Complete the following tasks:

1. Complete these user stories:
    - **As a vanilla git power-user that has never seen GiggleGit before, I want to understand how to use this software and as well as the differences between giggleGit and GitHub, so that I can determine in what fields of use can it assist me in.**
    - **As a team lead onboarding an experienced GiggleGit user, I want to provide user manuals that contain information about advanced commands in GiggleGit and examples on how to use them so that so that it is easy to work on big projects.**
    -
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 user who is a professor, I want to be able to share data and files so that GiggleGit will be convenient for teaching techniques.**
    - **task 1: Design a feature that allows for sharing data between several users.**
    - **ticket 1:**
    - **title: Create a work interface for each user.**
    - **description: Assign an account with a username and password to each user who wants to use GiggleGit, and store each user's data or files they save in their account.**
    - **ticket 2:**
    - **title: Create a visible-to-other-users-interface and a hidden-interface for each account.**
    - **description: For each data file the user creates and wants to save/store in their account, include a required question that gives the choice of either hiding the file from other users or making it public.**
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 a concern related to software engineers (developers and designers of the product) because it doesn't describe a functionality of the product, meaning that this authentication is not something a user encounters when using the software. It is closer to a non-functional requirement (relating to the security of the software). It is probably a requirement.**

Goal for user studies: Check if the product is reliable with correct syncing that plays the correct sound file.
Non-goal: The availability of software use when maintenance is performed by CodeChuckle software engineers.
non functional requirement
1- Reliability
Functional (how to achieve)

1- The sound file shouldn't be in the system when it is not needed. Only import or add the sound file to the system when it's time for syncing, and delete it right after.

2- Add a 2-second pop up icon simultaneously with the sound playing in case the user's device is muted.

2- Interoperability
Functional (how to achieve)

1- Collaboration between developers of SnickerSync and the main interface to allow data access.

2- The ability for the interface to give access to the SnickerSync tool when syncing.