Agile Software Development

Agile, User Stories, and Agile Project Management

Workbook 3

Version 6.0Y

Table of Contents

Module 1

Agile Software Development

Section 1–1

Agile Software Development

Agile

- **Building software is complex and not just because writing the code is difficult!**
  - Keeping track of changing priorities, who is working on what, and what tasks need to be completed before others can be started is difficult
- **Software developers and product managers have come up with a number of excellent tools to make it easier to plan projects to keep everyone on the same page**
- **Agile is a structured and iterative approach to project management and product development**
- **With an Agile approach, we focus on being open to change during product development and respond to that change without going off the rails**

Waterfall: The Bad Way!

- **The term Waterfall was coined as an example of what not to do.**
  - Yet, a lot of companies do just this
- **In Waterfall, project management looks like this:**
  - Everything is planned upfront
  - All features are developed from the start
  - All features are tested at the end
  - The product is shipped to the customer
- **But what happens when:**
  - You have a poorly articulated vision of the end product
  - Project requirements changes during development?
- **It's often a *long* time before stake holders see anything that works**

Agile Philosophy: The Manifesto

- **The *Agile Manifesto* is a document that defines the core values behind the Agile philosophy**
  - Its goal is to help development teams work more efficiently and sustainably
- **As a class, let's go look at the Agile Manifesto**
  - `https://agilemanifesto.org/`
- **Discussion: What do these mean values to you?**

Agile Philosophy: The 12 Principles

- **Agile's "founders" came up with the Agile Manifesto be defining 12 principles that would help organizations be more flexible, responsive, and adaptive to changes**

- **As a class, let's go look at the 12 Agile Principles**

  – `https://agilemanifesto.org/principles.html`

- **Discussion: What do you think these help software development teams?**

The Scrum Framework

- **Scrum is a framework that helps teams work together**

- **It encourages teams to:**
  - self-organize while working on a problem
  - learn through their experiences
  - reflect on their successes and failures in order to continuously improve

- **Scrum team members belong to one of three roles:**
  - product owner
  - scrum master
  - development team member

- **Scrum is just one of the many iterative and incremental Agile software development frameworks which are used widely in software development**
  - However, it is very popular and will be examined here

Sprints

- **Scrum relies on delivering working software in short cycles called Sprints**
  - The use of sprints is at the very heart of Scrum and other Agile methodologies
- **A sprint is a short period during which a scrum team works to complete a specific amount of work**
  - Often, sprints are 2-3 weeks long
- **Sprints enable a team to:**
  - Get fast feedback
  - Continuously improvement
  - Rapidly adaptation to change
  - Accelerate delivery
- **Sprint planning is important! You must decide on:**
  - what is the sprint goal
  - how long the sprint is will last
  - where you team will start

The Daily Standup

- **When we talk about Agile, we are referring to a methodology to plan and complete projects**
  - In order to do this, there are a number of rituals teams participate in to help the Agile process keep moving
  - They include the ***daily standup*** and the ***retrospective***
- **During the sprint, team members meet daily at the *daily scrum* or *standup***
  - It is an important part of the day!
- **It is here team members to talk about:**
  - their successes
  - the issues they face
  - their blockers
- **They may reach out and ask for help**
  - Or help might be offered
- **The scrum master usually facilitates the daily scrum**
  - However, it is the development team's responsibility to run the standup

- **Questions are often:**
  - What did you accomplish yesterday?
  - What do you plan to accomplish today?
  - Where are your blockers?
- **It is very important that team members be comfortable surfacing the problems they are encountering in order for the process to work**

Retrospectives

- **It's important to reflect back on what you've done in order to better prepare for the future**
  - ***Retrospectives*** are one way to do this reflection
- **A retrospective usually happens at the end of a sprint or large project**
- **Some of the common forms they take include:**
  - Start/Stop/Continue
  - Happy/Sad/Confused
  - Positives/Deltas
- **No matter how you do your retrospective, the goal is to hear from multiple people in the group and derive action items for your next sprint or project**

Research: Learn More

- **Take a little time to get a better sense of what the Scrum implementation of an Agile process looks like**
  - Scrum Alliance: Why Scrum?
    - `https://www.scrumalliance.org/about-scrum`
  - Scrum Roles
    - `https://www.agile42.com/en/agile-community/agile-info-center/scrum/scrum-roles`
  - Scrum Glossary
    - `https://www.scrum.org/Resources/Scrum-Glossary`

Module 2

User Stories

Section 2–1

User Stories

User Stories

- **If we have an idea that's going to change the world, how do we get started?**
- **If we want millions of users to love our new social media platform, how do we get there?**
- **Tackling large problems is impossible without a plan**
  - We need to break apart large project into smaller, manageable tasks
  - We do this by creating User Stories
- **The goal of User Stories is to focus on features with the user in mind**
  - Being able to write concise user stories will make projects easier to accomplish and make collaboration with multiple developers easier
  - HOWEVER, sometimes there are developer user stories too!
- **User Stories come from requirements**
  - Each requirement of the application should be converted into one or more user stories
  - By focusing on the user interaction, it is easy to verify if the requirement has been met

What is a User Story?

- **What is a user story?**

  Format

  - (role) can (action) -

  ex: As a (role) I want (something) so that (benefit)

  Example

  As a ***web site user*** I want a ***navigation bar*** so that ***I can switch between pages***

- **Recommended Reading and Examples:**
  - Mountain Goat Software - User Stories
    * https://www.mountaingoatsoftware.com/agile/user-stories
  - GSA
    * https://tech.gsa.gov/guides/user_story_example/

Prioritizing User Stories

- **Before we start to work on a project during a sprint, we want to prioritize our stories**

- **To help prioritize our stories, we may think about the following questions:**
  - What is time-sensitive and needs to happen now?
  - What will add the most value to the project and get it closer to a shippable product?
  - What work can happen in parallel?
  - What work depends upon another story being finished?

- **There is no "right" way to prioritize your tasks but the goal should be to get a *minimum viable product* (MVP) as soon as possible**
  - The MVP is a version of a product with just enough features to be usable by early customers
  - These customers then provide feedback that is used for future enhancements

Exercises

In this exercise you will convert requirements into User Stories. Remember some requirements may be broken down into more than one User Story (i.e. the requirement may affect different types of user experiences differently: `admin` vs `customer`).

Your stories should be specific to the type of user who is using the application.

You will notice that the requirements can be somewhat vague, but your user stories should not be vague. They should be specific to the needs of each user.

Application: Hardware Store

EXAMPLE

Requirement:

`Display products list`

User Story:

`As a customer,` `I want to` **`view a list of products,`** `so that I can` **`determine what I want to buy.`**

EXERCISE 1

Requirements:

1. A customer should be able to search for a product
2. A customer should be able to filter the products list
3. A customer must be logged in to make a purchase
4. A store owner should be able to add new products
5. A store owner should be able to see a list of orders
6. We should be able to remove a product
7. We need to update product inventory
8. We need to print a Order Pick List for each order in the store warehouse

Module 3

Agile Project Management Tools

Section 3–1

Agile Project Management Tools

Tools

- **Once you have a number of stories, you need some way to track:**
  - who is working on what
  - what has yet to be completed
  - what has already been done
- **There are a number of tools that will help you manage these user stories by providing kanban boards, including:**
  - GitHub Projects
  - Trello
  - Asana
- **A kanban board is an agile project management tool that can help you visualize work**
  - It will help you manage your work by specifying work-in-progress, work completed, and backlogs (yet to start)
  - Tools can also specify other states that a work can be in
- **We will use GitHub project boards in this class**

Issue Tracking

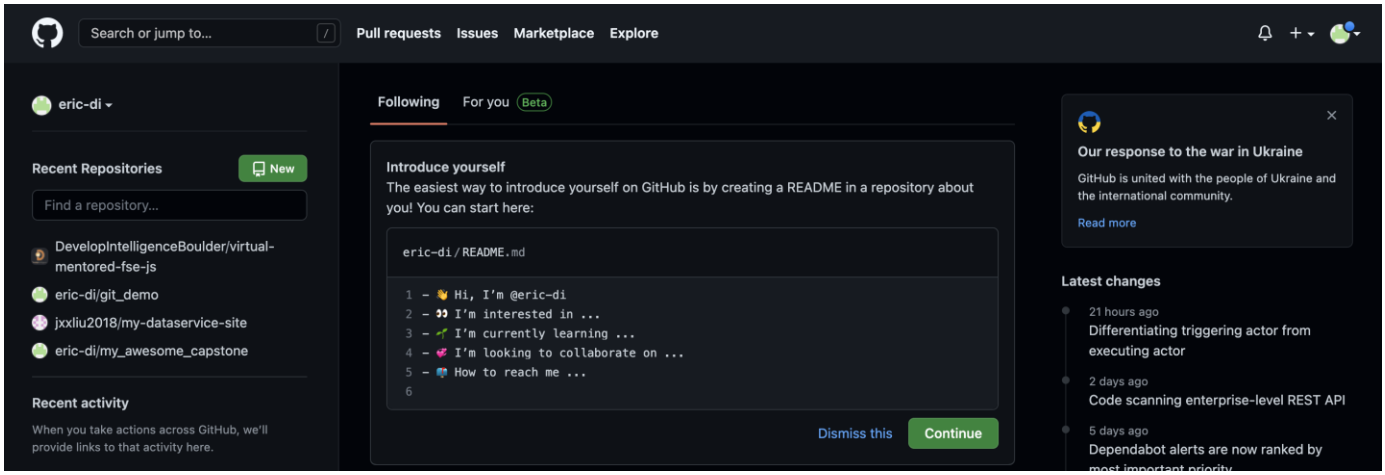
- **These tools all help us with something called *issue tracking***
  - Issue tracking is really just a specialized to-do list.
- **This to-do list has an extra layer supporting a review process for the task**
- **Instead of a task just being completed, you may have several states indicating where the task is in the workflow**
- **Usually, the last step in the workflow indicates that the finished work has been approved**
  - This helps prevent problems from slipping through the cracks by ensuring that there are at least two sets of eyes on every resolution

Setting up a GitHub Project Board

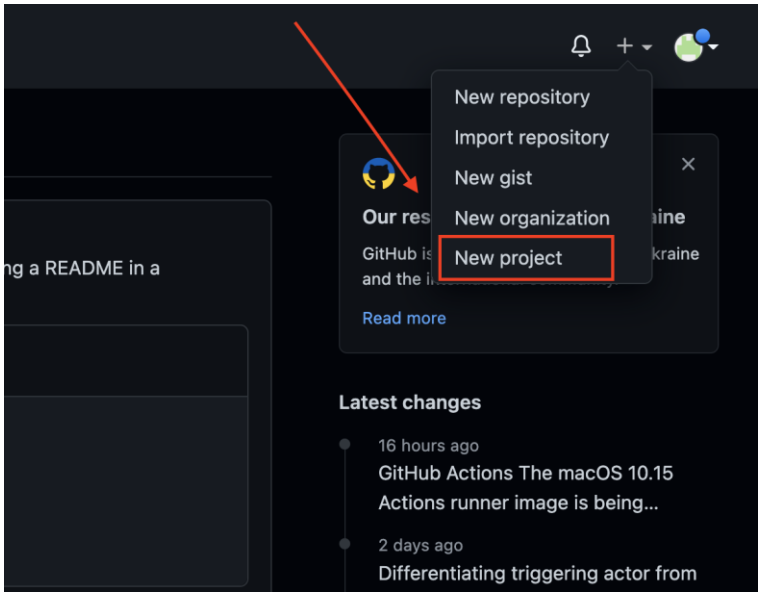
- **Now that we have an idea of what the Agile approach to Software Development projects encompasses, lets:**
  - set up our project board
  - start capturing and prioritizing the user stories
- **We will use a GitHub Project board that can be linked to the repository with our code**

Creating a Project Board on GitHub

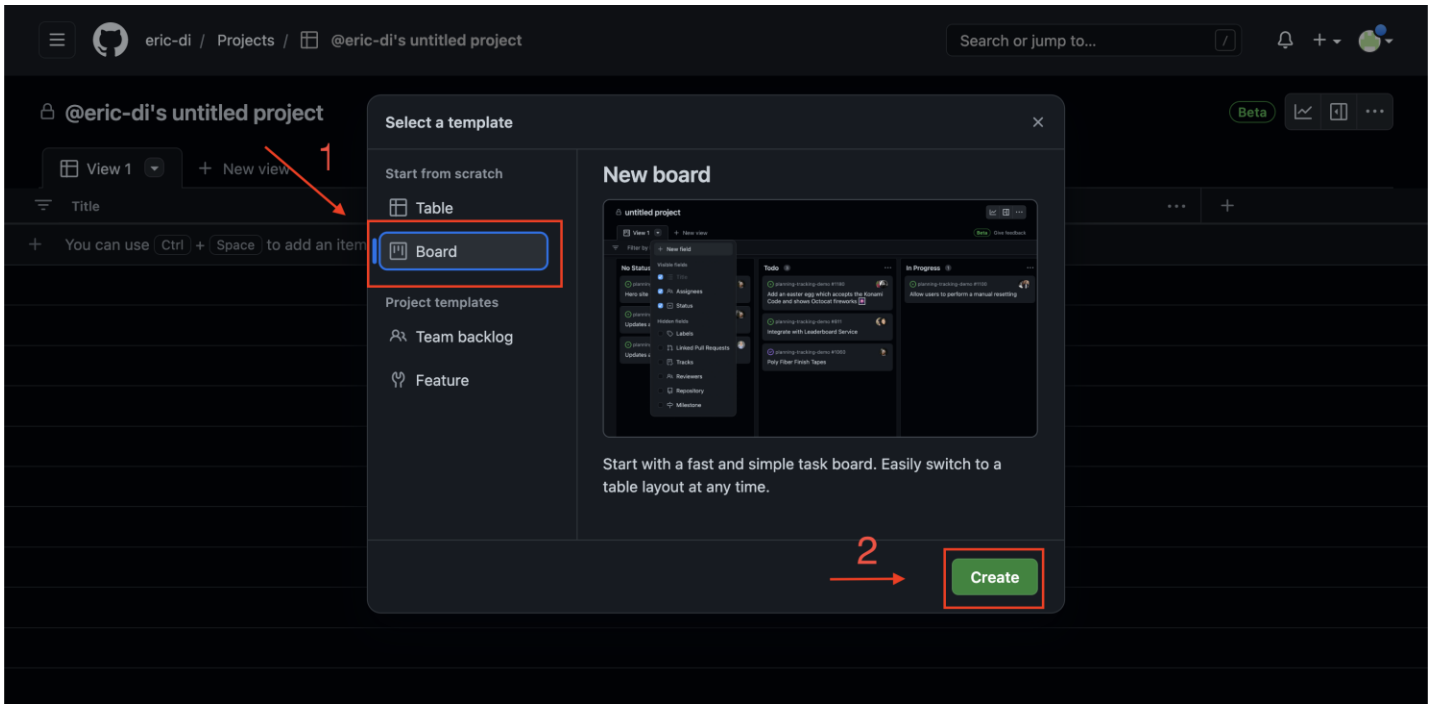
- **Start by going to `https://github.com/` to create your project board**
  - Make sure you are signed in
  - If signed in, you should see a page similar to the below image



- **Click the `New Project` option under the + menu on the top right of the screen next to your avatar**

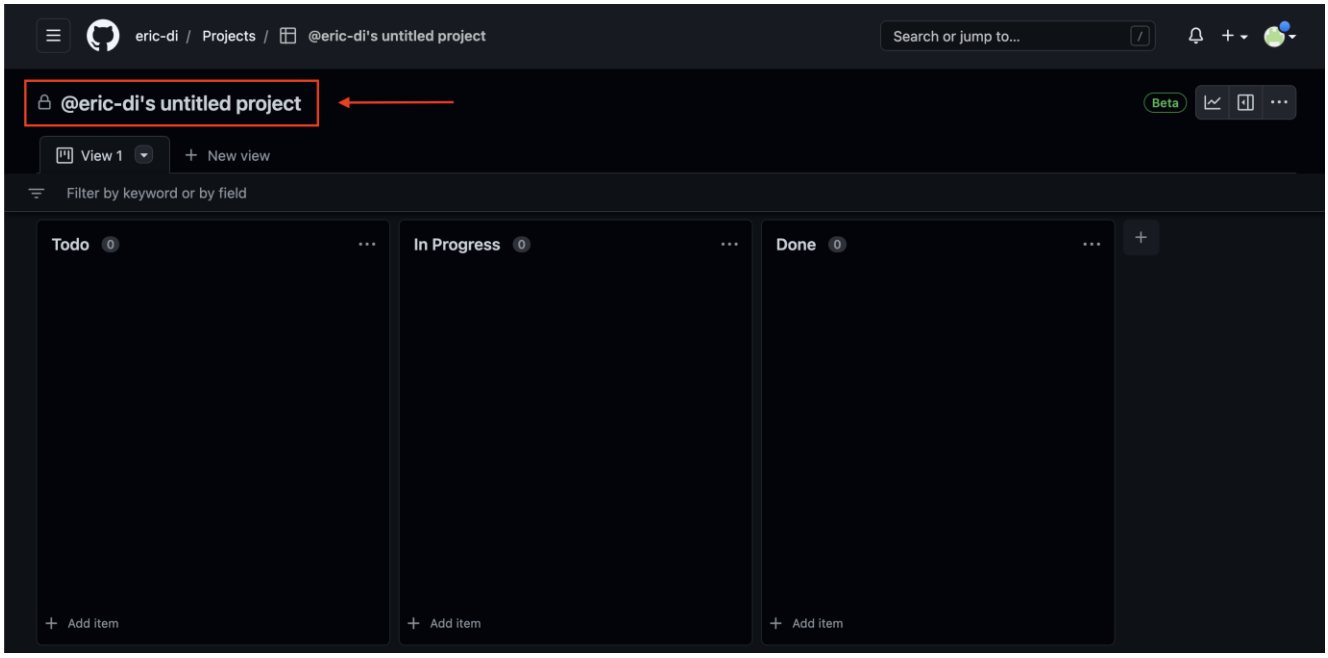


- **This will bring up the following screen to create the project**
  - We are going to select `Board` on the left menu and then click `Create` on the bottom left of the dialog

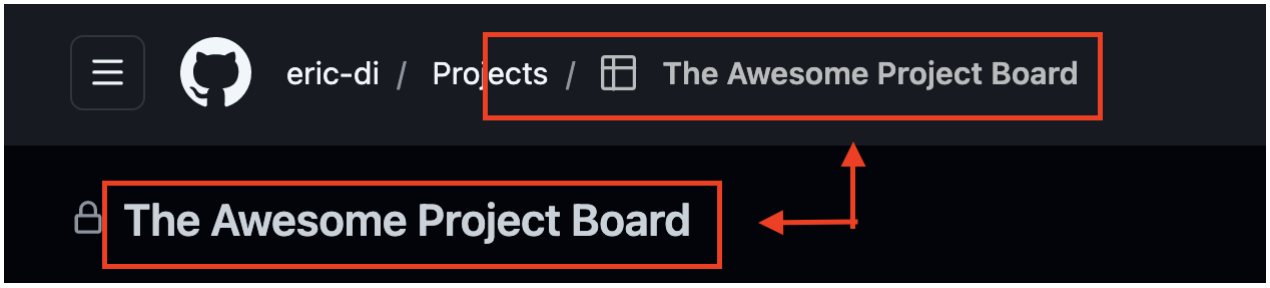


Renaming the Board

- Now that we have the project board we can rename it by clicking on the name in the top left corner of the board

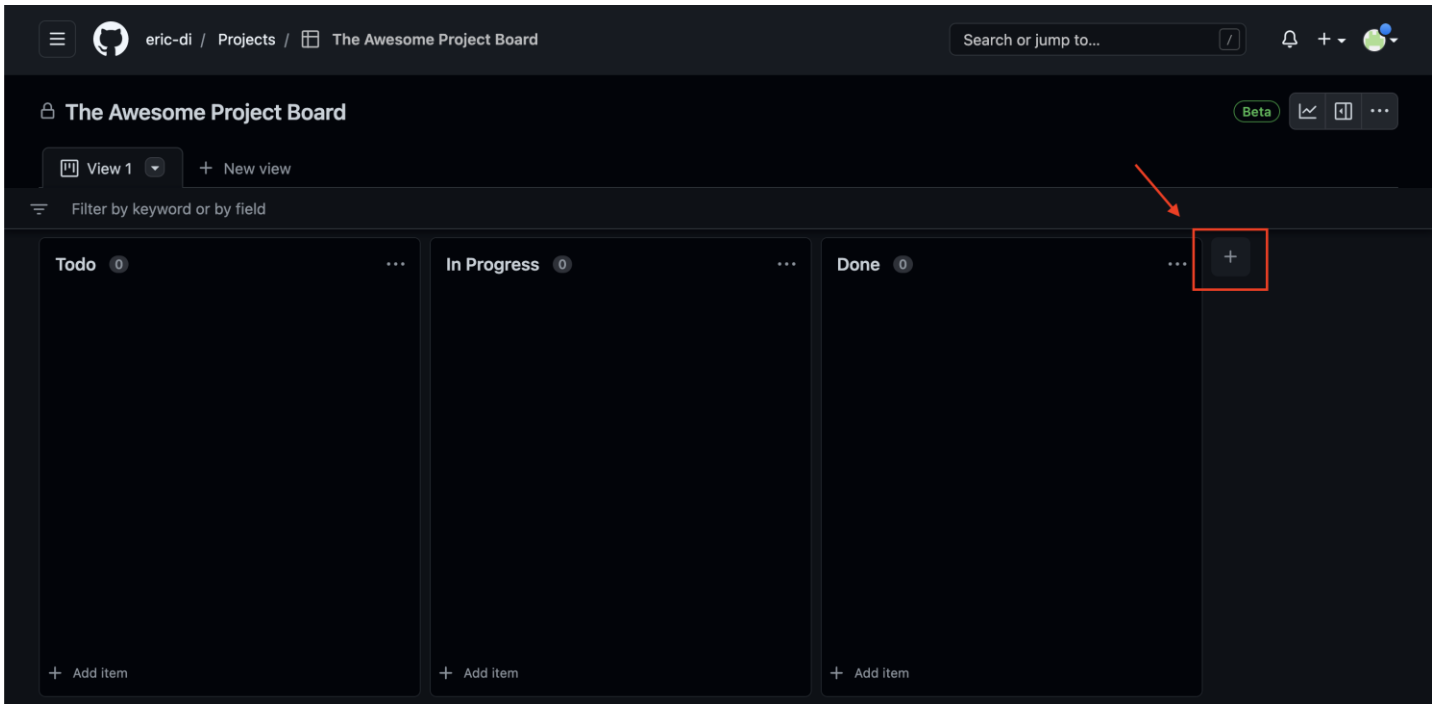


- If updated correctly you should see the new name reflected in the following places on the project board



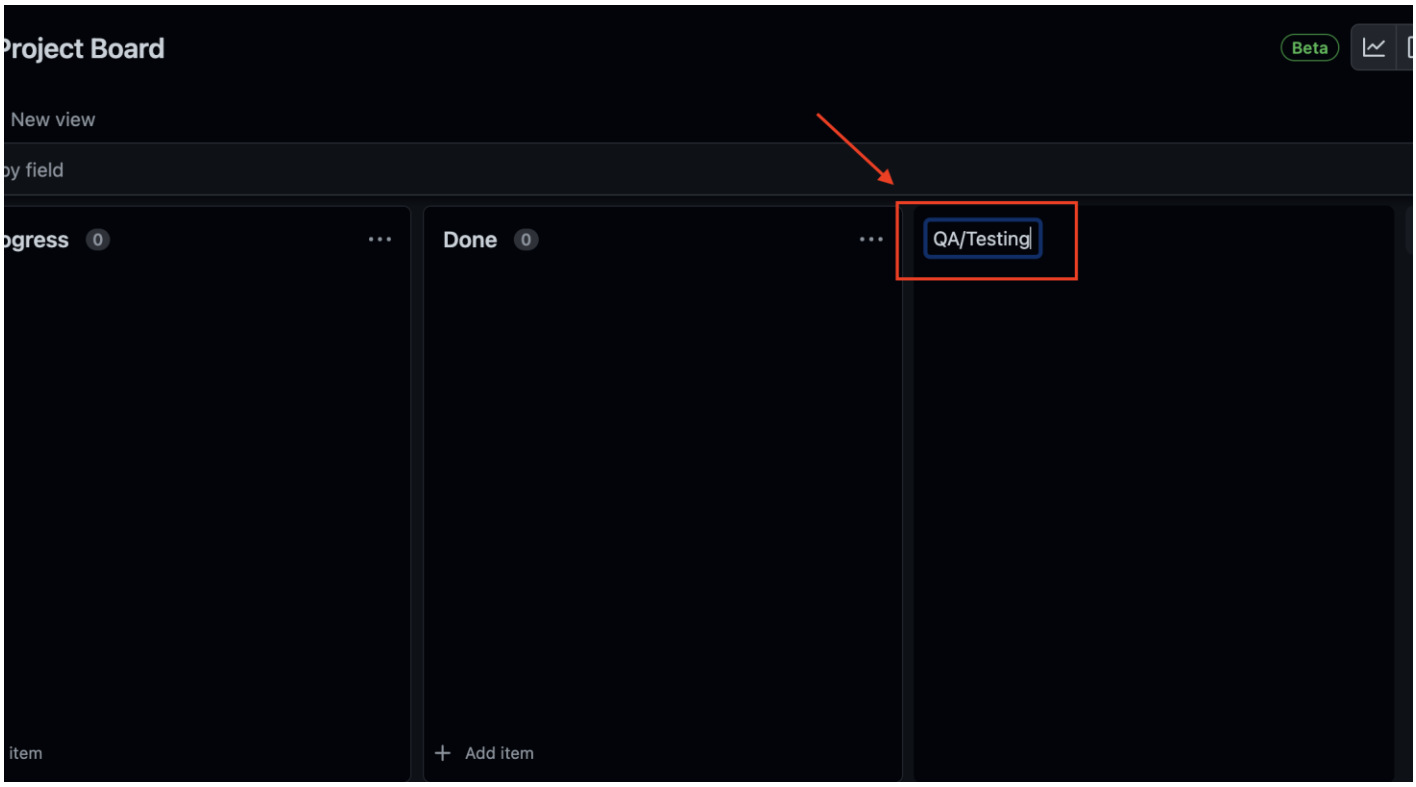
Adding Columns

- **Let's add a new column to our board**
  - The columns help us keep track of where we are at with each ticket or user story in the development process
  - Todo, In Progress, and Done are automatically added, but you can add others
- **Click the + button to the right of the existing columns**



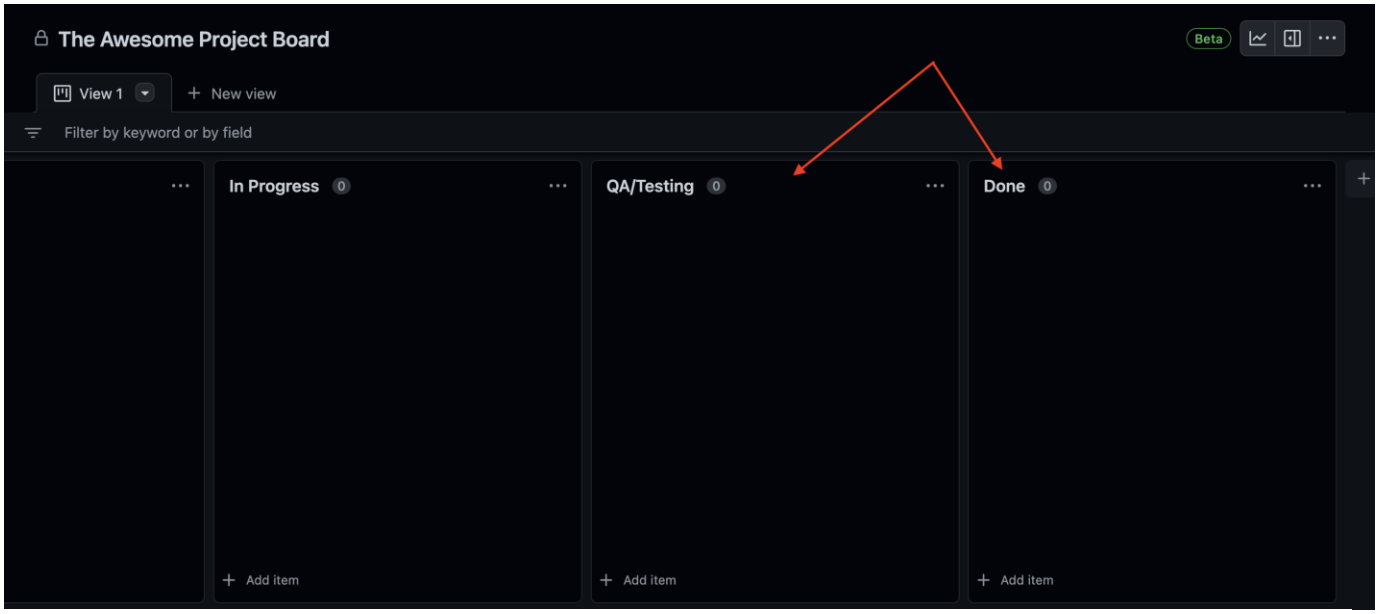
- **After clicking the button, you are prompted to enter a column name**
  - Let's enter QA/Testing

- **NOTE The QA/Testing column is often seen on project boards**
  - It usually represents the state that a task is in as it goes through final testing before being pushed and merged to the main branch on GitHub



Rearranging Columns

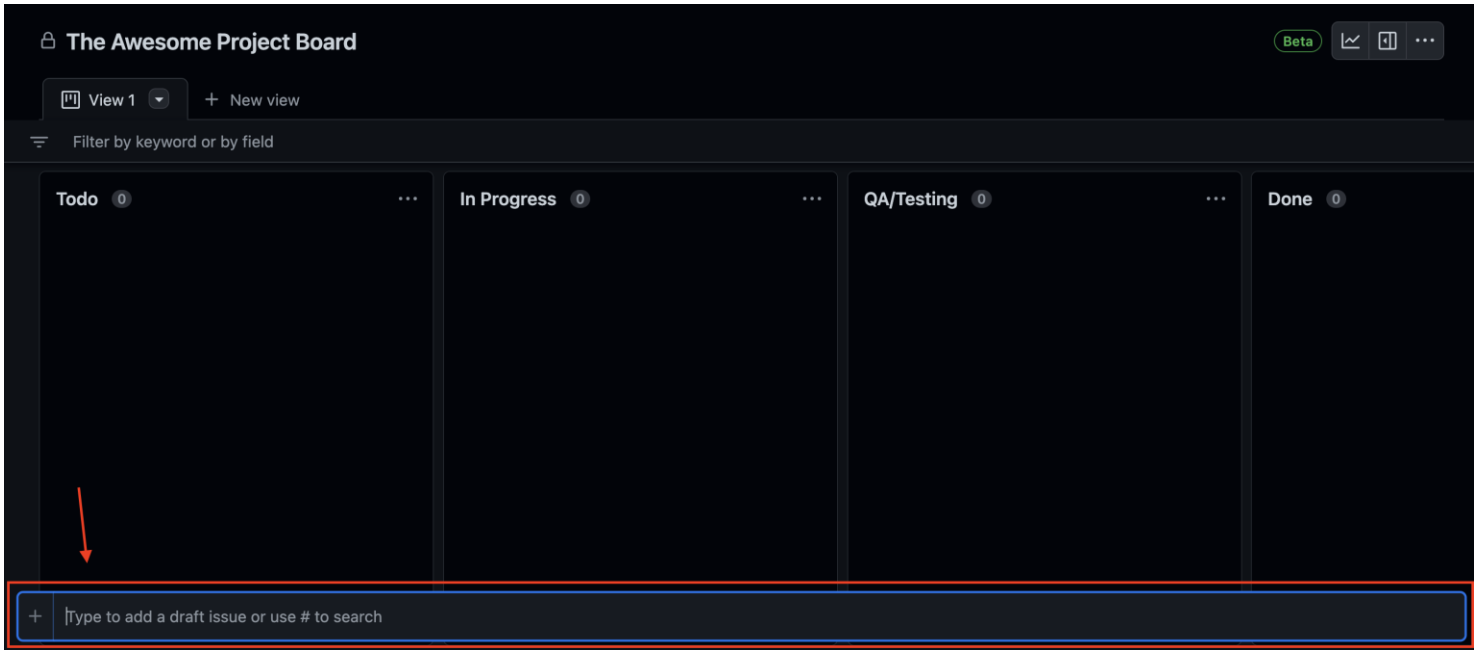
- **We can re-arrange the order of our columns to represent the workflow we want for our tasks**
  - You can move the new `QA/Testing` column to the left of the `Done` column by clicking near the name of the card and dragging it to the spot you want



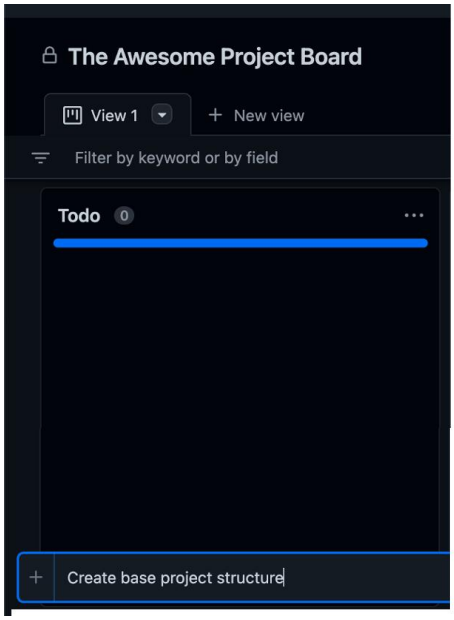
Adding User Stories

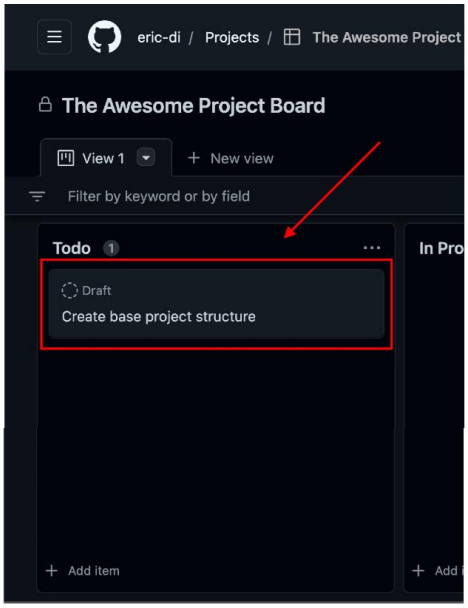
- **Now that we have our board setup, we can add some user stories (or tasks) to our board in the Todo column**
  - Todo is considered our backlog and where all our user stories should be created; we will prioritize the tasks in this column before we start developing our project

- **Click the + Add item link at the bottom of the Todo column**
  - This an input box for us to describe our task



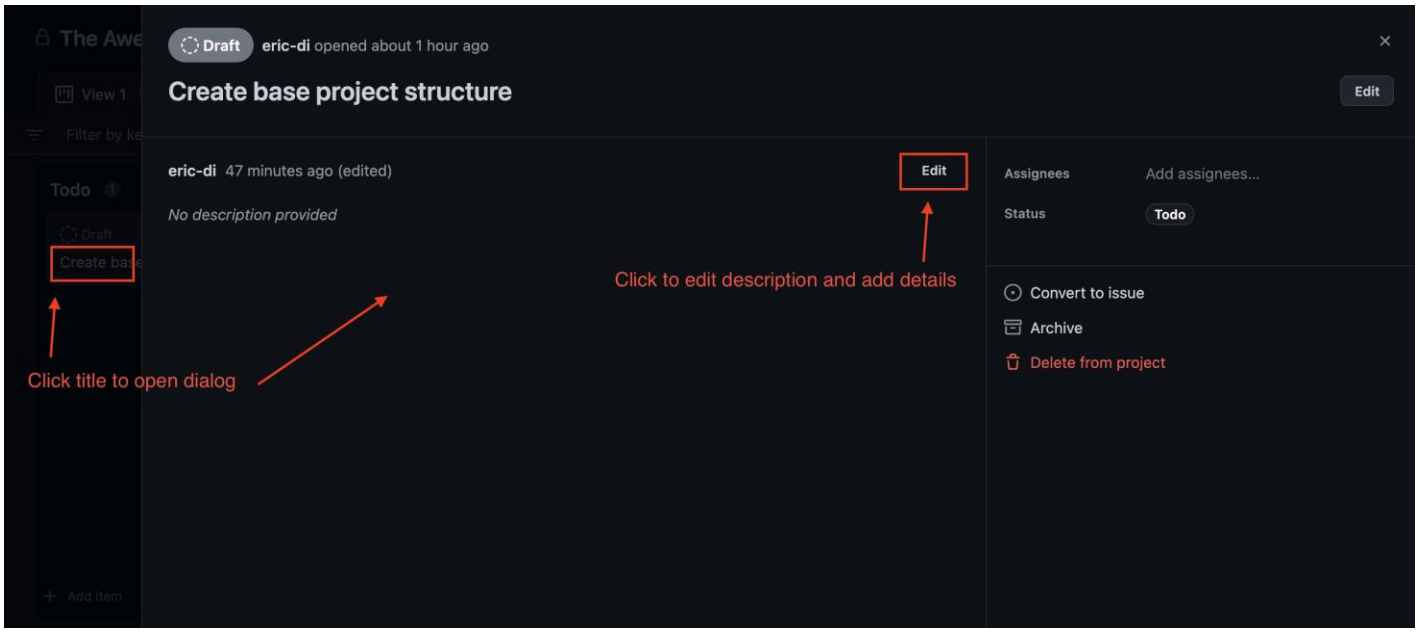
  - Create your first ticket by typing "Create base project structure" in the box and hitting enter
  - This will add the user story to the board as a draft
    * We will be keeping all of our user stories/tasks as drafts
    * GitHub allows you to turn tasks into issues on a specific repository but that is out of the scope of this process so just ignore the word draft above the task title


The Awesome Project Board
View 1
New view
Filter by keyword or by field
Todo 0
Create base project structure

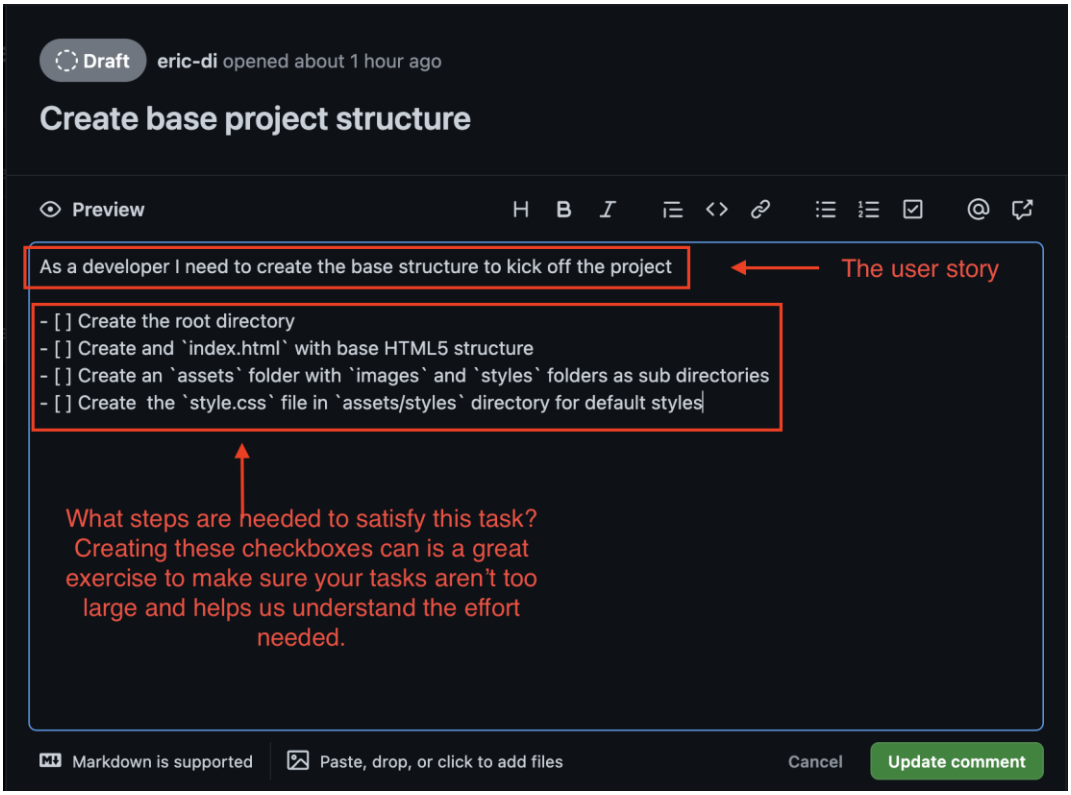

eric-di / Projects / The Awesome Project
The Awesome Project Board
View 1
New view
Filter by keyword or by field
Todo 1
Draft
Create base project structure
Add item

Adding Details to a User Story

- **Once you have created the task, you can click on its title to edit the description and details**
  - Once you click the `Edit` button to update the description of the task, you can add the task details.



- **The details include the user story and acceptance criteria that outline what steps need to be taken to satisfy the task**
  - Once you have added the details, click `Update comment` at the bottom of the form


Draft
eric-di opened about 1 hour ago
Create base project structure
Preview
As a developer I need to create the base structure to kick off the project
The user story
- [ ] Create the root directory
- [ ] Create and `index.html` with base HTML5 structure
- [ ] Create an `assets` folder with `images` and `styles` folders as sub directories
- [ ] Create the `style.css` file in `assets/styles` directory for default styles
What steps are needed to satisfy this task? Creating these checkboxes can is a great exercise to make sure your tasks aren't too large and helps us understand the effort needed.
Markdown is supported
Paste, drop, or click to add files
Cancel
Update comment

Sample Markdown for Starting User Stories

- **Here is some markdown you can use to create a few tasks**
  - Use these tasks on your project board to get you started
  - There will be many more tasks you will add if you use the board correctly during the planning

Title - Create base project structure
Description - As a developer, I need to create the base structure to kick off the project

- [ ] Create the root directory
- [ ] Create an `index.html` with base HTML5 structure
- [ ] Create `css`, `images` and `scripts` folders as subdirectories
- [ ] Create `style.css` file for default styles

> **Note** At any time you can click the task's title to see its details.

Title - Create local repo and link it to GitHub
Description - As a developer, I want to keep track of my changes to the project code over time and have it linked to GitHub so I have a backup of the codebase.

- [ ] Run `git init` to create local repo
- [ ] Make the first commit using `git add .` then `git commit -m "first commit" `
- [ ] Follow instructions on GitHub to link local and remote repo

Title - Link Bootstrap to the project
Description - As a developer, I want to link bootstrap to the site so I can leverage the UI libraries responsiveness features and components for rapid development

- [ ] Follow instructions at https://getbootstrap.com/docs/5.2/getting-started/introduction/#quick-start

Title - Add Bootstrap Navbar to home page
Description - As a developer, I want to create a Navbar to move between my pages

- [ ] Design Navbar using Bootstrap reference docs and provide links placeholders to each of my pages that I will build

Final Thoughts

- **Create all the tasks you think you will need and validate them with user stories**
- **Prioritize the tasks so that what you need for your MVP (Minimum Viable Product) is at the top of the list**
- **Commit and push to version control often**
  - Make sure you commit after closing each task, but you may choose to make commits more often
- **Move your tasks through the workflow**
  - The task(s) you are actively working on should be moved to the In Progress column
  - Ultimately, the tasks will be completed and should be moved to the Done column

Exercises

EXERCISE 1

In this exercise you will create a GitHub project board for the Hardware Store user stories that you created earlier.

Step 1 - Log into GitHub.com and create a new project board. Name the board `Hardware Store Project`.

Step 2 - Add a new `Testing` column to the board

Step 3 - Add a card for each of the User Stories that you created to the `ToDo` list (the stories that you created in the `Module 2` exercise)

Step 4 - Modify your User Story Cards to give them a description. Use the description field to explain the details of the story.