



YOUR GUIDE TO SETTING UP A GITHUB PROFILE AND JOINING ULI



WHAT IS GITHUB?

Github is a collaborative platform for software developer and their teams to store and manage their projects.

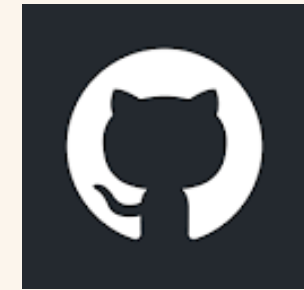
For Developers:

Github allows developers to work together on the software code, track its development and progress.

For Non-Tech Team Members:

While non-developers don't contribute to the code directly, they can use its project management features to keep track of their contributions (whether design, documentation, content creation etc).

Github therefore provides a full picture of work on the table, and relevant team members can keep track and update their code and other contributions in one place.

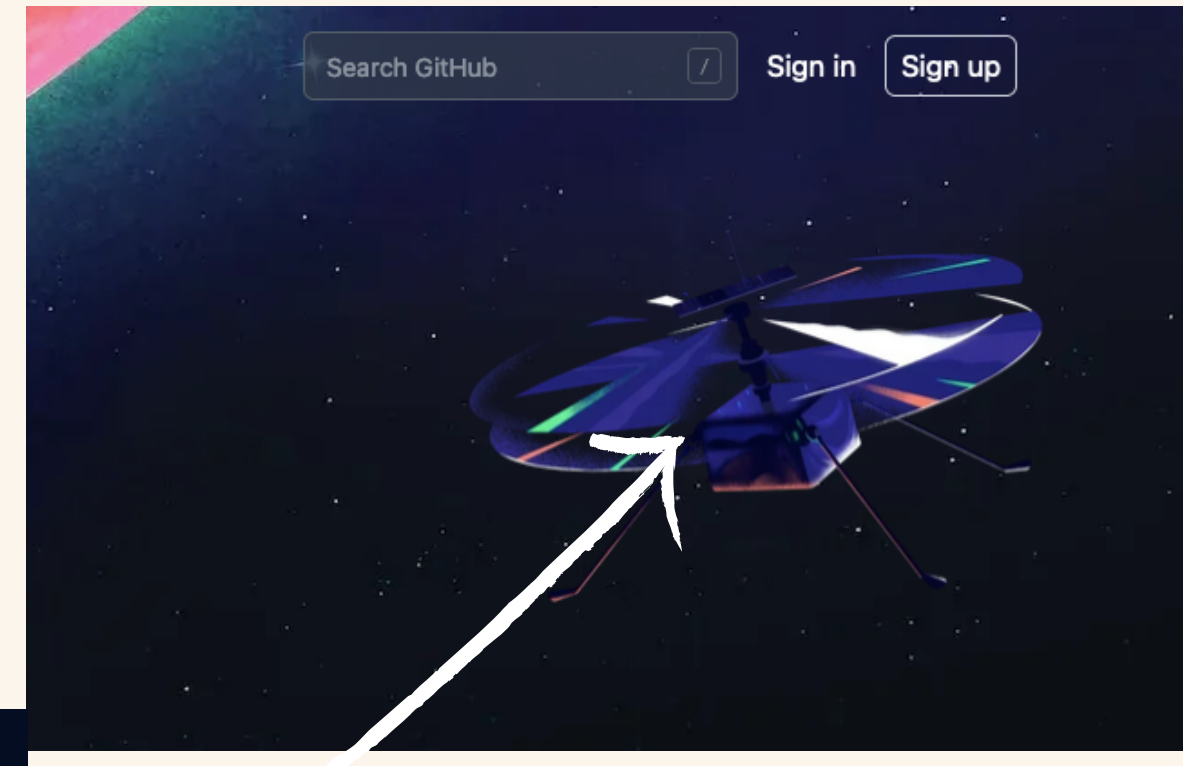


Why is Uli using Github?

- to manage the development of its software
- to keep a track of tasks, deadlines and manage expectations
- to also engage in discussions that could improve it.

STEP ONE: SIGN UP

- Go to Github.com
- Click on the sign up option

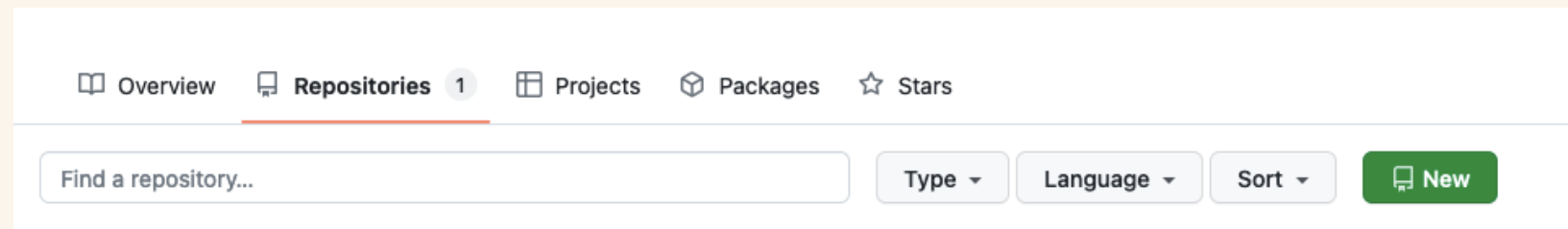


- Enter the details requested.
- Once the verification is complete, you will be directed to your Account page.

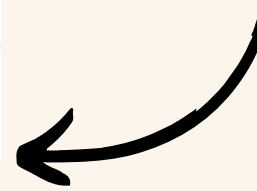
For reference, you can also take a look at this link: <https://docs.github.com/en/get-started/onboarding/getting-started-with-your-github-account>

A screenshot of the GitHub sign-up form. The form is titled 'Welcome to GitHub! Let's begin the adventure'. It contains three input fields: 'Enter your email*', 'Create a password*', and 'Enter a username*'. The email and password fields have green checkmarks next to them, indicating successful input. The username field has a pink arrow next to it. A 'Continue' button is located at the bottom right of the form. A white arrow points from the right side towards the password field.

CREATING YOUR 'ABOUT ME' PAGE



Once you've set up your profile, create a new repository with your Github username





Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*

Owner * Repository name *

  /

Great repository names are short and memorable. Need inspiration? How about **stunning-guacamole** ?

Description (optional)

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

☐ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

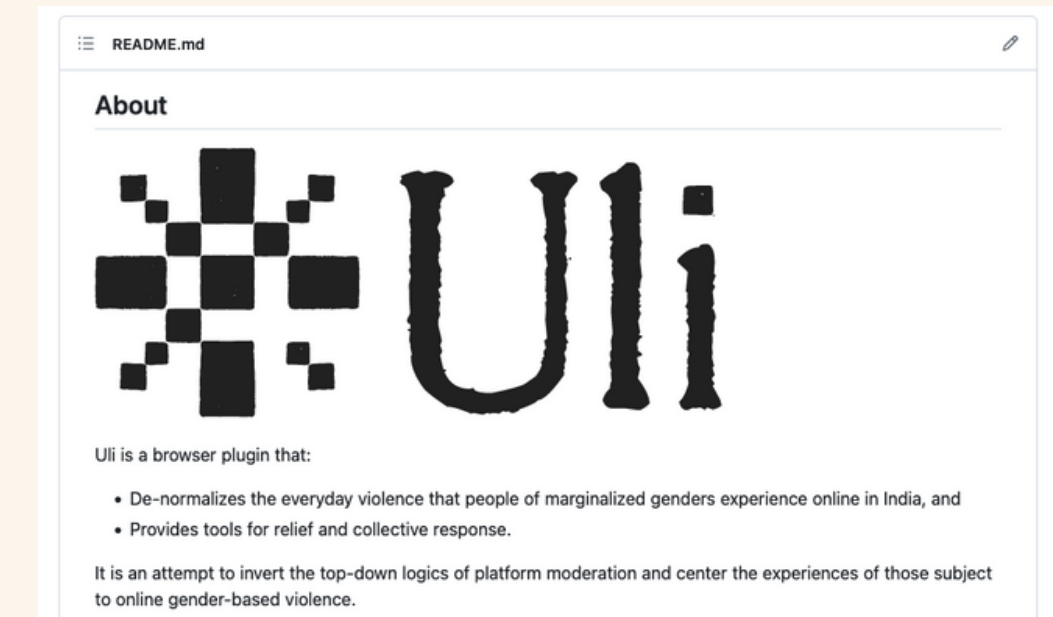
Add .gitignore

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

- Name your repository the same as your username.
- Keep this repository **public**.
- Check the box to initialise the repository as a ReadMe file. ReadMe files allow you to describe your project/profile.
- Add a brief description about your professional profile, your interests, and any other information you'd like to share. There is no set rule!
- Here's an introduction to ReadMe files:
<https://docs.github.com/en/repositories/managing-your-repositorys-settings-and-features/customizing-your-repository/about-readmes>

- Once you've set up your profile, go to Uli's Github profile:
<https://github.com/tattle-made/Uli>
- Here you'll find more details about the project- reach out to the team and request them to add you to the project.



Uli's ReadMe

- Check out the Uli project's Github page, and scroll down to read more about it.

WHAT IS A REPOSITORY?

A repository is like a folder, within which tasks are categorised based on topic, and function.

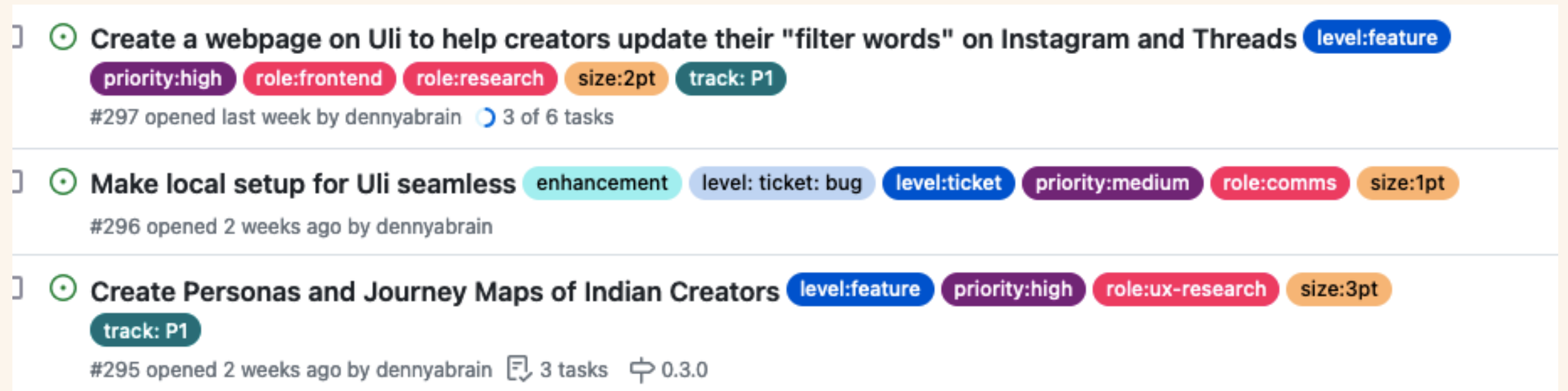
For example, we have a repository for Uli.

Uli is the folder, within which different subsets of work are created as projects- if we were to create a website for Uli, we would create a project titled "Website for Uli", under which we create tasks to be completed.

So where do we create these tasks on Github? In the **Issues** section.

WHAT ARE ISSUES?

Issues are tasks, bugs, reports- i.e. everything in a project that needs to be fixed, completed or improved.



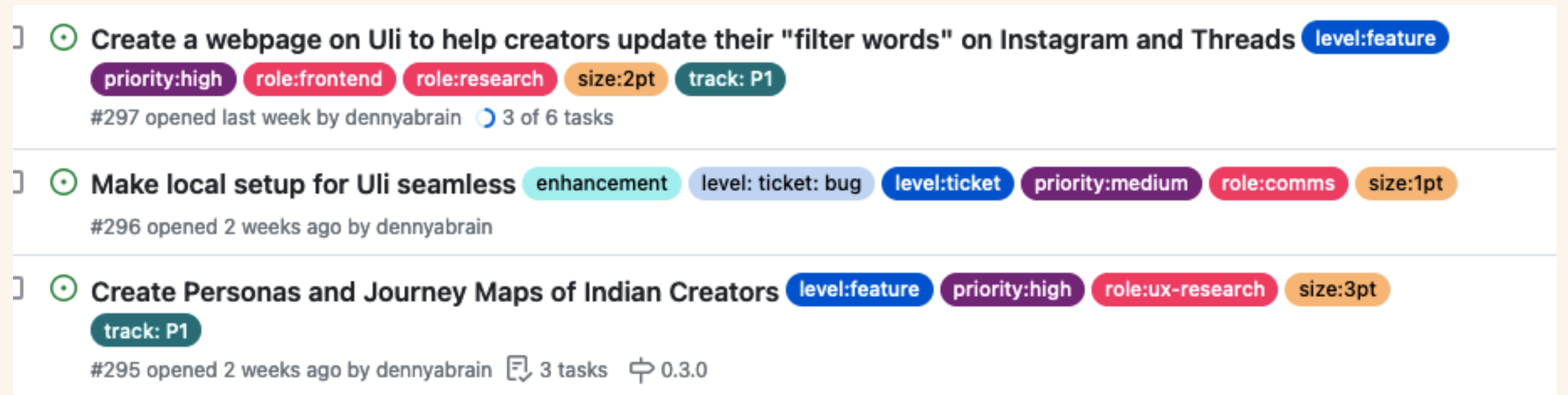
Now where do you keep track of all these issues and how they are progressing?

The Projects Tab.

WHAT ARE LABELS?

When you look at the Issues tab on the repository, you'd see tags beneath the title of different issues. Let's relook the image from the last slide.

The bubble boxes in different colours, with priority, roles, size etc are **labels**.



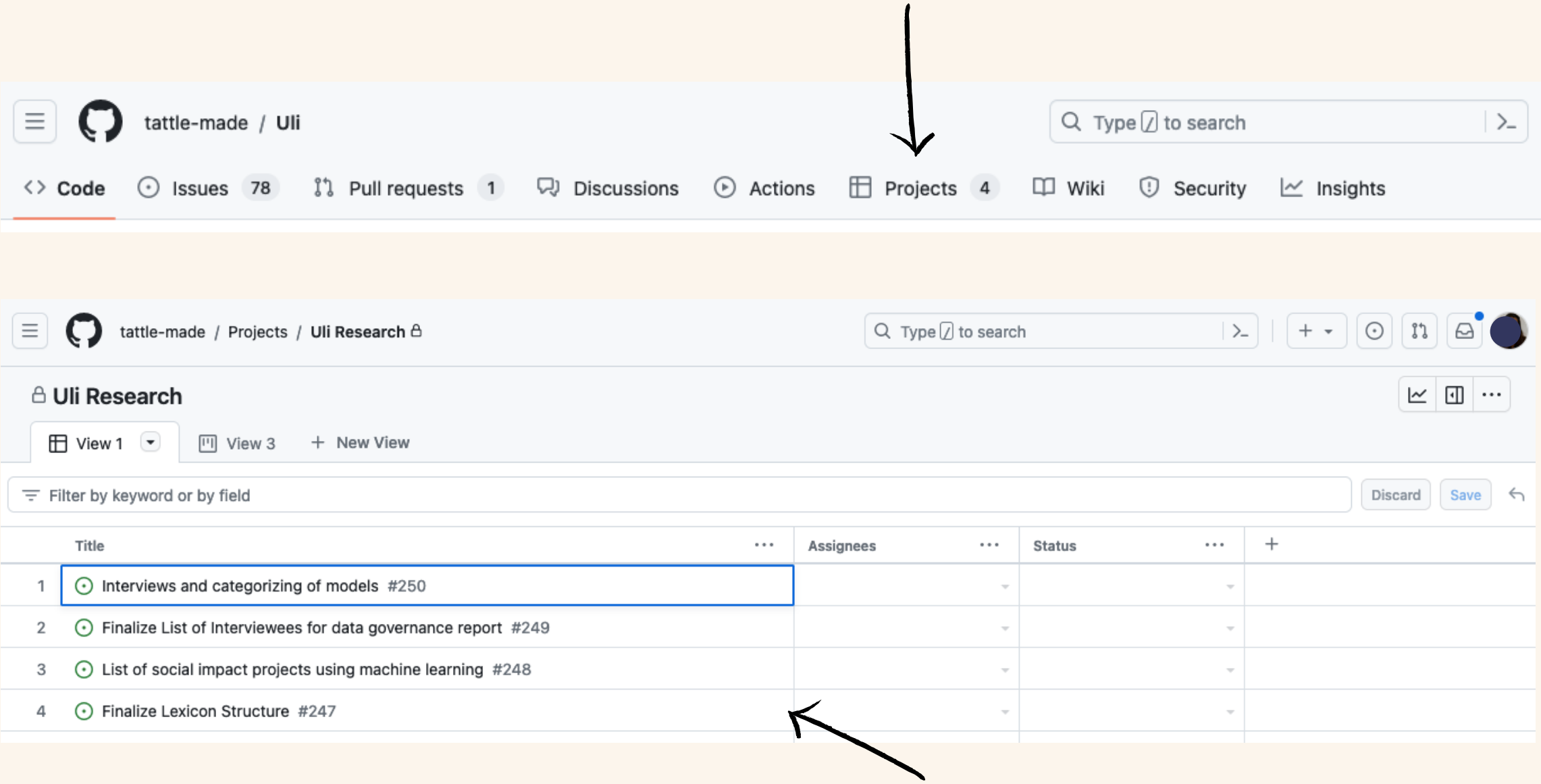
Labels are used to categorise issues in a repository.

For instance, labels such as 'role' help people from different roles identify which issues apply to them. Project team members decide what labels they require to make categorisation and execution of issues easier. While some are default labels, most teams can make custom labels that they believe would work for them.

WHAT CAN YOU DO ON THE PROJECTS TAB?

The projects tab gives you an organized view of all the work on hand, and you can get started on those that apply to you.

If you click on any of the items, you'll be redirected to the Issues tab, where you will see more details about what the issue is, and the process for rectifying it.

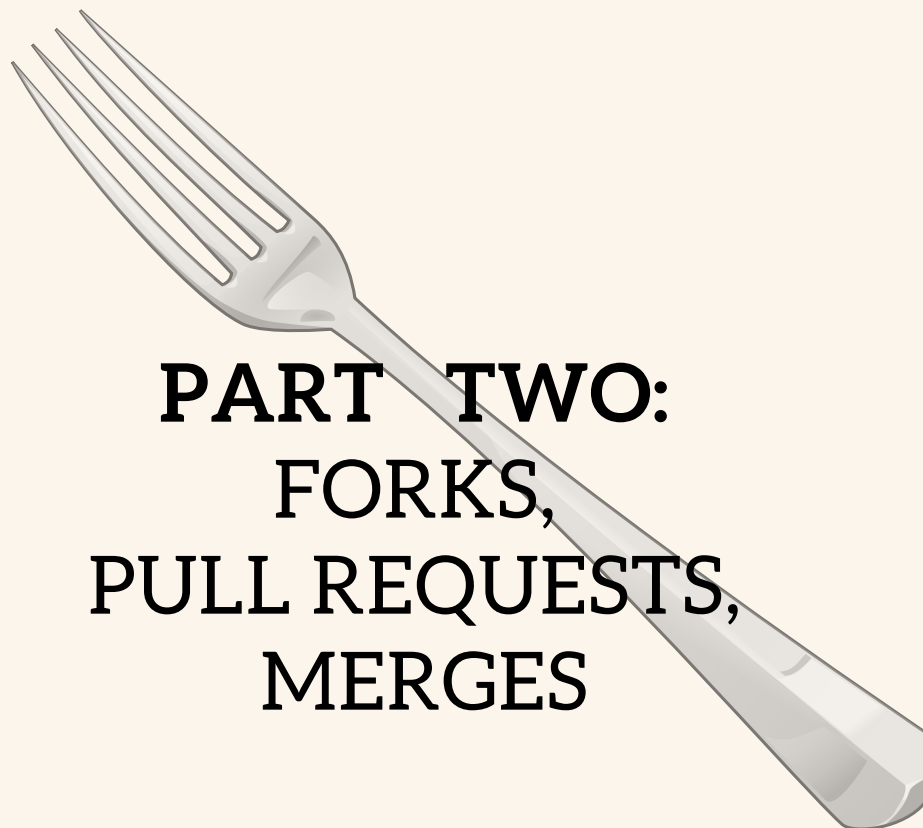


If you want to read more about the Projects tab, take a look at this:
<https://docs.github.com/en/issues/planning-and-tracking-with-projects/learning-about-projects/about-projects>

PRACTICE EXERCISES

Now that you have a fair idea of Github, here's some exercises we would like you to try:

- **Create your first issue:** Create an Issue titled 'Introduce Yourself- [Insert Name]'. In the description box, introduce yourself to the team in a few sentences.
- **Adding labels:** Check the issues in the repository, you may see there are labels in different colours and description (ex: "documentation"). To the issue you created above, add the label 'good first issue'.
- **Star the Uli repository:** There is an option to 'star' a repository to make it easier for you to navigate to it later on, open the Uli repository and click on this option so you have it saved.



**PART TWO:
FORKS,
PULL REQUESTS,
MERGES**

FORKING A REPOSITORY

When you need to make changes to the codebase, you 'fork' a repository. This means that you make a copy of the main codebase, and in this copy, you make the requisite changes..

Forking allows changes to be made without direct changes being made to the codebase, which would impact the functionality of the software if issues arise during the process.



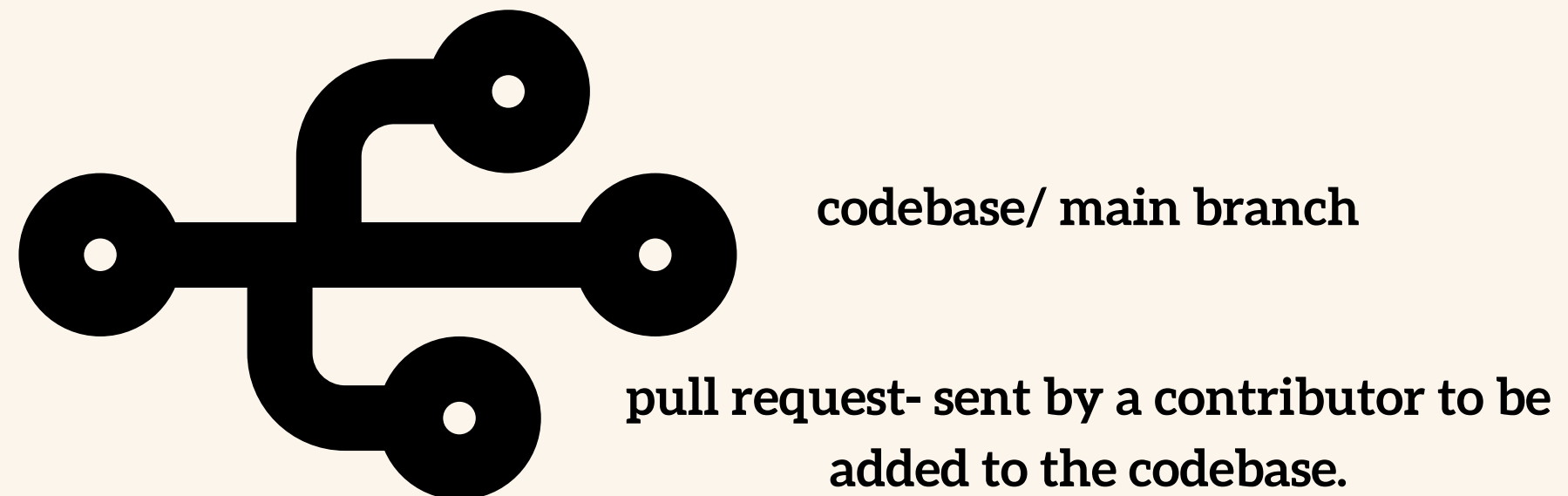
PULL REQUESTS

Every project has one main code file- let's call this the main/ codebase.

This codebase cannot directly be changed/worked on by any contributor.

If a change needs to be made to it, then you send your changes as a 'request', where someone on the team who reviews these suggestions will take a look, and allow it to pass through to the next level, and include it to the main codebase.

When you submit/contribute to the project, and need the codebase to be changed, you are sending a 'pull request'.



PRACTICE EXERCISE

Submit a Pull Request: Choose your first issue, ideally one with a small size for this exercise, fork the repository, and submit a pull request.



✱Uli