Agile

1. Complete these user stories:
    - As a vanilla git power-user that has never seen GiggleGit before, I want to be able to easily adapt to new versions of the commands I am familiar with.
    - As a team lead onboarding an experienced GiggleGit user, I want to make use of their existing knowledge to enhance their integration into our established production environment.
2. Create a third user story, one task for this user story, and two associated tickets.
    - As a novice developer using GiggleGit for the first time, I want to be able to learn how to use GiggleGit quickly and effectively
        i. Task: Create an easily followed quick-start tutorial/walkthrough.
            1. Ticket 1: Create setup instructions for installing and configuring GiggleGit
                a. We will make a step by step guide that uses simple language to describe the best ways to set up a GiggleGit workflow. This will include detailed explanations and pictures when possible.
            2. Ticket 2: Create simple interactive exercises to ensure the basics are understood.
                a. We will implement an optional interactive walkthrough of the most basic functions of the platform to aid new users. There should be several simple exercises to help them get used to the system.
3. This is not a user story. Why not? What is it?
    - As a user I want to be able to authenticate on a new machine
    - It does not have a benefit, it is just a feature description/request.

Formal Requirements

1. List one goal and one non-goal
    - Goal: Launch built in feedback surveys into the user interface to collect ratings and reactions to the SnickerSync feature and its related user experience.
    - Non-goal: Target specific metrics or types of users with the survey; for example professional developers and experienced users.
2. Create two non-functional requirements. Here are suggestions of things to think about:
    - Who has access to what
    - PMs need to be able to maintain the different snickering concepts
    - A user study needs to have random assignments of users between control groups and variants

- Non-functional requirement 1: Security
    i. Functional Requirements:
        1. Securely store survey data and study results in the ChuckleCode database so that only PMs and their teams can view the feedback
        2. Keep users' personal information safe (if it is collected) by protecting the survey results information
- Non-functional requirement 2: Randomness
    i. Functional Requirements:
        1. Create a randomization method for assigning users to control groups and variants for the different concepts
        2. Keep track of which survey responses are from which study group for later analysis

3. For each non-functional requirement, create two functional requirements (for a grand total of four functional requirements).