Formal Requirements

Project Requirements

- **Goal**: Create a stable and engaging user experience for SnickerSync, allowing users to enjoy the humor while managing merges effectively.
- **Non-Goal**: Implement machine learning algorithms to determine the most humorous merge automatically.

Non-Functional Requirement 1: Maintainability

- **Functional Requirements**:
    1. **Modular Code Structure**:
        - Break down the SnickerSync tool into modules such that each module handles a specific part of the syncing and merging functionality. This ensures that the codebase is easy to maintain and extend in the future.
    2. **Code Documentation**:
        - Ensure that each function and module has comprehensive documentation, allowing new developers to understand and modify the code effectively.

Non-Functional Requirement 2: Security

- **Functional Requirements**:
    1. **Role-Based Access Control (RBAC)**:
        - Implement role-based access control to ensure only authorized users (such as Product Managers) can modify SnickerSync settings. Developers should have restricted access, ensuring sensitive configurations are secure.
    2. **Audit Logging**:
        - Maintain an audit log that records every change made to the SnickerSync configurations, including details such as user ID, timestamp, and the nature of the change to track potential security issues.

Agile

Theme

- **Get GiggleGit demo into a stable enough alpha to start onboarding some adventurous clients**

Epic

- **Onboarding experience**

User Stories and Tasks

1. **User Story 1**: As a vanilla git power-user that has never seen GiggleGit before, I want to have clear, step-by-step documentation on how to install and use the GiggleGit tool so that I can use it effectively without spending too much time figuring it out.
    - **Task**: Create user documentation for GiggleGit
        - **Ticket 1**: Write installation guide
            - Provide a clear step-by-step guide for installing GiggleGit, covering multiple platforms.
        - **Ticket 2**: Create usage documentation
            - Write a detailed guide covering common Git workflows with GiggleGit's unique features.
2. **User Story 2**: As a team lead onboarding an experienced GiggleGit user, I want to have a set of tutorial materials or onboarding guides that I can use to train my team members, ensuring everyone can use GiggleGit's unique features consistently.
    - **Task**: Develop onboarding materials for team training
        - **Ticket 1**: Create tutorial videos
            - Develop video tutorials demonstrating how to use GiggleGit's main features.
        - **Ticket 2**: Create quick-reference guides
            - Develop concise reference documents that team leads can distribute to their members.
3. **User Story 3**: As a curious developer interested in trying new version control tools, I want to be able to play around in a sandbox environment without affecting any of my real projects, so I can understand GiggleGit's unique features in a risk-free way.
    - **Task**: Create a sandbox environment for new users
        - **Ticket 1**: Set up isolated sandbox environment
            - Develop a sandbox environment where users can try out GiggleGit without affecting their current repositories.
        - **Ticket 2**: Add onboarding documentation to the sandbox
            - Create comprehensive documentation that explains the features of GiggleGit and how to try them in the sandbox environment.

Analysis of the Given Statement

- **Statement**: ″As a user, I want to be able to authenticate on a new machine.″
    - This is not a user story because it lacks a "why"—a complete user story should explain the user's motivation or the value they get from the feature. It could be improved to: "As a user, I want to be able to authenticate on a new machine so that I can easily access my projects from any device."