Agile

User Story 1: As a vanilla git power-user that has never seen GiggleGit before, I want to quickly understand how GiggleGit's meme-based merge system differs from the traditional git merges. This is to efficienty evaluate its potential benefits for my workflow.

User Story 2: As a team lead onboarding an experienced GiggleGit user, I want to easily assign appropriate memes to each team member, ensuring efficient and humorous collaboration within our project.

User Story 3: As a developer new to version control, I want to learn GiggleGit's basic operations through an interactive tutorial, so I can start contributing to my team's codebase with confidence and humor.

> **Task:** Create an interactive GiggleGit tutorial
>
> **Ticket 1:** Design the tutorial framework
> Develop a step-by-step tutorial framework that guides new users through basic GiggleGit operations. Include interactive elements that allow users to practice commands and meme selections in a safe, sandboxed environment. Ensure the tutorial covers key concepts like committing, branching, merging, and resolving conflicts with memes.
> **Ticket 2:** Implement meme selection minigame
> Create a fun, interactive minigame that teaches users how to select appropriate memes for different merge scenarios. Include a variety of merge conflict types and corresponding meme categories. Provide instant feedback on meme choices to help users understand the impact of their selections on the merge process.

Regarding the statement "As a user I want to be able to authenticate on a new machine": This is not a user story because:

1. It lacks the full "As a [role], I want [goal/desire] so that [benefit]" structure.
2. It doesn't provide context about the user's role or the benefit they're seeking.
3. It's focused on a specific technical implementation (authentication) rather than a user-centric goal.

This statement is more similar to a feature requirement or a technical task. It describes a specific functionality without providing the context of why it's important to the user or how it fits into the larger user experience.

Formal Requirements

Goal: Create an engaging and intuitive diff tool that enhances the code review process through humor-based interactions.

Non-Goal: Develop a comprehensive meme creation platform within SnickerSync.

Non-functional Requirements

1. User Access Control

Functional Requirements:

1. Implement role-based access control (RBAC) to manage user permissions for different features of SnickerSync.
2. Provide secure authentication mechanisms, including multi-factor authentication, for all user accounts.

2. Experimental Design Flexibility

Functional Requirements:

1. Develop a randomization system that assigns users to control and experimental groups for A/B testing of SnickerSync features.
2. Create a configurable interface for PMs to define and modify different "snickering concepts" without requiring code changes.