CAS CS 411, Fall 2024 Sydney Stokes
Homework #2 October 7, 2024

Agile

1.
   - User story 1: As a vanilla git power-user that has never seen GiggleGit before, I want to be able to quickly understand how merging with memes works in GiggleGit so that I can clearly see how it differs from other version control systems and determine if it would be beneficial for my team to use.
   - User story 2: As a team lead onboarding an experienced GiggleGit user, I want to have the ability to view the merge activity of my team members to make sure that each one understands the system and is using it effectively.

2. User story 3: As a student and aspiring developer who is trying to learn and familiarize myself with different version control systems, I want to have access to beginner-friendly tutorials in GiggleGit so that I can learn how to manage merges using memes in an easy-to-understand way.

   - Task: Create a series of beginner-friendly tutorials for new users on how to use GiggleGit.
     - Ticket 1: Design a variety of beginner-friendly merge tutorials
       - Details: Create introductory tutorials that walk new users through the basic functionality of merging with memes in GiggleGit. Ensure the tutorials use simple language, provide step-by-step guidance, and includes screenshots/screen recordings to make it easy for beginners to follow and understand.
     - Ticket 2: Implement beginner tutorials in GiggleGit UI
       - Details: Integrate the designeds tutorial into the GiggleGit interface. Ensure that navigation to these tutorials is straightforward so that users can easily access the tutorials from the main dashboard, and that the tutorials naturally progress through all the necessary steps.

3. "As a user I want to be able to authenticate on a new machine" is not a user story because it doesn't include the benefit that will be gained from the desired feature. Additionally, it doesn't give the specific type of user that is looking for this particular feature. Thus, it is simply a non-functional requirement.

Formal Requirements

1.
   - Goal: Create a unique, engaging, and efficient user experience for syncing files that enhances user satisfaction and integrates seamlessly with the existing GiggleGit interface.

- Non-Goal: ...

2/3.
- Non-functional requirement 1: Performance / Scalability
  - Functional requirements:
    * Should handle the syncing of large files (up to 10GB) without performance delays, errors, or failure.
    * Must support multiple users simultaneously synching different repositories without performance delays.
- Non-functional requirement 2: Reliability
  - Functional requirements:
    * System should automatically retry any failed sync operations up to three times before notifying the user of the issue.
    * System should thoroughly document and log all sync activities, including any errors and failures, with timestamps for future troubleshooting and analysis.