Agile:

User Story 1:

As a seasoned Git user unfamiliar with GiggleGit, I want to quickly grasp and experience meme-based merges so I can determine if it fits my workflow.

Task:

API Integration and Demo: Showcase a functional meme-based merge.

Ticket 1: Securely Configure Environment (including API keys).

Ticket 2: Develop and Document Meme-Driven Merge Demo

User Story 2

As a team lead onboarding an experienced GiggleGit user, I want to go through a guided onboarding process that emphasizes the advantages and unique aspects of meme-managed merges, so that I can effectively train my team and feel confident using the new system.

Task:
Onboarding Walkthrough: Develop a step-by-step guide integrated into the demo interface.

Ticket 1: Create Onboarding Tutorial
Design an interactive tutorial that guides users through the key features and benefits of meme-managed merges.

Ticket 2: Feedback Collection Mechanism
Set up a system to gather user feedback during onboarding to identify areas for improvement and refine the process.

User Story 3

As a curious developer at CodeChuckle, I aim to track and record the results of merge operations based on memes to assess their efficiency and dependability for future enhancements. Tasks include:

Setting Up Merge Logging System: Develop infrastructure to capture detailed logs for each merge operation.

Ticket 1: Implement Merge Logging - Design a logging system that documents metadata for each merge, such as timestamps, user IDs, and the outcomes of the merges.

Ticket 2: Create Merge Analytics Dashboard - Construct a simple dashboard to display the logged merge data and performance statistics over time.

4.
This statement lacks context and a clear benefit. A complete user story should include who the user is, what they need to do, and why they need it or the value.

Formal Requirements:

1.Goal and Non-goal

Goal:
Integrate SnickerSync into Gigglegit so that users can visualize code differences with a twist during merge reviews.

Non-goal:
Change the entire GiggleGit user interface for the sake of the Snickersync feature.

2.Non-Functional Requirement 1: Security & Access Control
Only personnel shall be able to access and modify SnickerSync diff logs and configurations.

Functional Requirement 1: The system must enforce role-based access control or accessing SnickerSync logs, ensuring that only authenticated and authorized users can view detailed diff data.

Functional Requirement 2:
All data transmitted by SnickerSync must be encrypted in transit and at rest to maintain confidentiality and integrity.

3.Non-Functional Requirement 2: Maintainability & Scalability

Non-functional Requirement:
SnickerSync must be maintainable and scalable to support continuous user studies and accommodate future feature expansions.

Functional Requirement 1:
PMs must be able to modify diff parameters without requiring downtime or redeployment, using a configuration management system.

Functional Requirement 2: The system must support random user assignments between control and variant groups during user studies, automatically logging user assignments and interaction metrics for subsequent analysis.