**1. Add Description (with Emojis & Link)**

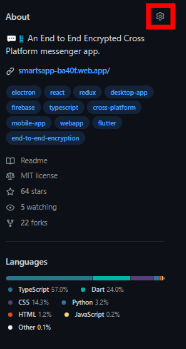
This is the **lowest hanging fruit** in creating an impressive **GitHub repository**. You can just add a short description of the **repository** to heighten its appeal. Make sure to throw in some **emojis** too 😉.

You also get **extra brownie points** 🎂 for adding **relevant links** too!

Text

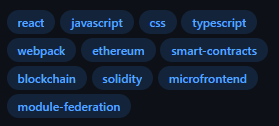
Description automatically generated

You can update the **description** and the **link** from the top right **settings icon**



**2. Add Tags**

Adding **tags** is the next and most crucial step in this article. It not only makes it easier for **GitHub** to promote the **repository** but also makes it **look much more professional**!



**Tag** related settings are also available in the **repository configuration** shown in the previous step

**3. Add License**

No point denying it, this is the most boring step, but essential for **fair use of the repository**. Moreover, it also shows up in the info tab, so just get it done with.



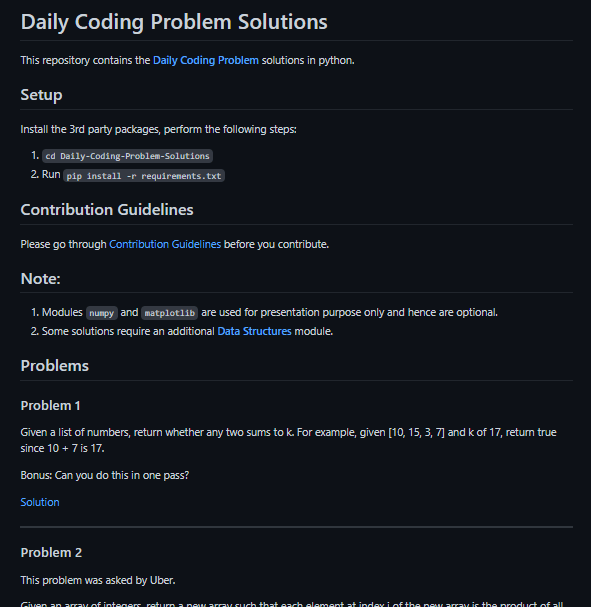
To add a **License**, create a file called LICENSE at the **root of the project**

**4. Add Readme**

The **ReadMe** is the first thing a visitor sees when they visit the **repository**, so taking some time to create a polished **repository** is crucial. It should describe **what problem the code solves**, **how to set it up**, and any **other relevant information**.



There is always some way to make the **ReadMe** engaging. Even if you don’t have much to write about the **repository** itself. Take the example of a **repository** containing **coding problem solutions**. You can easily **list out the problems** in the **ReadMe** and **link to the solutions**.



To add a **ReadMe**, create a ReadMe.md file at the **root of the project**.

**5. Use Descriptive Commit Messages**

Using descriptive **commit messages** might look like a chore, but it too helps in taking the appeal of the repository higher. Make sure the **commit message** is explanatory.

There are various schools of **commit message** structures, any of which is acceptable. I personally like the following one:

<type>[optional scope]: <description>

If you want to dive deeper into how it is structured, check out [this article](https://www.conventionalcommits.org/en/v1.0.0-beta.2/)

A screenshot of a computer

Description automatically generated with medium confidence

**6. BONUS: Use Multiple Languages**

Using **multiple languages** in your repository is a great way to grab the user’s attention, but this is highly context-dependent. You should **NOT** use **multiple languages** only to increase the appeal, but only if it is required.

Graphical user interface, text, application

Description automatically generated

Even if you falter in this department, as long as you implement the other tricks, **you are completely covered**!