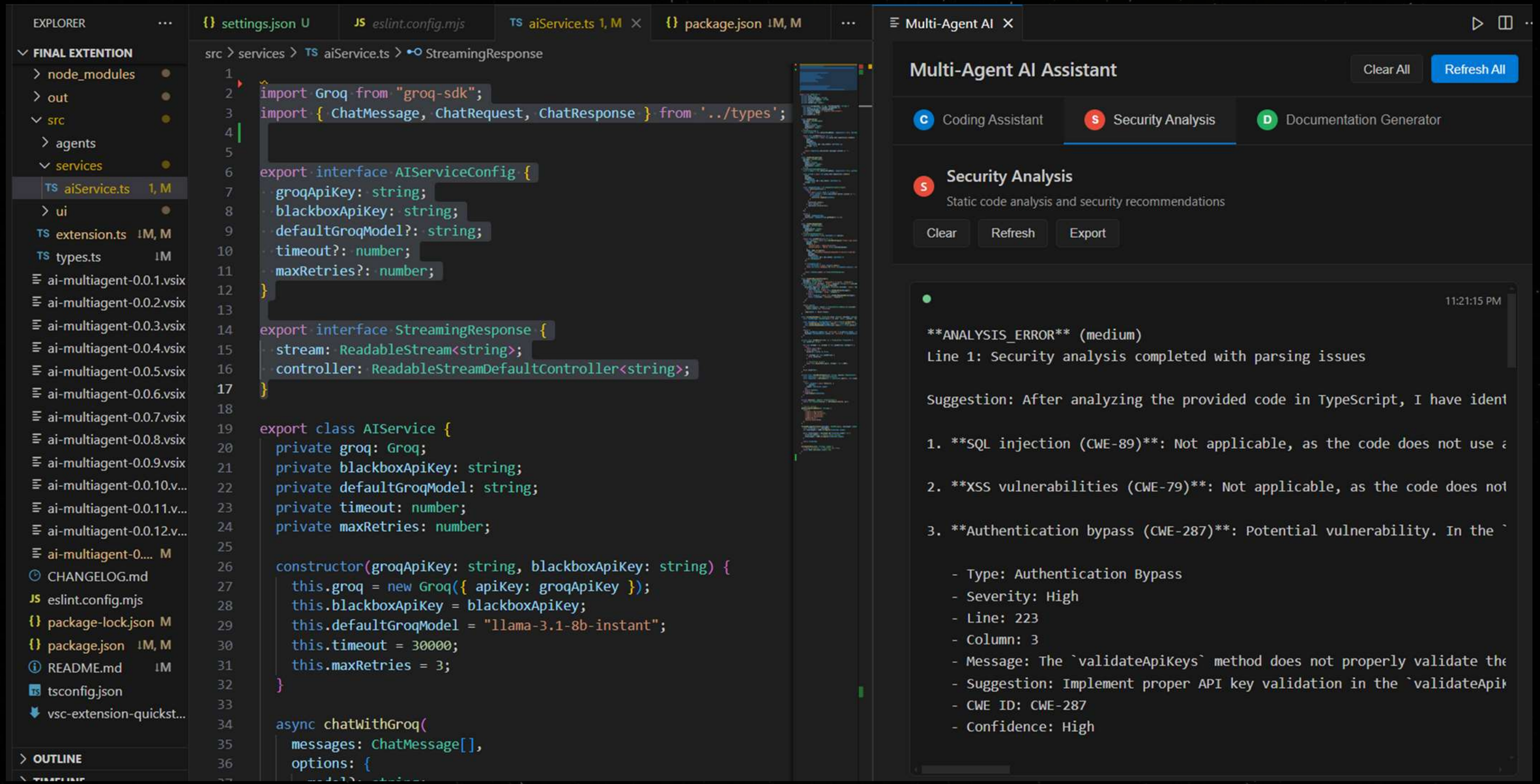
Creates step-by-step solutions for common issues, reducing debugging time.

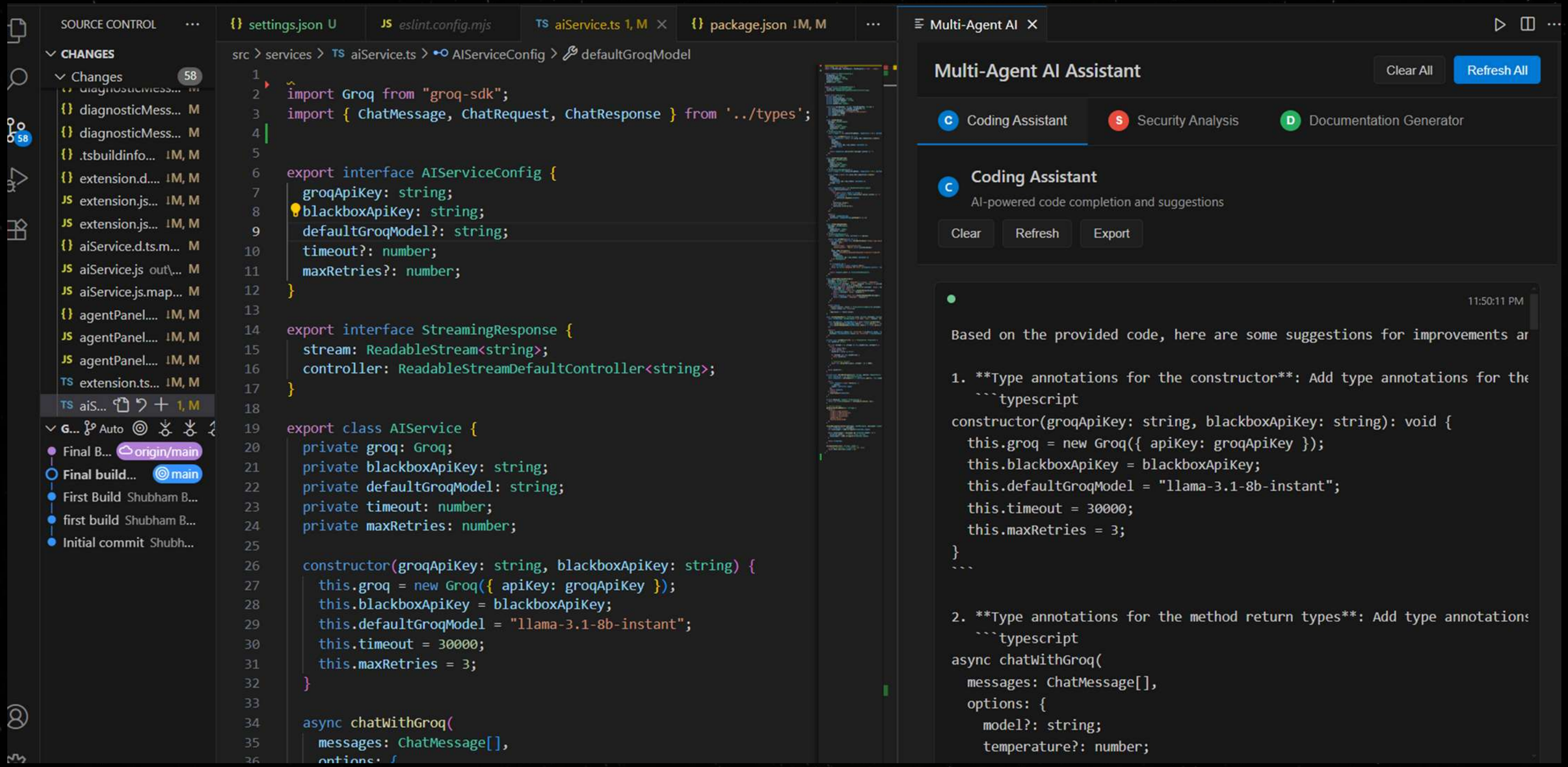
Code Explanations

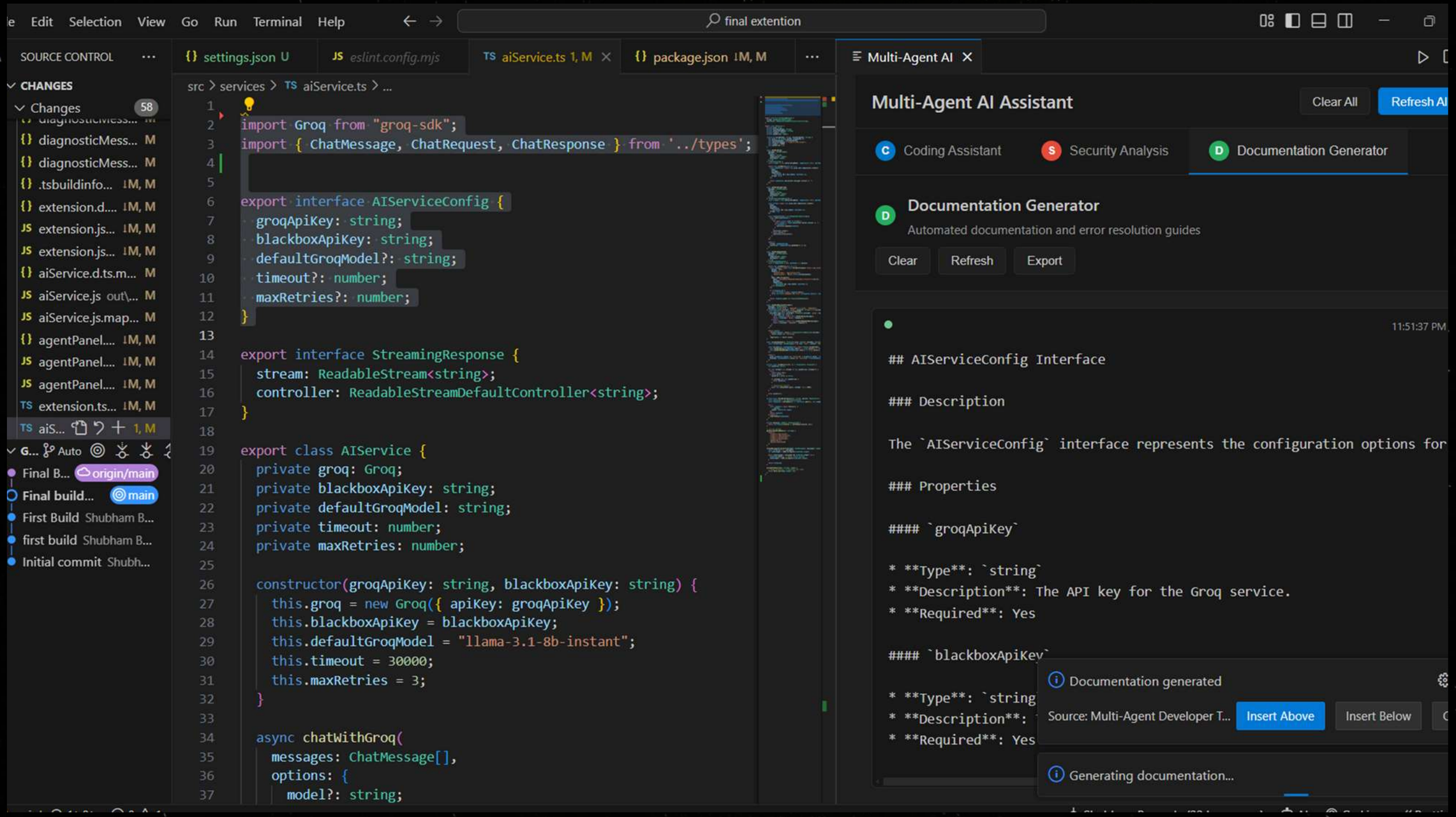
Provides detailed explanations for complex code blocks, improving readability and maintainability.


VS CODE EXTENSION
AGENTS
CODING AGENT
Code Analysis
Optimization
Code Completion
Explanation
Test Generation
Pattern Recognition
DOCUMENTATION AGENT
Code Docs
API Docs
Error Docs
Security Docs
Testing Docs
Performance Docs
Architecture Docs
Quality Analysis
Change Logs
Comprehensive Reports
store/retrieve results
error docs
error docs
primary AI service
primary AI service
config options
config options
store/retrieve results
code suggestions, explanations, tests
docs, troubleshooting, tests
explanations, documentation
high-level docs, reports
code analysis, security review
security docs, vulnerability analysis
code quality, architecture
architecture docs, change logs
SHARED SUBSYSTEMS
AI SERVICE INTEGRATION
DeepSeek R1
Groq
CONFIGURATION SYSTEM
OUTPUT FORMATS
Markdown
HTML
Plain Text
DETAIL LEVELS
Basic
Detailed
Comprehensive
FEATURE TOGGLES
Examples
Testing
Performance
manage cache
CACHING SYSTEM
MANAGEMENT
Clear
Size Tracking
Map-based
Hash Keys
USER ROLES & USE CASES
DEVELOPER
Learning
Debugging
Testing
Stakeholder
Security Auditor
Project Manager

Demo Images


EXPLORER
FINAL EXTENTION
node_modules
out
src
agents
services
aiService.ts 1, M
ui
extension.ts M, M
types.ts M
ai-multiagent-0.0.1.vsix
ai-multiagent-0.0.2.vsix
ai-multiagent-0.0.3.vsix
ai-multiagent-0.0.4.vsix
ai-multiagent-0.0.5.vsix
ai-multiagent-0.0.6.vsix
ai-multiagent-0.0.7.vsix
ai-multiagent-0.0.8.vsix
ai-multiagent-0.0.9.vsix
ai-multiagent-0.0.10.v...
ai-multiagent-0.0.11.v...
ai-multiagent-0.0.12.v...
ai-multiagent-0.... M
CHANGELOG.md
eslint.config.mjs
package-lock.json M
package.json M, M
README.md M
tsconfig.json
vsc-extension-quickst...
OUTLINE
settings.json U
eslint.config.mjs
aiService.ts 1, M
package.json M, M
src > services > aiService.ts > StreamingResponse
import Groq from "groq-sdk";
import { ChatMessage, ChatRequest, ChatResponse } from '../types';
export interface AIServiceConfig {
groqApiKey: string;
blackboxApiKey: string;
defaultGroqModel?: string;
timeout?: number;
maxRetries?: number;
}
export interface StreamingResponse {
stream: ReadableStream<string>;
controller: ReadableStreamDefaultController<string>;
}
export class AIService {
private groq: Groq;
private blackboxApiKey: string;
private defaultGroqModel: string;
private timeout: number;
private maxRetries: number;
constructor(groqApiKey: string, blackboxApiKey: string) {
this.groq = new Groq({ apiKey: groqApiKey });
this.blackboxApiKey = blackboxApiKey;
this.defaultGroqModel = "llama-3.1-8b-instant";
this.timeout = 30000;
this.maxRetries = 3;
}
async chatWithGroq(
messages: ChatMessage[],
options: {
Multi-Agent AI
Multi-Agent AI Assistant
Clear All
Refresh All
Coding Assistant
Security Analysis
Documentation Generator
Security Analysis
Static code analysis and security recommendations
Clear
Refresh
Export
11:21:15 PM
ANALYSIS_ERROR (medium)
Line 1: Security analysis completed with parsing issues
Suggestion: After analyzing the provided code in TypeScript, I have ident
1. **SQL injection (CWE-89)**: Not applicable, as the code does not use a
2. **XSS vulnerabilities (CWE-79)**: Not applicable, as the code does not
3. **Authentication bypass (CWE-287)**: Potential vulnerability. In the `
- Type: Authentication Bypass
- Severity: High
- Line: 223
- Column: 3
- Message: The `validateApiKeys` method does not properly validate the
- Suggestion: Implement proper API key validation in the `validateApik
- CWE ID: CWE-287
- Confidence: High


SOURCE CONTROL
CHANGES
Changes 58
diagnosticMess... M
diagnosticMess... M
.tsbuildinfo... M
extension.d... M
extension.js... M
extension.js... M
aiService.d.ts.m... M
aiService.js out\... M
aiService.js.map... M
agentPanel... M
agentPanel... M
agentPanel... M
extension.ts... M
aiS... 1, M
Auto
Final B... origin/main
Final build... main
First Build Shubham B...
first build Shubham B...
Initial commit Shubh...
settings.json U
eslint.config.mjs
aiService.ts 1, M
package.json M
src > services > aiService.ts > AIServiceConfig > defaultGroqModel
import Groq from "groq-sdk";
import { ChatMessage, ChatRequest, ChatResponse } from '../types';
export interface AIServiceConfig {
groqApiKey: string;
blackboxApiKey: string;
defaultGroqModel?: string;
timeout?: number;
maxRetries?: number;
}
export interface StreamingResponse {
stream: ReadableStream<string>;
controller: ReadableStreamDefaultController<string>;
}
export class AIService {
private groq: Groq;
private blackboxApiKey: string;
private defaultGroqModel: string;
private timeout: number;
private maxRetries: number;
constructor(groqApiKey: string, blackboxApiKey: string) {
this.groq = new Groq({ apiKey: groqApiKey });
this.blackboxApiKey = blackboxApiKey;
this.defaultGroqModel = "llama-3.1-8b-instant";
this.timeout = 30000;
this.maxRetries = 3;
}
async chatWithGroq(
messages: ChatMessage[],
Multi-Agent AI
Multi-Agent AI Assistant
Clear All
Refresh All
Coding Assistant
Security Analysis
Documentation Generator
Coding Assistant
AI-powered code completion and suggestions
Clear
Refresh
Export
11:50:11 PM
Based on the provided code, here are some suggestions for improvements ar
1. **Type annotations for the constructor**: Add type annotations for the
```typescript
constructor(groqApiKey: string, blackboxApiKey: string): void {
this.groq = new Groq({ apiKey: groqApiKey });
this.blackboxApiKey = blackboxApiKey;
this.defaultGroqModel = "llama-3.1-8b-instant";
this.timeout = 30000;
this.maxRetries = 3;
}
```
2. **Type annotations for the method return types**: Add type annotations
```typescript
async chatWithGroq(
messages: ChatMessage[],
options: {
model?: string;
temperature?: number;
```


Edit Selection View Go Run Terminal Help
final extention
SOURCE CONTROL
CHANGES
Changes 58
settings.json U
eslint.config.mjs
aiService.ts 1, M
package.json !M, M
src > services > aiService.ts > ...
import Groq from "groq-sdk";
import { ChatMessage, ChatRequest, ChatResponse } from '../types';
export interface AIServiceConfig {
groqApiKey: string;
blackboxApiKey: string;
defaultGroqModel?: string;
timeout?: number;
maxRetries?: number;
}
export interface StreamingResponse {
stream: ReadableStream<string>;
controller: ReadableStreamDefaultController<string>;
}
export class AIService {
private groq: Groq;
private blackboxApiKey: string;
private defaultGroqModel: string;
private timeout: number;
private maxRetries: number;
constructor(groqApiKey: string, blackboxApiKey: string) {
this.groq = new Groq({ apiKey: groqApiKey });
this.blackboxApiKey = blackboxApiKey;
this.defaultGroqModel = "llama-3.1-8b-instant";
this.timeout = 30000;
this.maxRetries = 3;
}
async chatWithGroq(
messages: ChatMessage[],
options: {
model?: string;
Final B... origin/main
Final build... main
First Build Shubham B...
first build Shubham B...
Initial commit Shubh...
Multi-Agent AI
Multi-Agent AI Assistant
Clear All
Refresh AI
Coding Assistant
Security Analysis
Documentation Generator
Documentation Generator
Automated documentation and error resolution guides
Clear
Refresh
Export
11:51:37 PM
AIServiceConfig Interface
Description
The `AIServiceConfig` interface represents the configuration options for
Properties
`groqApiKey`
* **Type**: `string`
* **Description**: The API key for the Groq service.
* **Required**: Yes
`blackboxApiKey`
Documentation generated
Source: Multi-Agent Developer T...
Insert Above
Insert Below
Generating documentation...

Key Features & Technical Integration

Key Features

- **Real-time Assistance:** Inline suggestions with contextual awareness and caching.
- **Unified Interface:** Dedicated side panel with a tabbed interface for seamless agent management.
- **Export Capabilities:** Save outputs in various formats (TXT, MD, JSON).
- **Visual Feedback:** Color-coded status indicators and progress bars.

Technical Integration

- **AI Engines:** Powered by Groq Cloud API and BlackBox AI APIs for robust performance.
- **Streaming Responses:** Supports real-time interaction for dynamic feedback.
- **Configurable Standards:** Customizable security rules and compliance standards.
- **Multi-provider AI:** Enhanced reliability through support for multiple AI providers.

Setup & Next Steps

Getting started is simple:
Download Extention – **Multi-Agent Developer Tools**

- **API Keys:** Configure your Groq Cloud API and BlackBox API keys in VS Code settings.
- **Compatibility:** Fully compatible with major programming languages and VS Code themes.
- **Seamless Integration:** Transforms your VS Code into an AI-enhanced development hub.

Team Members

- Shubham Bansode – Team Leader
- Tejas Durge
- Ajay Surse
- Vaishnavi Dhanavade

DISCORD ID’s

- shubham_111916
- tejasdurge.
- ajaysurse_pvt
- vaishnavi2789

We appreciate your time and attention!