

## WFP Requirements Template

# FastDoc AI

## Software Requirements Specification

Module: WFP1 | Version 1.1 | 22.3.2026

### Version 1.1 — Changelog (22.3.2026)

- **FR-01:** Added specification of "near cursor" proximity rule (3 lines above/below).
- **FR-08 → Must Have:** LLM connection (local or remote) is now a Must Have requirement.
- **FR-08b (new):** Must Have — system detects missing/unreachable LLM and notifies user clearly.
- **FR-09:** Updated to reflect that both local (Ollama) and remote LLM providers can be configured.
- **QS-A:** Updated — local LLM is default and Must Have; remote LLM opt-in remains user-controlled.
- **Admin:** Project board link added; GitHub repo invitation for leonardo1710 requested.

## 1. Team Members

- Atay Muhammed Ihsan
- Sheikh Annas-Daud

**Project Board:** [Link to project board — to be added by team]

**GitHub Repository:** Invitation to **leonardo1710** is pending — please add this collaborator to the repository.

## 2. Product Scope

FastDoc AI is a Visual Studio Code extension that automatically generates inline documentation for functions and methods. When a developer positions their cursor on or near a function and triggers the extension, it analyzes the function signature and generates a complete documentation block — such as a JSDoc comment for JavaScript/TypeScript or a docstring for Python — and inserts it directly into the editor.

The primary goal of FastDoc AI is to reduce the time and effort developers spend on writing documentation manually. Documentation is a critical part of software quality, but it is often skipped under time pressure. By automating this step, FastDoc AI makes good documentation habits easier to follow.

The system is composed of two components: a VS Code extension that handles editor integration, user interaction, and language detection; and an LLM backend — defaulting to a locally running model via Ollama, with optional support for remote LLM providers. By defaulting to a locally running model, FastDoc AI ensures that no source code is transmitted to external services, which is important in professional and enterprise settings where code confidentiality is required.

The extension is designed to be lightweight, easy to install, and usable without deep configuration. A developer should be able to install it, have Ollama running, and generate their first documentation block within minutes.

## 3. Functional Requirements

This section specifies the behavioral requirements of the FastDoc AI extension. Each requirement is expressed as a user story and includes acceptance criteria and a MoSCoW priority rating. All requirements must be verifiable by the project lecturer.

### Must Have (FR-01 – FR-08b)

| ID           | User Story                                                                                                                                                                                                                  | Acceptance Criteria                                                                                                                                                                                                                                                                                    | Priority  | Owner  |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|--------|
| <b>FR-01</b> | As a developer, I want to generate a JSDoc comment or Python docstring for the function at my cursor position, so that I can document my code quickly without writing it manually.                                          | Given a function or method is at or near the cursor (defined as: the cursor is on the function signature line, or within 3 lines above or below it), when 'FastDoc: Generate Documentation' is triggered, a properly formatted block is inserted above the function without overwriting existing code. | Must Have | Atay   |
| <b>FR-02</b> | As a developer, I want the extension to detect the programming language of the currently open file, so that the correct documentation format is applied automatically.                                                      | Given a supported file is open (JS, TS, Python), when generation is triggered, the output format matches the file language — JSDoc for JS/TS, docstring for Python.                                                                                                                                    | Must Have | Atay   |
| <b>FR-03</b> | As a developer, I want to trigger documentation generation via the VS Code Command Palette, so that I can access the feature using only the keyboard.                                                                       | Given the extension is installed and a file is open, when I open the Command Palette and search 'FastDoc', the command appears and executes successfully.                                                                                                                                              | Must Have | Sheikh |
| <b>FR-04</b> | As a developer, I want the extension to detect existing documentation blocks.                                                                                                                                               | Given a function with an existing documentation block, when the developer opens or focuses on that function, then the extension visually indicates that documentation already exists.                                                                                                                  | Must Have | Atay   |
| <b>FR-05</b> | As a developer, I want to be able to regenerate and overwrite existing documentation, and be clearly informed when a function already has a documentation block, so that I can make an informed decision about updating it. | Given I am informed of existing documentation, when I choose to rewrite it, then the system generates a new documentation block and replaces the old one in full.                                                                                                                                      | Must Have | Sheikh |
| <b>FR-06</b> | As a developer, I want all generated documentation to include the function name, parameter names, parameter types (where inferable), and return type, so that the documentation is meaningful and complete.                 | Given a function with named parameters and a return statement, when documentation is generated, the output contains @param entries for each parameter and a @returns / :returns: entry.                                                                                                                | Must Have | Atay   |

| ID            | User Story                                                                                                                                                                                                                                                         | Acceptance Criteria                                                                                                                                                                                                                                                                                                                                         | Priority  | Owner  |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|--------|
| <b>FR-07</b>  | As a developer, I want the extension to be installable from the VS Code Marketplace, so that setup requires no manual download or configuration beyond the LLM backend.                                                                                            | Given the extension is published to the VS Code Marketplace, when a developer searches 'FastDoc', they can install it with one click and begin using it after configuring their LLM backend.                                                                                                                                                                | Must Have | Sheikh |
| <b>FR-08</b>  | As a developer, I want the extension to connect to an LLM (local or remote) for documentation generation, so that my documentation is generated automatically without requiring an internet connection by default.                                                 | Given an LLM backend is configured and reachable (local Ollama by default, or a remote provider via settings), when the command is triggered, the extension communicates with the configured LLM endpoint and returns a documentation block.                                                                                                                | Must Have | Sheikh |
| <b>FR-08b</b> | As a developer, I want the extension to detect when the configured LLM is unavailable or misconfigured (e.g. Ollama not running, or missing API key/token for a remote provider), and notify me clearly, so that I understand why documentation generation failed. | Given the LLM backend is unreachable or the required credentials (API key / token) are absent, when the command is triggered, the extension shows a clear human-readable error (e.g. 'Ollama is not reachable, please start Ollama and try again' or 'Remote LLM API key is not configured — please add it in Settings') and does not modify the open file. | Must Have | Sheikh |

### Should Have (FR-09 – FR-15)

| ID           | User Story                                                                                                                                                                                | Acceptance Criteria                                                                                                                                                                                                                   | Priority    | Owner  |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|--------|
| <b>FR-09</b> | As a developer, I want to configure the LLM provider (local Ollama or remote) and model name in the VS Code settings panel, so that I can customize the extension for different projects. | Given the extension is installed, when I search 'fastdoc' in VS Code Settings, I can view and modify provider type (local Ollama or remote), model name, and API endpoint URL or API key. Changes take effect on the next invocation. | Should Have | Sheikh |
| <b>FR-10</b> | As a developer, I want to see a loading indicator while documentation is being generated, so that I know the extension is working.                                                        | Given a generation request has been sent, when the response has not yet arrived, a status bar message or progress notification is shown in VS Code.                                                                                   | Should Have | Atay   |
| <b>FR-11</b> | As a developer, I want to generate documentation using a right-click context menu option in the editor, so that I have an alternative to the Command Palette.                             | Given a file is open and the cursor is on a function, when I right-click in the editor, a 'Generate Documentation' option appears in the context menu and triggers generation.                                                        | Should Have | Atay   |

| ID                          | User Story                                                                                                                                                                       | Acceptance Criteria                                                                                                                                                              | Priority    | Owner  |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|--------|
| <b>FR-1 2</b>               | As a developer, I want to be able to configure a new keyboard shortcut for documentation generation, so that I can trigger it faster without opening the Command Palette.        | Given the extension is installed, when I assign a shortcut in VS Code Keybindings for 'FastDoc: Generate Documentation', the command executes when that shortcut is pressed.     | Should Have | Sheikh |
| <b>FR-1 3</b>               | As a developer, I want the extension to sanitize the LLM output before inserting it, so that malformed or incomplete documentation blocks do not break my source file.           | Given the LLM returns an incomplete or malformed comment block, the extension detects this and either retries once or shows an error rather than inserting invalid content.      | Should Have | Sheikh |
| <b>FR-1 4</b>               | As a developer, I want the extension to log generation events and errors to the VS Code Output panel, so that I can debug issues without needing external tools.                 | Given any generation attempt (successful or failed), when the command runs, a timestamped entry is written to a dedicated FastDoc output channel.                                | Should Have | Atay   |
| <b>FR-1 5 (prev. FR-08)</b> | As a developer, I want to use a locally running Ollama LLM as the default documentation generation backend, so that my source code is never sent to external servers by default. | Given Ollama is installed and running with a configured model, when no remote provider is configured, the extension communicates exclusively with the local Ollama API endpoint. | Should Have | Sheikh |

### Could Have (FR-16 – FR-20)

| ID            | User Story                                                                                                                                                                | Acceptance Criteria                                                                                                                                                           | Priority   | Owner  |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|--------|
| <b>FR-1 6</b> | As a developer, I want to preview the generated documentation in a side panel before it is inserted, so that I can review and approve it first.                           | Given documentation has been generated, when the preview option is active, the output is shown in a VS Code webview panel with 'Insert' and 'Discard' buttons.                | Could Have | Atay   |
| <b>FR-1 7</b> | As a developer, I want to be able to manually edit the generated documentation in the preview side panel before approving it, so that I can correct or refine the output. | Given the preview panel is open with generated documentation, when the developer clicks on the documentation text, then the content becomes editable inline within the panel. | Could Have | Atay   |
| <b>FR-1 8</b> | As a developer, I want to choose between multiple generated documentation variants, so that I can select the most accurate one.                                           | Given the LLM is configured to generate alternatives, when documentation is triggered, up to 3 variants are presented in a Quick Pick menu and the selected one is inserted.  | Could Have | Sheikh |

| ID           | User Story                                                                                                                                                     | Acceptance Criteria                                                                                                                                                     | Priority   | Owner  |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|--------|
| <b>FR-19</b> | As a developer, I want the extension to show telemetry data locally (e.g., number of docs generated this session), so that I can track my usage.               | Given the extension is running, when I open the FastDoc status bar item, a summary shows how many documentation blocks have been generated in the current session.      | Could Have | Sheikh |
| <b>FR-20</b> | As a developer, I want the extension to remember the last successfully used model name per workspace, so that I do not need to reconfigure it on each project. | Given I previously generated documentation in a workspace with a specific model, when I open that workspace again, the same model is pre-selected without manual input. | Could Have | Atay   |

### Won't Have (FR-21 – FR-29)

| ID           | User Story                                                                                               | Acceptance Criteria                                                | Priority   | Owner |
|--------------|----------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------|------------|-------|
| <b>FR-21</b> | Support for programming languages beyond JavaScript, TypeScript, and Python (e.g., Go, Rust, Java, C++). | Out of scope — language coverage limited to JS, TS, Python in v1.  | Won't Have | —     |
| <b>FR-22</b> | Automatic documentation generation for an entire file or project in one action.                          | Batch generation is out of scope; generation is cursor-based only. | Won't Have | —     |
| <b>FR-23</b> | A graphical configuration UI within VS Code (beyond the standard settings panel).                        | The VS Code settings panel is sufficient for v1.                   | Won't Have | —     |
| <b>FR-24</b> | Automatic installation or management of Ollama on the user's machine.                                    | Ollama must be installed separately; bundling is out of scope.     | Won't Have | —     |
| <b>FR-25</b> | Documentation generation for anonymous functions, arrow functions, or lambdas.                           | Only named function declarations and methods are supported in v1.  | Won't Have | —     |
| <b>FR-26</b> | Sync or share generated documentation with external services (e.g., Confluence, Notion).                 | No external integrations are planned for v1.                       | Won't Have | —     |
| <b>FR-27</b> | Multi-language UI or localization of extension messages and notifications.                               | The extension will be English-only in v1.                          | Won't Have | —     |
| <b>FR-28</b> | Support for VS Code versions below 1.70.                                                                 | Compatibility is limited to VS Code 1.70 and above as per QS-C.    | Won't Have | —     |
| <b>FR-29</b> | Mandatory use of a local LLM as the only supported backend.                                              | Remote LLM providers are supported as an opt-in option from v1.    | Won't Have | —     |

## 4. Quality of Service Requirements

This section defines non-functional requirements that constrain how the extension must behave in terms of performance, security, compatibility, and maintainability. These requirements apply across all functional features.

| ID   | Category                       | Requirement Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Priority    |
|------|--------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|
| QS-A | Data Privacy                   | By default, the extension must not transmit any source code to external APIs or cloud services. All LLM processing must occur locally via Ollama as the default configuration. A local Ollama instance must be supported as a Must Have requirement. Users may explicitly opt into a remote LLM provider (e.g. via API key in settings), in which case the extension must clearly inform the user that code will be sent externally. This opt-in must be explicit and never the default.                                                                                                                                                                                                                                                                                                                                                                  | Must Have   |
| QS-B | Response Time                  | The extension must return a generated documentation block within 10 seconds of the command being triggered, assuming the LLM backend (local or remote) is running on or accessible from a standard developer machine (8 GB RAM, mid-range CPU). On slower hardware or high-latency remote connections the threshold may be exceeded, but the editor must remain responsive and not freeze or block user input during generation.                                                                                                                                                                                                                                                                                                                                                                                                                          | Should Have |
| QS-C | Platform Compatibility         | The extension must function correctly on Windows, macOS, and Linux. It must be compatible with VS Code version 1.70 and above. The extension must be tested on each platform as part of the development process before release.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Must Have   |
| QS-D | Error Handling & User Feedback | The extension must handle all foreseeable error conditions gracefully and without crashing or hanging the editor. Covered cases include: (1) Ollama not running or unreachable — show 'Ollama is not reachable, please start Ollama and try again'; (2) Remote LLM API key missing — show 'Remote LLM API key is not configured — please add it in Settings'; (3) No function detected at the cursor position — show 'No function found at cursor, please reposition and try again'; (4) Unsupported file type — show 'This file type is not supported by FastDoc'; (5) LLM returns malformed or empty output — show 'Documentation could not be generated, please try again'; (6) Request timeout — show 'Request timed out' and release the editor. In all cases the extension must use the VS Code notification API and must not modify the open file. | Must Have   |
| QS-E | Prerequisites & Documentation  | The extension README must document all prerequisite software including the minimum required Node.js version, Ollama setup, and instructions for configuring a remote LLM provider (if desired). Installation and first-use instructions must be clear enough that a developer unfamiliar with the project can have the extension running within 15 minutes by following the README alone, without external support.                                                                                                                                                                                                                                                                                                                                                                                                                                       | Must Have   |
| QS-F | Code Maintainability           | The extension codebase must follow consistent code style conventions enforced by a linter (e.g., ESLint). All functions and modules must have inline comments explaining their purpose. The project must include a package.json with clearly defined scripts for building, linting, and testing the extension.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Should Have |
| QS-G | Extension Startup Performance  | The extension must activate within 2 seconds of VS Code launching on a standard developer machine. It must not perform any network calls or heavy computation during the activation phase. Activation should be deferred until the first command invocation where possible.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Should Have |

| ID   | Category                       | Requirement Description                                                                                                                                                                                                                                                                                                         | Priority    |
|------|--------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|
| QS-H | Test Coverage                  | All core modules of the extension — including language detection, function parsing, LLM API communication (local and remote paths), and documentation insertion — must have unit tests. A minimum of 70% code coverage must be achieved and verifiable via the test script defined in package.json.                             | Should Have |
| QS-I | Accessibility of Notifications | All user-facing notifications, status bar messages, and error dialogs produced by the extension must use plain, readable language without technical jargon. Messages must be concise (under 120 characters where possible) and must not interfere with the user's active editor focus or require dismissal to continue working. | Could Have  |

## 5. Constraints and Assumptions

---

### 5.1 Constraints

- The extension must be implemented using the VS Code Extension API in TypeScript or JavaScript. No other editor platform or runtime is in scope.
- Ollama must be installed and managed separately by the user when used as the local LLM backend. The extension does not bundle, download, or configure Ollama on the user's behalf.
- The extension communicates with Ollama exclusively via its local HTTP API. Direct model inference inside the extension process is not permitted.
- If a remote LLM provider is configured, communication must occur via HTTPS only. The API key or token must be stored in VS Code's secure settings storage.
- All source code written for this project must be stored in a version-controlled Git repository. Releases must be tagged.
- The extension must not require administrator or root privileges to install or run on any supported operating system.
- The project must be completable by a two-person student team within the timeframe of the module. Features requiring disproportionate effort relative to their value must be deferred.

### 5.2 Assumptions

- The developer's machine has sufficient CPU and RAM to run a local Ollama model (e.g., llama3 or codellama) without significantly degrading the development workflow.
- The Ollama HTTP API remains stable and backward-compatible throughout the development period. Breaking API changes would require rework of the communication layer.
- Users have a basic familiarity with VS Code extensions and are comfortable installing and starting Ollama from the command line, or obtaining an API key from a remote LLM provider.
- The VS Code Extension API for the features used (editor text insertion, command registration, settings contribution, status bar) remains stable across VS Code versions 1.70 and above.
- Function definitions in supported languages (JS, TS, Python) can be reliably detected by scanning the lines surrounding the cursor (within 3 lines above and below) using regex or simple AST heuristics, without requiring a full language server.
- The LLM model (local or remote) has sufficient instruction-following capability to produce consistently formatted JSDoc or docstring output when given a structured prompt.
- Internet access is available during development for installing npm dependencies and publishing to the VS Code Marketplace, but is not required at runtime when using a local Ollama backend.

## 6. Out of Scope

The following items are explicitly excluded from this project. They are listed here to prevent scope creep and to document conscious decisions made during requirements analysis.

- A dedicated backend server or cloud infrastructure of any kind. FastDoc AI runs on the developer's local machine or communicates with a pre-existing remote LLM API.
- User authentication, licensing, or any form of account management. The extension has no concept of users or sessions.
- Persistent storage of generated documentation outside of the source file itself. No history, cache, or export functionality is planned.
- Documentation quality validation or correctness checking. The extension inserts LLM output as-is (after sanitisation) and does not assess whether the generated content is semantically accurate.
- Support for documentation formats other than JSDoc and Python docstrings, such as Doxygen, JavaDoc, TSDoc extended tags, or XML doc comments.
- Automated or scheduled documentation generation triggered by file saves, commits, or CI pipelines. All generation is strictly on-demand via explicit user action.
- A web-based or standalone version of the tool outside of VS Code. The product is a VS Code extension exclusively.
- Accessibility compliance testing (e.g., WCAG) for any UI elements introduced by the extension, beyond the plain-language notification requirement in QS-I.

## 7. Glossary

| Term                       | Definition                                                                                                                                                                                |
|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>VS Code Extension</b>   | A plugin for Visual Studio Code that extends its functionality using the VS Code Extension API.                                                                                           |
| <b>JSDoc</b>               | A markup language used to annotate JavaScript/TypeScript source code with structured comments describing functions, parameters, and return values.                                        |
| <b>Docstring</b>           | A string literal placed at the beginning of a Python function or class definition, used as inline documentation.                                                                          |
| <b>Ollama</b>              | An open-source tool that allows developers to run large language models locally on their own hardware without sending data to external services.                                          |
| <b>LLM</b>                 | Large Language Model — an AI model trained on large amounts of text data, capable of generating human-readable text based on a given prompt. May be local (Ollama) or remote (API-based). |
| <b>Remote LLM Provider</b> | An external cloud-based API offering LLM capabilities (e.g. OpenAI-compatible APIs) that requires an API key or token for access. Opt-in only in FastDoc AI.                              |
| <b>Command Palette</b>     | A VS Code UI feature (Ctrl+Shift+P / Cmd+Shift+P) that provides quick access to all extension commands by name.                                                                           |
| <b>MoSCoW</b>              | A prioritisation framework using four levels: Must Have, Should Have, Could Have, and Won't Have.                                                                                         |
| <b>AST</b>                 | Abstract Syntax Tree — a tree representation of the syntactic structure of source code, used here to detect function boundaries reliably.                                                 |
| <b>ESLint</b>              | A static analysis tool for JavaScript and TypeScript that enforces code style rules and catches common programming errors.                                                                |
| <b>Proximity Rule</b>      | The rule defining when a cursor is considered "near" a function: the cursor must be on the function signature line, or within 3 lines above or below it (FR-01).                          |