User Story 1:

As a vanilla git power-user that has never seen GiggleGit before, I want to quickly understand the interface and key differences, so I can confidently merge code using memes.

Task: Create a quick start guide highlighting key features.

Tickets:

- **1) Write Quick Start Guide**: Create a detailed guide with visuals explaining GiggleGit's interface and how merges are controlled with memes.
- **2) Add Quick Start to Demo**: Integrate the quick start guide into the demo environment for easy access.

User Story 2:

As a team lead onboarding an experienced GiggleGit user, I want to ensure they can quickly access training resources so they can onboard new team members themselves.

Task: Create an onboarding training module for team leads.

Tickets:

- **Build Training Module**: Develop a comprehensive training module for experienced users to onboard new team members.
- **Add Progress Tracking**: Include functionality to track completion of the training module for team leads.

User Story 3:

As a project manager evaluating GiggleGit for my team, I want to see how it handles conflicts during merges so I can assess if it's a suitable replacement for our current VCS.

Task: Develop a conflict resolution demo.

Tickets:

- **Build Conflict Demo**: Implement a demo showcasing how meme-driven conflict resolution works in GiggleGit.
- **Test Conflict Scenarios**: Run tests on different merge conflict scenarios to ensure the demo works flawlessly.

Non-User Story:

"As a user I want to be able to authenticate on a new machine" is a **system requirement** rather than a user story. It describes a feature but lacks specific user context or benefit, which are key components of a user story.

Formal Requirements:

Goal:

Ensure SnickerSync enables smooth, intuitive syncing while incorporating a humorous, meme-driven interface that enhances user engagement.

Non-Goal:

Avoid redesigning the entire interface to cater to highly specific edge cases that diverge from the core SnickerSync experience.

Non-Functional Requirement 1:

Only administrators can modify and maintain the snickering concepts and syncing algorithms.

Functional Requirement 1.1: Administrators must have a designated interface to edit snickering rules and sync settings without impacting user data.

Functional Requirement 1.2: Changes made by administrators to snickering rules must trigger an automatic version history log that records who made changes and when.

Non-Functional Requirement 2:

The user study platform must support random assignment of users into control groups and experimental groups for testing different snickering experiences.

Functional Requirement 2.1: The system must assign users to groups randomly and log the group assignment to ensure a balanced distribution across control and experimental variants.

Functional Requirement 2.2: The platform must generate reports that compare user interactions across different snickering groups, providing data on performance metrics and user engagement.