Quick How-To on Kanban Boards for Agile Projects

The Kanban board is used to track tasks in the form of features described by user stories. They are supported by Github Projects as well as several other tools. They can also be done offline using a literal board.

In a nutshell, the board is divided into columns. Tasks are captured as program features and are written on cards. Cards start in the left hand column and move right as you work on them. Completed tasks are in the right most column.

Features

Program features are developed as user stories with an eye towards expected system behavior. They initially begin as as rough ideas but are *worked* into clear **user stories**.

The story begins by specifying the **user roll** in question, their **desired functionality**, and the **benefit** of that functionality. Included in this are whatever details and assumptions we know, or think we know, about the feature.

```
As a [role]
I need [functionality]
So that [benefit]

Details and Assumptions:
  * ....
```

Users can be actual program users, program developers, program maintainers, investors, and generally any *stakeholder* in the program. This means functionality and its benefit can include user-facing features, code-enhancements, infrastructure enhancements, etc. The wider you cast your net, the better you'll understand your project.

Once a high-level feature is specified, you can include more fine-grained scenarios in the story. These we specify using *Gherkin* syntax:

```
Given [some context]
When [certain action is taken]
Then [the outcome of action is observed]
(And Given...When...Then...)
```

Once again, the focus is on external system behaviors where contexts can be user actions, developer actions, or any fine-grained situation that your pre-specified stake-holder might find themselves in when interacting with this feature.

Feature Size Estimates

Once a feature has been worked up into a user story and you're ready to deliver that feature, then you need to *estimate the overall size of that feature*. Notice we're not saying "time to complete". Time is fickle, what you want is to refine your ability to size a feature with the understanding that larger features will likely take more time. We'll express *size as an integer from 1 to 8* with 1 being small and 8 being large.

The Boards

The Kanban board is a series of columns into which we place our features/user-stories. This might actually mean post-its organized into columns on a board, but often means a digital platform in which we place “cards”. Stories move from left to right where the right most column are completed stories and the left most column are unworked ideas. Here’s a typical column sequence (left to right order).

1. **New Items** New feature ideas/needs. Not yet worked into proper stories. Just ideas, bugs, needs that you’ve run into while addressing
2. **Icebox** Features that are low priority and which you have no plans to work on soon.
3. **Product Backlog** Worked features that you need/want to complete.
4. **Sprint Backlog** Features designed for work right now, but not necessarily being worked on. *Should have a limit on the number of cards in the column.*
5. **In Progress** Features you are currently working on. *Should have a limit on the number of cards in the column.*
6. **Review** Features that are done but need further review or testing. *Might have a limit on the number of cards in the column.*
7. **Done** Completed Features.

Working the Board

We can use the board to manage and prioritize work as follows:

1. (**Feature Refinement**) Work features in *New Items* into proper user stories and place into *Icebox* or *Product Backlog*. Any story in the backlog should include a size/time estimate. Revisit and refine features in the product backlog.
2. (**Sprint Planning**) Prioritize features in the product backlog and decide on a appropriate set of features for your **sprint**. Moves these to the *sprint backlog* and put them in *order them by priority order*. Sprints should be limited to a fixed number of tasks.
3. (**Sprint!**) *Until the two-week sprint is complete:* Choose a feature from the sprint backlog and move it to in progress. When completed, move it to review. When all reviews are done, move it to *done*.
4. (**Sprint Reflection and Retrospective**) When the sprint is over:

- If any features are incomplete, break the feature into two features, the part complete (with new estimate!) and the part incomplete. Place the incomplete part in the product backlog and the complete part in done.
- Compute your *velocity* - total size completed divided by initial size of the sprint. **Use this when planning your next sprint**
- Reflect on your sprint. How were your estimates? Were they way off? Do you need to adjust estimates in the backlog? What worked well for you? What should you do differently next sprint?

5. Go back to step 1.

The Sprint

It should be clear by now that a sprint is a **two-week** period in which you will deliver a pre-determined set of features. The goal is to choose a set of features that are important and whose total size you can manage in a *single sprint*. Sprints should have a cap on the number of features, regardless of size. As you learn about your pace and improve feature size estimates, you can adjust, but in general the goal is find that goldilocks amount of work that fits within your fixed-time increment.

Planning and Prioritizing

The trick to sprint planning is to have clear guidance on your overall goal and priorities. At a macro-level that's have a clear vision for what you hope to achieve. For your capstone that can also be what you hope to learn or what skills your hope to gain through the project. In industry this comes from the goals of the organization and the high-level understanding of what your system should be and should do. At a more fine-grained level, priorities can shift between bug-fixes and the addition of new features. Sometimes you need to shift of change your priorities based on something you learned or encountered on a previous sprint. The important part here is that you have a clear picture of what you hope to achieve and gain over the long run and that when you select you prioritize features and plan your sprint, you can justify those choices based on those larger, longer-term goals.

Minimum Viable Products

One final concept that can influence your sprint planning is working towards a **minimum viable product**. What's meant by this is a functioning system that is meant to test some hypothesis about what will or won't work for your intended user. For example, this can be a program with what you think should be the final interface but without much backend functionality. Such a program can be used to verify your interface plans. The key here is that the program isn't approached as simply a subset of the intended features for the overall system but as a means to validate some ideas, features, or other system plans through real-user interaction.

Due to the time constraints of the capstone, students often end up with what might be considered a minimum viable product related to their original idea. The program does some of what they planned on and does it well, but still lacks some of the features they initially thought might be a part of the system.