Author: Judah

Date: Mar 4, 2023

Topic: Kanban Board/Issues Tracker

The Kanban board will act as the major source of documentation, for issues related to both our engineering process and the Quiz software that we build. All items pertaining to the project, software-wise or engineering process-wise, will be represented as an issue on the board. The engineering process includes, but is not limited to, individual status updates, delays encountered in the lifecycle of the project, requests for help, to list a few.

Prabin created the first draft for our Kanban board, and upon some careful thought and review, I decided on the following design/structure for our Kanban/storyboard. This structure was presented to and accepted by the entire team.

The three main components of the board are **views**, **columns** and **labels**.

Views are used to demarcate the different phases of the lifecycle of the project. For our particular team setting, because we are adopting the SCRUM methodology, we have 4 main views; **Product Backlogs**, **Todo: Assignment 2**, **Todo: Assignment 3**, **Todo: Assignment 4**.

As the name signifies, the **Product Backlog** will encapsulate all desired user stories and features, tasks and issues related to the engineering process. The views titled/named after each **Sprint** will contain associated issues.

Columns signify the **statuses** of issues within each **view** (or phase) of the project cycle, or the workflow. Making use of filters under a specific view, we are able to hide irrelevant **columns** (or statuses) from a particular view.

Below is the structure of the **views** and **columns** of our project:

- **Product Backlogs**
    - **New/Design**:
        - all new issues (user stories, features, or other) will originate here.
        - all issues here must not be assigned to a Sprint/Iteration yet, not until they have been well documented/designed under the comments section of the issue.
    - **Ready**:
        - once the issue has been well documented (with detailed description and design), it will be moved here.
        - any issues under this status must be assigned to the appropriate Sprint cycle.
        - if the issue is software-related the team lead must assign it to a team member, following discussion with that member
        - if the issue is process-related, the creator of the issue should self-assign the issue to themselves or the respective team member

- **Todo: Assignment 2 | Todo: Assignment 3 | Todo: Assignment 4**
    - **Ready**:
        - due to the filters, only issues assigned to a particular Sprint will appear under the **Ready** status of the relevant Sprint view.
        - the issue assignee is responsible for moving the issue to the appropriate status column
        - the issue assignee is also responsible for including the relevant description/documentation, under the issue, as they change the status of the issue (move it across the columns)
    - **In Progress**:
        - all issues here must be **actively** being worked on, otherwise they must not be under this status column
    - **In Review**:
        - issues here have been completed by the assignee
        - if it is a software issue, it is awaiting review and/or approval by the respective team lead
            - the issue must be updated to include the link to the committed code/pull request
        - if it is an engineering process issue, it is awaiting review/approval by the self-assignee or the team member whom the issue concerns or relates to
    - **Rework**:
        - issues (features) that have been reviewed by team lead, and for which changes have been suggested must be placed under this status, and **not** the **In Progress** status. This helps to truly delineate tasks.
    - **Done**:
        - software-related issues/tasks that have been reviewed and approved for merging must be moved to this status, either by the approver (team lead) or the issue assignee
            - when the related code is merged, the issue can be closed
        - process-related issues that have been resolved must be placed in this status and closed by the relevant party

Labels are used for granular description of issues (software or process-related). They describe permanent attributes about an issue, so **ideally** they should not be changed after initial tagging (unless they were initially used incorrectly).

Below are the labels to be used on each created issue:

- ***Hierarchy***	-	user story	feature		
- ***Scope***	-	team	individual		
- ***Kind***	-	documentation	code	process	test
- ***Type***	-	core	requested		
- ***Testing***	-	manual	unit		
- ***Frequency***	-	cycles			

NOTE that the ***requested*** label is reserved **exclusively for** the user stories and features that the Professor stipulates that each team member should create for Assignment 2. As such, they should be treated as separate and are not core functionality that the app would have by **default**

(core functionality). They are additional user stories/features that the team will consider for implementation in future Sprints using the MOSCOW standard (**M**ust, **S**hould, **C**ould, **W**on't).

NOTE that the ***cycles*** label is reserved just for those issues that will be common across all Sprints. In our setting, they are the **Architecture and Code design**, **Conduct Code Review**, **Conduct Performance Review** issues.

All non-testing, software-related issues must have at least one label from the first four categories.

<u>*Example 1*</u>:

Judah creates an issue for the User API's. He uses the ***user story***[1] label to indicate the issue's top-level hierarchy. The user story will be assigned to an individual member so he tags it as ***individual***[2] and labels it as ***code***[3] since it is software-related. Finally, he tags it as a ***core***[4] functionality, because it is such.

<u>*Example 2*</u>:

Judah creates an issue related to the User API's user story he created earlier. This issue will be a ***feature***[1], indicating it's a bottom-level hierarchy, and it is tied to a user story. He labels it as the ***individual***[2] and ***code***[3] since it is software-related. Finally, he tags it as a ***core***[4] functionality, because it is such.

<u>*Example 3*</u>:

Judah is running behind on the expected delivery for a particular feature. This is because he has a major exam (for another course) that he is studying for and he has not fully familiarized himself with the library he needs to complete the feature. So he creates a new issue detailing why he will be late on completing his assigned task, and labels it as a ***userstory***[1], **individual**[2] and a ***process***[3] related issue.

<u>*Example 4*</u>:

Judah creates a new ***userstory***[1] issue for the **Establish Code Review Standards** task. He appropriately labels it as ***documentation***[2], and since it is one of the required components of the project, as per the Professor's guidelines, he tags it as ***core***[3]. Finally, because this is a group task, he labels it as a ***team***[4] issue.

<u>*Example 4*</u>:

Judah wants to test a new feature recently pushed to the main branch. He creates a new ***userstory***[1] issue. He labels the issue appropriately as ***test***[2] and ***unit***[3] to indicate what kind of testing he will be doing.

Take note that all created issues are either a ***userstory*** or a ***feature***. From there, every other attribute stems.