CS411 HW- Beren Yagmur Donmez

Agile

1. **Complete these user stories:**

* As a vanilla Git power-user that has never seen GiggleGit before, *I want to easily understand how to manage merges using its meme-driven interface so that I can teach it to my students and make Git learning more fun.*
* As a team lead at onboarding an experienced GiggleGit user, *I want my team to use GiggleGit's meme-driven merge system effectively for our financial software projects, ensuring efficient workflows and leveraging the platform's unique features.*

**2. Create a third user story, one task for this user story, and two associated tickets.**

* As a designer creating memes for GiggleGit users, I want to create different meme designs that can be used in various committing, coding and collaborating scenarios, so that users can express their feelings,experiences and accomplishments during the merge process in a fun and expressive way.

#### ***Task for Me (the Developer):***

* Implement a user-friendly interface for uploading and managing memes.

##### ***Ticket 1:***

* **Create meme upload feature for designers**
  + Develop a feature that allows designers to upload their meme designs to GiggleGit easily. This feature should include options to categorize memes according to their themes, such as “ resolution,” “celebration,” “conflict,” “frustration,” or “roadblock.” Make sure that the upload process is intuitive and supports common image formats.

##### ***Ticket 2:***

* **Design meme approval system for quality control**
  + Implement a feature for review and approval of submitted memes before they are available to users. This should include a notification mechanism for designers once their memes have been reviewed, as well as a feedback option for improvements if necessary.

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

A good quality user story is independent, negotiable, valuable, estimable and testableThe original user story is a basic statement that lacks depth and clarity. In contrast, the correct user story is detailed and comprehensive, specifying the user’s role, a clear goal, and the reasons behind the request. This additional context helps development teams understand the priorities and impacts of the requested feature, facilitating better planning and execution. Overall, the correct user story exemplifies the importance of specificity and clarity in effectively communicating user needs. The user story "As a user, I want to be able to authenticate on a new machine" is not an effective example because it lacks the necessary specificity and context. First, the phrase "a user" does not provide enough detail and insight about who the user is. A more precise description, such as "a team lead onboarding an experienced GiggleGit user," like the one provided in the assignment gives a better understanding of the user’s responsibilities and relevance. This clarification lets the developer to adjust the feature so that it meet the needs of a particular user type. Moreover, this might describe a clear task but it doesn't explain the broader purpose or the user’s larger objective. A better goal explanation should focus on the bigger picture and show how the task contributes to a critical function. If we shift the focus to the user’s overall objective and the value it brings, the goal becomes more relevant and better connected to the user's needs. In addition, restating the user story which was “As a user I want to be able to authenticate on a new machine” does not to tell the “why”of the feature which makes it difficult us developers to see the value of it. If I state my user story again “As a designer creating memes for GiggleGit users, I want to develop a diverse collection of memes that reflect various coding scenarios, so that users can express their feelings and experiences during the merge process in a fun and relatable way.” we can see that I provide a “so that” part for a clearer understanding of the reason behind the request and show how it supports business objectives and provide value to users.

Formal Requirements

1. List one goal and one non-goal

Goal: Allow users to sync their files while receiving meaningful snicker sounds that reflect the nature of the merge, such as a playful snicker for successful merges and a confused snicker for conflicts.

Non-goal: Implement advanced sound customization options where users can upload or create their own snicker sounds.

1. Create two non-functional requirements. Here are suggestions of things to think about:

**Non-Functional Requirement 1:** Only designated PMs should have the ability to manage and update the different snickering concepts within SnickerSync.

**Functional requirements for the 1st Non-Functional Requirement**

-The system must perform permission checks before allowing any modification actions (create, update, delete) on snickering concepts to ensure that only PMs can execute these actions.

-The system must redirect non-PM users attempting to access the management features to a designated access denied page, clearly stating their permission issues.

**Non-Functional Requirement 2:** The system must efficiently and automatically allocate users to control and experimental groups for user studies to ensure unbiased results.

**Functional requirements for the 2nd Non-Functional Requirement**

-​​The system must notify users of their group assignments when they register to ensure that they know their participation in the study.

-The system must track the number of users in each group and adjust assignments in the future to maintain a balanced sample size throughout the user study.