Project Description: GiggleGit is a version control system where merges are managed with memes. It tries to provide a fun and engaging way for developers to collaborate while maintaining the core function of Git. Instead of traditional merge messages, users interact with meme-based diff and merge processes to resolve conflicts.

Agile Requirements for GiggleGit

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

Epic: Onboarding experience

User Stories:

1. As a vanilla git power-user that has never seen GiggleGit before, I want to quickly understand how it works so I can start using it.
   Task: Provide an introduction for new users.
   Ticket 1: Creating a getting started guide
   -Write a step-by-step guide explaining GiggleGit basics and how it differs from Git.
   -Include example workflows and meme-based merge explanations.
   Ticket 2: Implement a guided onboarding flow
   -Develop an interactive tutorial that walks users through their first merge.
   -Include ltips and hints within the UI.
2. As a team lead onboarding an experienced GiggleGit user, I want to have an official best practices guide so my team can work efficiently.
   Task: Document best practices for teams.
   Ticket 1: write a “best practices” doc
   -Outline recommended workflows for teams using GiggleGit.
   -Include troubleshooting tips and common
3. As a new user, I want to be able to undo a merge so I don’t get stuck with a bad meme-based merge
   Task: Implement rollback for meme-based merges.
   Ticket 1: Add support for merge rollback
   -Create a feature so users can revert a meme-based merge.
   -Ensure rollback maintains previous commit integrity.
   Ticket 2: Display merge rollback history
   -Show a list of past merges that can be rolled back.
   -Provide timestamps and user information for each merge.

This is not a user story. Why not? What is it?

- **As a user I want to be able to authenticate on a new machine**

As a user, I want to be able to authenticate on a new machine" is not a valid user story because it does not explain the reason or benefit behind the action. A valid user story should include the purpose, such as ensuring security or protecting private information.

Project Description:
SnickerSync is a diff tool in GiggleGit that helps users compare and approve code changes. It makes the process more fun by adding memes.
Formal Requirements for SnickerSync

Goal: Ensure that SnickerSync provides a reliable diffing experience that helps users collaborate effectively.

Non-Goal: SnickerSync will not generate custom memes for diffs.

Non-Functional Requirement 1: Access Control - Only authorized users should be able to approve or modify diffs.

- Functional Requirement 1.1
    - Implement role-based access control where only some rolls can approve or reject diffs.
- Functional Requirement 1.2
    - Provide audit logs showing who accessed or modified a diff.

Non-Functional Requirement 2: User Study Implementation

- Functional Requirement 2.1
    - Develop a system that randomly assigns users to different variants.
- Functional Requirement 2.2
    - Log user interactions while keeping identities anonymous.