Agile

- 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 that has never seen GiggleGit before, I want to get a walk-through on how to use GiggleGit in order to get the most benefit out of it.
    - Task: Create an accessible and user-friendly tutorial
        - Ticket 1: Create visuals to depict basics of GiggleGit
            - We need to break down the basics of GiggleGit with the use of a diagram or flowcharts/
        - 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 guide the team to transitioning to a new software by providing onboarding sessions.
    - Task: Create multiple onboarding sessions for people less technologically-inclined
        - Ticket 1: Schedule onboarding sessions
            - Create a calendar for different onboarding sessions: Intro, Intermediate, Advanced sessions.
        - Ticket 2: Provide answers for common questions and reflect on onboarding process
            - Compile a list of frequently asked questions and assemble an FAQ.
            - Ask customers how the onboarding process can improve
- User story 3: As a first-time user I want to get a notification every time a merge is successful.
    - Task: Send confirmation email when a merge by anyone in the team has been successful.
        - Ticket 1: Generate an email that everyone on the team will receive
            - Ensuring that the email has easy to understand language with a timestamp and a reference to the merge that was just added
        - Ticket 2: Set the email to be sent to all members of a specific team
            - Ensuring there are different shared inboxes for different teams so that each team receives the appropriate confirmation email
- This is not a User story. It is not a user story because it is not a feature that they are asking for, but a requirement that we would need to implement (non-functional).

Formal Requirements
Goal: Test the user experience of syncing merge conflicts with SnickerSync and assess 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 according to organization or department so that only the appropriate people have access to it
  - Users who have access to certain files should be able to see them

Non-functional Requirement 2: Testability

- Functional Requirements
  - SnickerSync must be able to randomize the testing users so that we ensure the data is accurate and unbiased
  - SnickerSync must handle multiple random assignments of users to test their features