Agile:
User Story 1: As a vanilla git power-user that has never seen GiggleGit before, I want to seamlessly transition from git to GiggleGit, with as small a learning curve as possible.

- Task: Provide command-line parity between Git and GiggleGit:
    - Ticket 1: Research important and common commands used in Git
        - We need to create a list of the most important commands used in Git as well as the most commonly used commands, except commands relevant for merging, as that will be done via memes.
    - Ticket 2: Implement the commands
        - We need to implement all the commands from that list such that their syntax and function are as similar as possible

User Story 2: As a team lead onboarding an experienced GiggleGit User, I want to quickly integrate them into my current team's repositories and workflow to allow them to contribute as soon as possible.

- Task: Develop a feature such that team leads can create or upload team-specific onboarding documents.
    - Ticket 1: Create Team Guidelines section in UI
        - Design a section of the UI with and interface with role-based access control (so only teamleads can do this) to create, edit, and upload onboarding documents and exercises
    - Ticket 2: Implement cloud support for onboarding documents
        - Allow onboarding documents to be viewed, edited, and uploaded onto the cloud to allow for collaboration and access from different location

User Story 3: As a longtime GiggleGit user I want to change the memes used to control merges over time to ensure they stay the freshest and dankest possible:

- Task: Create a change meme option to swap out the memes used to control merges,
    - Ticket 1: Create a UI to create new memes
        - Create a secure UI with access control to ensure team leads and longtime users can upload images and add text to those images, to create new memes which can be used for merging
    - Ticket 2: Allow swapping of memes
        - Create a UI for team leads to approve the swapping of memes, ensuring the new memes hold the same functionality as the old memes.

"As a user I want to be able to authenticate on a new machine" is not a user story for multiple reasons:

1. It doesn't specify anything specific about the user, other than that the user is a user.
2. It doesn't provide any purpose or evidence that the user needs this functionality
3. It doesn't show any benefit given to the user, so no reason to think this is important

This is basically just a functional requirement, something the system should do

Formal Requirements:

Goal: Gather quantitative and qualitative feedback from all different types of users about SnickerSync.

Non-Goal: We aren't trying to gather feedback on GiggleGit as a whole or other features in GiggleGit, all that we care about for this particular.

Non-functional requirements:

1. Users need to be assigned randomly to either a control group or a variant of SnickerSync without bias
    a. Functional Requirements:
        i. Implement a random assignment algorithm, which randomly selects out of a database of users which users to give variants and controls
        ii. The group each user is assigned to needs to be recorded such that researchers can ensure that the random assignment algorithm is working as it is intended
2. Product Managers and only product managers need the ability to modify different snickering concepts so they can maintain them
    a. Functional Requirements:
        i. Provide a UI to allow PM's to modify and view changes made to SnickerSync and variants
        ii. Enable a system of version control for snicker concepts, allowing mistakes to be undone