Project Description:

CodeChuckle is a startup whose product is GiggleGit, a version control system "where merges are managed by memes." (It saddens me to say that this was a joke written by ChatGPT for 131)

You have just been hired as employee number n for some small number n. They have the dev chops to make a demo, but you are their first serious developer.

Here is a theme and an epic:

- **Theme:** Get GiggleGit demo into a stable enough alpha to start onboarding some adventurous clients
- **Epic:** Onboarding experience

Starting of User Stories:

- **User Story 1:** As a vanilla git power-user that has never seen GiggleGit before, I want to import an existing repository from other version control system, so that I can import my previous work or other people's work without consuming time on reuploading or manually adjust to adapt GiggleGit.
- **User Story 2:** As a team lead onboarding an experienced GiggleGit user, I want to have a concise tutorial to efficiently train new member, so that new member could quickly get used to the unique features in GiggleGit.
- **User Story 3:** As a manager of group project that working on multiple branches, I want to see the frequency of merge conflicts across different branches, so I can easily identify the area that needs improvement
  - ➢ **Task**: Develop a feature for tracking and reporting the conflict frequency
    - ◆ **Ticket 1:** Creating a conflict recoding features
      - Build a feature that record the occurrence of merge conflicts, which will show the branches involve, name, conflict type, time, and frequency. Make sure that the information can be easily retrieved through user query.
    - ◆ **Ticket 2:** Create a visual representation for conflict frequency
      - Develop a visual representation, which indicate the frequency of merge conflicts across different branches over time, that could be easily interpret by user. This could be a matrix, chart, heatmap, or another visual format. The representation could also be interactive that allow search or export?

This is not a user story. Why not? What is it?

- ○ **<u>"As a user I want to be able to authenticate on a new machine"</u>**

The statement "As a user I want to be able to authenticate on a new machine" is not a user story because it does not specify the actor (it is too general by saying the person is a user but not specifying the group of people) but only indicate the actions that the person wants to take. Additionally, the statement does not show the benefits or value that the user will get from performing the action. Without pointing out the benefits, we cannot align the work with the business value and the users' needs.

Formal Requirement:

Goal: Develop a user-friendly interface for SnickerSync that integrate with base GiggleGit packages, adding interesting, snickering audio during the merging process (trigger this audio when merging)

Non-Goal: Develop a snicker generator that dynamically trigger or adjust the snickering audio based on the merging result. (e.g. lower the volume of audio when encountering merging conflict)

Non-Functional Requirement 1: Access Control

- **Description**: Ensuring that the different users in the project have corresponding ability to access or control the tool's settings and features.
- **Functional Requirements**:
    - Access control based on role of the user or developer in the project. Implementing that certain features or settings could only be modify and used by project manager.
    - Restrict the access permission of developers that does not privileged to write the snickering concepts database but only read the database. So that only certain people with permission could add or delete items in the database

Non-Functional Requirement 2: Support random assignment for user studies

- **Description**: The system allow manager to randomly assign users into variants or controlled group for unbiased user studies, which strengthen the internal validity.
- **Functional Requirements**:
    - The automation of randomly assign users to either controlled groups or variants based on different purposes and pre-defined information of the users in the backend system.
    - Ensure that SnickerSync will record the actions from user interaction with the system based on different groups that user is being assigned, the data should be stored in a well-designed structure for further analysis.