Agile Requirements (CodeChuckle):

User Story 1: As a vanilla git power-user that has never seen GiggleGit before, I want to quickly understand the differences between GiggleGit and standard git in order to easily start using GiggleGit's advanced features to achieve productivity that is just as good if not better than that of using vanilla git.

User Story 2: As a team lead onboarding an experienced GiggleGit user, I want to have all of the current team project's GiggleGit features and permissions instantly set up for my new team member in order to allow them to start making progress on the project as quickly as possible.

User Story 3:

As a novice software developer who has never even used vanilla git or any other VCS before, I want to learn and grasp the basics of version control with GiggleGit through an interactive tutorial, in order to start implementing version control in my own projects using GiggleGit.

Task: Develop an Interactive Tutorial to teach VCS through GiggleGit.

- **Ticket 1**: Write the step-by-step instructions that will lead the user through the interactive tutorial.
- **Ticket 2**: Implement interactive exercises and quizzes throughout the tutorial that act as learning checkpoints.

Not a user story:

"As a user I want to be able to authenticate on a new machine" is not a user story because firstly it does not refer to a specific actor, it merely says "user" which is very general. It also doesn't state any benefit to being able to authenthicate on a new machine, which is required for a user story.

Formal Requirements (SnickerSync User Testing)

Goal: Conduct comprehensive user testing for SnickerSync to ensure it functions effectively as a diff tool and integrates seamlessly with GiggleGit.

Non-Goal: User testing will NOT focus on GiggleGit's base features and will only evaluate SnickerSync's unique features.

Non-Functional Requirement 1: Usability

1. **Functional Requirements:**
    a. Develop diverse testing scenarios that cover all typical use cases of SnickerSync to make sure we get comprehensive feedback.

b. Implement a system for collecting and organizing user feedback, which could include surveys, focus groups, and live recording during testing sessions.

Non-Functional Requirement 2: Reliability

1. **Functional Requirements:**
    a. Perform stress testing to evaluate SnickerSync's performance when faced with high load and rapid inputs to ensure it maintains functionality without crashing or losing data.
    b. Test the system’s ability to recover from crashes or errors effectively, ensuring that it can return to a known good state without manual intervention, thus maintaining its reliability.