Agile

- 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 seamlessly transition from my traditional git workflow to GiggleGit.
    - Task: Simplify onboarding steps for first-time users.
        - Ticket 1: Create an onboarding guide
            - Write a clear, step-by-step guide that explains how GiggleGit works compared to traditional Git, focusing on key similarities and differences.
        - Ticket 2: Add in-app tooltips for essential actions
            - Implement tooltips in the interface to explain meme-driven merges and other key features.
- User Story 2: As a team lead onboarding an experienced GiggleGit user, I want to easily set up permissions and roles for my team.
    - Task: Implement team permission controls.
        - Ticket 1: Develop role-based access control
            - Create a system where team leads can assign roles to members, giving them permissions to approve or suggest meme-driven merges.
        - Ticket 2: Add a UI for managing team permissions
            - Build a user interface that allows team leads to add/remove members and assign roles with ease.
- User story 3: As a developer who loves a bit of humor, I want the option to personalize my meme collection, so I can inject more fun into my merge experience.
    - Task: Enable customizable meme collections.
        - Ticket 1: Design a meme library interface
            - Create a library where users can upload or choose memes to use during merges.
        - Ticket 2: Implement meme preferences in merge settings
            - Add functionality where users can set preferences for meme categories or specific memes to be used during merge conflicts.

The line: "As a user I want to be able to authenticate on a new machine" is not a user story because it lacks a user-centered motivation or outcome. Instead, I believe it should describe why the user wants to authenticate on a new machine and what benefit it provides.

Requirements

- Goal: Ensure the SnickerSync tool provides a seamless experience for syncing and merging code, enhancing the core functionality of GiggleGit.
- Non-Goal: Developing an AI-driven system that automatically generates or selects snickers based on code changes.
- Non-functional requirement 1: Security and Access Control
    - Functional requirements:
        - All users should be required to log in with proper credentials to access SnickerSync.
        - Implement role-based access control such that only PMs and designated administrators should have the ability to create and manage snickering concepts.
- Non-functional requirement 2: Experimentation and User Studies
    - Functional requirements:
        - The system must include a dashboard where PMs can view and track user behaviors and preferences based on their group assignment.
        - Users should be randomly assigned to either the control group (no snickering) or the experimental group (snickering variations) when participating in the study.