Agile Development: GiggleGit

1. User Story: As a vanilla git power-user that has never seen GiggleGit before, I want to easily understand how GiggleGit's meme-based merge system relates to traditional git merges, so that I can quickly adapt my existing workflow to GiggleGit.
2. User Story: As a team lead onboarding an experienced GiggleGit user, I want to have access to advanced configuration options and team management tools, so that I can efficiently set up our team's GiggleGit environment and customize it to our needs.
3. User Story: As a new GiggleGit user with limited git experience, I want a guided tutorial that introduces me to both basic git concepts and GiggleGit's unique features, so that I can start using the system productively without feeling overwhelmed.

   Task for the third user story:

   - Develop an interactive, step-by-step tutorial for new users

   Ticket 1:

   Title: Create tutorial content outline

   Details: Develop a comprehensive outline for the guided tutorial, covering basic git concepts (commit, branch, merge) and GiggleGit's unique features (meme-based merges, etc.). The outline should include key learning objectives for each section and a logical progression of topics.

   Ticket 2:

   Title: Implement interactive tutorial interface

   Details: Design and implement an interactive interface for the tutorial within the GiggleGit application. This should include progress tracking, interactive examples where users can practice concepts, and the ability to pause and resume the tutorial. Ensure the interface is intuitive and visually appealing, incorporating GiggleGit's meme-based theme where appropriate.

3. The reason why this example (“As a user I want to be able to authenticate on a new machine”) is not a user story is because it lacks the context of who the user is (e.g., developer, team lead, etc.). It doesn't provide the benefit or value to the user (the "why") and it's too focused on a specific implementation detail rather than the user's goal. In return, this is instead a feature request or a functional requirement. It describes a

specific functionality that the system should have, but it doesn't frame it from the user's perspective or explain the value it provides to the user.

Formal Requirements: SnickerSync

Goal:

To evaluate user engagement and efficiency when using the SnickerSync diff tool compared to traditional diff tools.

Non-goal:

To replace or deprecate existing diff tools in the GiggleGit ecosystem.

Non- Functional Requirement:

1. Security and Access Control:
   The SnickerSync tool must enforce role-based access control to ensure that only authorized personnel can access and modify core snickering concepts and user study data.
2. Randomized User Assignment:
   The system must support randomized assignment of users to control groups and variant groups for unbiased user studies.

Functional requirements:

For Security and Access Control:

1.1 User Authentication:

The system shall require users to authenticate using their GiggleGit credentials before accessing the SnickerSync tool.

1.2 Role-based Permissions:

The system shall provide different levels of access based on user roles:

- Regular users can only use the SnickerSync tool for diff operations
- PMs have additional permissions to modify snickering concepts
- Researchers have access to anonymized user study data

For Randomized User Assignment:

2.1 Automatic Group Assignment:

Upon a user's first interaction with SnickerSync, the system shall automatically assign the user to either a control group (using the vanilla interface) or a variant group (using the SnickerSync interface) using a randomization algorithm.

2.2 Group Tracking:

The system shall maintain a record of each user's group assignment (control or variant) throughout the duration of the user study, ensuring consistent experiences across multiple sessions.