Agile section:

User Stories:

- As a vanilla git power-user that has never seen GiggleGit before, I want to get started with GiggleGit without feeling overwhelmed or confused so that I can understand how the tool integrates with my usual git workflow.

- As a team lead onboarding an experienced GiggleGit user, I want to ensure my team members can easily collaborate using GiggleGit with the new merge system, so that our workflows remain smooth while also using this new tool.

Third User Story:

- As a project manager overseeing the adoption of GiggleGit by my team, I want to track the team's progress and identify potential issues during the onboarding process, so that I can ensure the transition to GiggleGit is successful without disrupting the project's timelines.

Task for the Third User Story

- Set up tracking and analytics for onboarding progress.

Tickets for the Task

1. Implement Progress Tracking Dashboard
   Create a simple dashboard that displays each team member's status within the GiggleGit onboarding process. Ensure that data is collected while the user is in the process of onboarding and displayed in a clear, easy to understand format.

2. Set up Notification System for Onboarding Delays
   Build a notification system that triggers alerts for the project manager when a team member is stuck on an onboarding step for an extended period. Notifications can be delivered by email.

This is not a user story. Why not? What is it?

- As a user, I want to be able to authenticate on a new machine.

This is not a user story because it doesn't explain the "why" or the value behind it. A well written user story focuses on the user's goal and the benefit they seek to gain, not just the feature they want.

What it is:
This statement describes a feature request rather than a user story.

Formal Requirements Section

1. One Goal and Non-Goal

Goal:

- Ensure that SnickerSync's syncing process enhances user engagement by making the merge experience more enjoyable and less stressful through humor.

Non-Goal:

- SnickerSync does not try to make new or alter the core functionality of Git, such as the version history or git commands like “merge”, “rebase”, etc.

2. Non-Functional Requirements

Non-Functional Requirement 1:

- The system must ensure that only users with appropriate permissions can access and modify the different "snickering" concepts in SnickerSync.

Functional Requirement 1.1:

- Implement role-based access control to allow only authorized users to manage SnickerSync’s snickering concepts.

Functional Requirement 1.2:

- Create an admin panel where project managers can review, add, edit, or delete snickering concepts while restricting access to regular users.

Non-Functional Requirement 2:

- The user study system must ensure the random assignment of users into control and experimental groups, with no overlap between groups.

Functional Requirement 2.1:

- Develop a backend algorithm that automatically assigns users to either the control group or the experimental variant group in a randomized and non-biased manner.

Functional Requirement 2.2:

- Create a monitoring tool that tracks user group assignments, ensuring that each user is placed into only one group and that the distribution remains balanced across multiple study iterations.