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 who has never seen GiggleGit before, I want to get a tutorial that shows me how to use GiggleGit compared to git power
    - Task: create a comparison tutorial
        - Ticket 1: Create a flowchart of how the tutorial should go
            - Create a flowchart that will model each step of getting acclimated to GiggleGit
        - Ticket 2: Create a UI for the tutorial
            - Develop the UI that GiggleGit customers will use.
- User Story 2: As a team lead onboarding an experienced GiggleGit user, I want to create a personalized onboarding session for experienced GiggleGit users.
    - Task: create a unique onboarding experience for those familiar with GiggleGit
        - Ticket 1: Schedule onboarding sessions
            - Have a calendar for the times experienced users will be onboarded
        - Ticket 2: Take reviews of the onboarding experience
            - Take down reviews of the onboarding process to know how it could be improved
- User Story 3: As a first-time user I want a specific meme to pop up when a merge conflict occurs.
    - Task: Generate the same meme every time a merge conflict occurs
        - Ticket 1: Choose the meme that would make sense for a merge conflict
            - Ensuring that a meme that makes sense pops up when a conflict occurs
        - Ticket 2: Ensure the system can catch a merge conflict
            - Determining whether or not a merge conflict has occurred to generate the merge conflict meme
- 3 is not a user story because it is not a feature you want, it is more like a functional requirement you would need to implement for a non-functional requirement

Formal Requirements

Goal: Test the user experience of syncing merge conflicts with SnickerSync, assessing how users react to the functionality of SnickerSync

Non-Goal: Allowing third-party systems to interact with SnickerSync for custom sounds

Non-functional requirement 1: Security

- Functional Requirements
    - Secure user files so no one else can get access to them
    - Only users who have permission should be able to access files

Non-functional requirement 2: Testability

- Functional Requirements

- SnickerSync must handle random assignments of users to test their reactions to the program
- SnickerSync must ensure that the randomization process is unbiased to make sure it is getting good data