Amelia Liston
CS 411

*All of my responses are in **bold**

Agile

1. Complete these user stories:
    - As a vanilla git power-user that has never seen GiggleGit before, I want to…
        i. **As a vanilla git power-user who has never seen GiggleGit before, I want to be able to use GiggleGit to merge my changes to my project.**
    - As a team lead onboarding an experienced GiggleGit user, I want to…
        i. **As a team lead onboarding an experienced GiggleGit user, I want to easily add them to our repository so they can begin making changes and merging those changes seamlessly.**
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
        i. **User Story: As a new software developer who is unfamiliar with GiggleGit, I want to view the common GiggleGit commands I can use to get my work done in the most efficient way possible.**
            1. **Task: Make common GiggleGit commands available to view.**
                a. **Ticket 1:**
                    i. **Write documentation page**
                b. **Ticket 2:**
                    i. **Design and create a webpage containing the documentation**
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
        i. **This is not a user story because authenticating on a new machine is not a user-related task, it is a developer-related task. A user is not using GiggleGit to do something authentication-related.**

Formal Requirements

CodeChuckle is introducing a new diff tool: SnickerSync—why merge in silence when you can sync with a snicker? The PMs have a solid understanding of what it means to "sync with a snicker" and now they want to run some user studies. Your team has already created a vanilla interface capable of syncing with the base GiggleGit packages.

Complete the following tasks:

1. List one goal and one non-goal
    - **Goal: Ensure SnickerSync provides a smooth and stable user experience to support the upcoming user studies.**
    - **Non-goal: Create new memes for SnickerSync.**
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
        i. ***Non-functional Requirement*: PMs should have access to manage the different snickering concepts and assign variants in SnickerSync for the user studies.**
            1. ***Functional Requirement 1*: Create a dashboard where PMs can log in and manage different snickering templates, assigning them to specific user groups.**
            2. ***Functional Requirement 2*: Prevent general users from making modifications to the snickering configuration by restricting their access.**
        ii. ***Non-functional Requirement*: The system should support random assignments of users into control and variant groups for A/B testing of SnickerSync features during user studies.**
            1. ***Functional Requirement 1*: Randomly assign new users to the control group or the test variant by implementing an algorithm, ensuring equal distribution across groups**
            2. ***Functional Requirement 2*: Track the group assignments in the database, ensuring that user feedback and performance metrics are correctly tagged to their respective group for analysis.**