Agile

1. Complete these user stories:
    - ○ As a vanilla git power-user that has never seen GiggleGit before, I want to… use this new tool to understand how merges work through memes and get as much information I can from GiggleGit. I want to learn how GiggleGit works and how it can teach me.
    - ○ As a team lead onboarding an experienced GiggleGit user, I want to… give this user an onboarding experience that matches their level of skill/understanding to avoid wasting anyone's time or energy.
2. Create a third user story, one task for this user story, and two associated tickets.
    - ○ Tasks should be a single phrase. (As should themes and epics. See those provided.)
    - ○ User stories should be one to three sentences.
    - ○ Tickets should have a title consisting of a single phrase and details that are long enough to sufficiently describe what needs to be done. You do not need to assign points to the tickets

- User story: As a GiggleGit user, I want to be able to securely log into my account so I can access my repositories and manage my projects.
- Task: Create a login page that is simple/easy to use and secure.
- Ticket 1: Create a design for the login page - Create a simple login page that has a place to input the users username and password. Then also a forgot password button that will take the user to a page where they can input their email and create a new password. The design should be simple but also match the rest of GiggleGits aesthetic.
- Ticket 2: Create user authentication - create a response where if the user has incorrect login it says login failed. Then if they try to login more than 5 times, disable login tied with the username and send the user an email to get back into their account.

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

- This is not a user story because it is not an overarching goal. This is too specific. It is instead a technical requirement. It is a specific want for the system, not a goal.

Formal Requirements

1. List one goal and one non-goal

   Goal: Make sure that it works across every operating system.

   Non-goal: It will not keep track of all of your actions and tell you how to resolve your errors.

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

- Special access: Only users with special authorization have access to all snicker features and can allow different features to different users
- Undo: allow users to undo after they have made a change

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

- Special access:
  - Those with special access should have a page where they can assign different features to different users
  - Those who do not have special access should not be able to see what they do not have permission to use
- Undo:
  - Have an undo button at all times on the users screen
  - Make sure the system is syncing after every change made so that the user can undo their most recent action