Requirements

**Goal**:

Enable user studies on SnickerSync by randomly assigning users between control groups and variants to evaluate the effectiveness of syncing with a snicker.

**Non-Goal**:

Ensure that all users prefer syncing with a snicker over other version control tools (this is a metric to be tested, not a predetermined outcome).

**Non-Functional Requirement 1**: Access Control

1. Functional Requirement:

Provide PMs with admin-level access to a control panel where they can create, edit, and manage various "snicker syncing" concepts used in the user studies.

1. Functional Requirement:

Implement role-based access control (RBAC), where PMs can view all data related to user study groups and results, while regular users only have access to their own sync-related data and snickering features.

**Non-Functional Requirement 2**: Random Assignment

1. Functional Requirement:

Develop a random assignment algorithm that evenly distributes users across control and variant groups each time they participate in a study to ensure a balanced sample size between groups.

1. Functional Requirement:

Store group assignments and study results in a database to ensure the same user is consistently assigned to the same group for the duration of a study, allowing for long-term analysis and repeatability.

**Theme:** Get GiggleGit demo into a stable enough alpha to start onboarding some adventurous clients.

**Epic:** Onboarding experience

**User Story 1:**

As a vanilla git power-user that has never seen GiggleGit before, I want to have access to basic git commands with minimal interference from the meme features, so I can use the version control system efficiently.

**Task**: Provide core git functionality.

**Ticket 1:** Implement core git commands

Ensure that all basic git commands (e.g., git pull, git push, git commit, git merge) are available and function exactly like they do in vanilla git to avoid confusion for power-users who are unfamiliar with the meme features of GiggleGit.

**Ticket 2:** Add option to disable meme features

Add a setting that allows the user to toggle meme integration on or off, allowing them to use GiggleGit in a purely "git-like" mode if they choose to do so.

**User Story 2:**

As a team lead onboarding an experienced GiggleGit user, I want to provide a smooth introduction to the meme features for the rest of my team, so they understand how to use the system but aren’t distracted by memes during important tasks.

**Task**: Provide onboarding documentation.

**Ticket 1:** Create an onboarding guide

Develop documentation that outlines the core differences between GiggleGit and vanilla git, focusing on how the meme-based merge system works. Include a step-by-step guide with screenshots to help new users understand the meme integration.

**Ticket 2:** Create a quick-start tutorial video

Produce a short video (3-5 minutes) that visually demonstrates how to use GiggleGit’s meme-based features in real-world team collaboration scenarios. Ensure the tutorial is concise and accessible for all levels of git experience.

**User Story 3:**

As a developer trying to troubleshoot a merge conflict, I want to understand how the meme-based resolution works, so I can resolve the conflict efficiently without breaking the codebase.

**Task**: Create a merge conflict resolver.

**Ticket 1:** Design the meme-based merge conflict interface

Design the UI for resolving merge conflicts with memes. The interface should present conflicting code alongside humorous, relevant memes, while still allowing the user to choose which lines of code to keep or discard.

**Ticket 2:** Integrate meme suggestions into merge conflicts

Implement an algorithm that selects appropriate memes based on the nature of the merge conflict. For example, a meme about bugs could appear when there are issues with conflicting code in a test file.

This statement lacks context and specificity that a proper user story should provide. A good user story should describe who the user is and why they want the feature. This helps ensure that the development team understands the user’s motivation and can prioritize the work appropriately. A better version might be: "As a developer setting up a new work environment, I want to authenticate my account on a new machine quickly, so I can continue my work without delay."