Part 1: Agile Theme, Epic, and User Stories

- **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 a smooth transition from traditional Git commands to GiggleGit, with clear documentation and minimal disruptions to my existing workflow, so that I can get started quickly without extensive learning.

2. As a team lead onboarding an experienced GiggleGit user, I want to see detailed documentation on best practices and setup configurations so that I can efficiently train new team members and avoid mistakes during their onboarding process.

Third User Story:

3. As a project manager, I want to track the progress of the GiggleGit adoption within my team by accessing usage statistics and user feedback so that I can monitor the team's efficiency and make informed decisions.

Tasks for the Third User Story: Implement a dashboard to display user statistics and feedback.

Associated Tickets:

- **Ticket 1: Create a dashboard for usage statistics**
    - Develop a web-based dashboard that aggregates data on individual users' activity within GiggleGit, including commit, merges, and meme usage. Ensure the dashboard is accessible to team leads and project managers.

- **Ticket 2: Collect and store user feedback**
    - Create a feedback collection form within the GiggleGit interface that allows users to rate their experience and submit suggestions for further improvement. After that, store the gathered feedback/data in a database accessible to the dashboard.

Part 2: Formal Requirements for SnickerSync

Goal:

- Ensure the new SnickerSync tool is easy to use and offers an engaging user experience during version control operations

Non-Goal:

- Develop new version control algorithms to improve merge accuracy. (We want to focus on enhancing the user experience with SnickerSync, not on altering the entire core Git functionality)

Non-Functional Requirements:

1. ***User permissions for access control:***
    - Only authorized users should be able to access and modify SnickerSync settings, ensuring that junior team members or external users cannot disrupt version control processes.
2. ***Performance expectations for snickering interactions:***
    - SnickerSync must load and sync within 2 seconds of triggering to ensure that it does not interfere with normal Git workflow efficiency.

Functional Requirements:

1. ***For User permissions:***
    - Create a role-based access system where only admin-level users can modify SnickerSync settings.
    - Ensure that any changes to SnickerSync settings are logged and can be reviewed by team leads or system administrators.
2. ***For Performance expectations:***
    - Optimize SnickerSync's initialization code to meet the 2-second response time, ensuring that it integrates smoothly with the Git process.
    - Set up an automated performance test suite that monitors SnickerSync's speed and notifies the team when it exceeds the 2-second threshold.