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: Seamlessly sync SnickerSync with GiggleGit.
Non-goal: Implement advanced features that will make the platform annoying to use.

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**

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

1) Ensure that the system is secure from unauthorized users and syncing properly aligns with the snicker.
   - The system must be secure by using captcha and two-factor authentication to prevent any hackers, bots, and any unwanted users from accessing personal projects.
   - Prevent sound delays with snicker by implementing triggers when syncing in progress. This is to enhance experience while using snickersync.
2) Be able to adapt to different formats of projects/be compatible with other platforms
   - The system must be able to support various platforms such as: source code files that use: C++, python, java, etc. and it should be able to run across different operating systems: windows, macOS, etc.
   - Containerization of programs where it can encapsulate applications, runtime, libraries, etc, into one unit so it is portable and compatible.