Lightning Talk

Doc Doctors

Rudra Patel, Farooq Khan, Vignesh Yampally,
Josh Lee, Thomas McNamara

Problem Statement

Challenges in Documentation:

- Keeping documentation up-to-date with rapid code changes is a common challenge in software development.
- Manual documentation is time-consuming and can become a secondary priority, leading to outdated or incomplete information.
- Inconsistencies in documentation can arise from multiple developers working on the same codebase.

Impact on Teamwork and Collaboration:

- Outdated documentation can cause miscommunication within teams, leading to errors and reduced efficiency.
- New team members may struggle with onboarding due to the lack of clear and accurate documentation.
- The overall quality of software can suffer without thorough and up-to-date documentation, impacting collaboration and the end product.

Proposed Solution

AutoDoc: Automating Documentation

1. Innovative Automation: AutoDoc leverages AI to automatically generate documentation for new code commits.
2. Real-time Updates: As code is committed, AutoDoc provides immediate documentation updates, including summaries, usage examples, and parameter descriptions.
3. Minimal Effort: Reduces the manual effort of documentation, ensuring it keeps pace with development cycles.

Aligning with Development Goals

1. Enhancing Efficiency: AutoDoc streamlines the documentation process, allowing developers to focus on coding rather than documenting.
2. Consistency and Clarity: Ensures consistent and clear documentation across all new commits, improving codebase readability.
3. Team Collaboration: Facilitates better team understanding and collaboration by providing up-to-date documentation for every change.

Use Cases

Use Case: Document Code for Team Sharing

Preconditions -

- Bob's code is undocumented.
- Both Bob and the recipient have AutoDoc access.

Main Flow

Bob submits code to AutoDoc [S1]. AutoDoc generates documentation [S2]. Bob sends documented code to a team member [S3].

Subflows

- [S1] Bob selects code for documentation in AutoDoc.
- [S2] AutoDoc creates comments and summaries.
- [S3] Documented code is shared with the team member.

Alternative Flows

[E1] AutoDoc requests clarification for ambiguous code.

Acceptance Criteria

Documentation accurately describes the code.
Team member understands the code with the provided documentation.

Non Functional Requirements

Usability - Clear Differentiation: Documentation generated by the tool will be distinctly marked to differentiate from manual comments.

Reliability - Accuracy Standard: The tool will achieve at least a 90% accuracy rate in the relevancy and correctness of the documentation it generates.
Performance:

Speed of Execution - The tool will generate documentation within 5 seconds, ensuring a swift development workflow.

Supportability - Language Updates: The system will support updates to accommodate newly created programming languages.

Implementation - Python-Based: The entire tool will be developed using Python for ease of integration with existing codebases and developer familiarity.

Functional Requirements

Automated Analysis: The tool automatically analyzes new code commits for documentation needs.

Test Case Suggestions: Suggests test cases to cover new functionality, aiding in thorough testing.

Non-Overwriting: Generates documentation without overwriting any existing documentation.

Multi-function Handling: Capable of handling multiple functions simultaneously for documentation.

Variable Ignorance: Identifies and ignores variables when generating documentation to focus on functional descriptions.

Relevance to SE Collaboration

Enhancing Team Communication:

- AutoDoc ensures that every team member has immediate access to the latest code documentation.
- Clear, auto-generated documentation minimizes misunderstandings and knowledge gaps.
- Facilitates onboarding of new team members by providing comprehensive code insights.

Improving Collaboration:

- Promotes effective peer code reviews by providing context with up-to-date documentation.
- Supports distributed teams by synchronizing understanding across different geographies.
- Encourages collective code ownership and continuous knowledge sharing within the team.

Acceptance Criteria for Effective Collaboration:

- Team members report reduced time deciphering code functionalities.
- Code reviews become more efficient, with fewer questions about basic function purposes.
- New team members integrate quicker with less dependency on other team members for codebase understanding.

Risks and Mitigations

Security Risks:

- **Data Protection:** Implementing authentication tokens to ensure only authorized users can access the AutoDoc system.
- **Code Privacy:** Use of secure channels for analyzing code to prevent exposure or leaks of sensitive information.
- **Access Control:** Role-based access controls to prevent unauthorized modification of documentation.

Malfunction Risks:

- **Error Handling:** Incorporating comprehensive error handling to catch and log exceptions without disrupting the user experience.
- **Backup Systems:** Creating backups of original code before any documentation is added to prevent loss in case of a malfunction.
- **Regular Updates:** Scheduled maintenance and updates to the AutoDoc system to address new bugs and vulnerabilities as they are discovered.

Acceptance Criteria for Risk Mitigation:

- No reported security breaches due to unauthorized access.
- Error rates are kept below a pre-defined threshold.
- System recovery measures are tested and validated for effectiveness.