## Agile

**Q1. Complete these user stories:**

As a vanilla git power-user that has never seen GiggleGit before, I want to experience GiggleGit's tools so I can understand the differences between GiggleGit and Git to know if it is worth moving my projects.

As a team lead onboarding an experienced GiggleGit user, I want to understand what they value and look for in merging and how they would prefer to handle their merge conflicts so I can ensure that they can seamlessly integrate their project using GiggleGit.

**Q2. Create a third user story, one task for this user story, and two associated tickets.**
**User Story:** As a project manager at a software company, I want to understand how GiggleGit will benefit my team's communication and collaboration, so I can better assess if I should adopt it into our workflow.

**Task:** Evaluate how GiggleGit contributes to better teamwork and communication.

**Tickets:**

1. Analyze how GiggleGit improves communication

   Research how GiggleGit can allow for better communication between team members. Access whether features like meme-based merging allows for clear and useful understanding of changes made. Determine if GiggleGit features lead to fewer miscommunications.

2. Analyze how GiggleGit promotes better teamwork

   Examine how GiggleGit allows for team collaboration through different unique features such as meme-based merging. Survey how GiggleGit affects productivity and conflict resolution speed. Explore how GiggleGit facilitates increased collaboration and if it makes it easier for members to share information and seek help.

**Q3. This is not a user story. Why not? What is it?**

This is not a user story because it is missing a "why." We don't know why the user wants to be able to authenticate on a new machine. Because of this, this is more of a functional requirement than it is a user story.

## Formal Requirements

Q1. **List one goal and one non-goal**

Goal: Create a user-friendly interface that allows users to easily view differences between branches in order to review and resolve differences.

Non-Goal: Do not focus on optimizing algorithm for comparing branches

**Q2 & Q3: Create two non-functional requirements. For each non-functional requirement, create two functional requirements (for a grand total of four functional requirements)**

Non-Functional Requirement 1: Security

- Functional Requirements:
    - Require user authentication when merging branches
    - Ensure that all data exchanged during uploading, syncing, and merging are encrypted

Non-Functional Requirement 2: Usability

- Functional Requirements:
    - Interface must be easy to navigate such that a tutorial is not needed for new users
    - Different interfaces for different roles for easier accessibility to what each member needs (e.g. simplified UI for a project manager that eliminates unnecessary technical details)