# Contributing

We would love for you to contribute and help make it even better

than it is today! As a contributor, here are the guidelines we would like you

to follow:

- [Question or Problem?](#question)

- [Issues and Bugs](#issue)

- [Feature Requests](#feature)

- [Submission Guidelines](#submit)

- [Coding Rules](#rules)

- [Commit Message Guidelines](#commit)

## <a name="question"></a> Got a Question or Problem?

Please, do not open issues for the general support questions as we want to keep GitHub issues for bug reports and feature requests.

## <a name="issue"></a> Found an Issue?

If you find a bug in the source code or a mistake in the documentation, you can help us by

[submitting an issue](#submit-issue) to our [GitHub Repository][github]. Even better, you can

[submit a Pull Request](#submit-pr) with a fix.

## <a name="feature"></a> Want a Feature?

You can \_request\_ a new feature by [submitting an issue](#submit-issue) to our [GitHub

Repository][github]. If you would like to \_implement\_ a new feature, please submit an issue with

a proposal for your work first, to be sure that we can use it.

First open an issue and outline your proposal so that it can be

discussed. This will also allow us to better coordinate our efforts, prevent duplication of work,

and help you to craft the change so that it is successfully accepted into the project.

\*\*All features require a proper design and review by team members and product owners.\*\* Before starting work, you might want to

discuss with us to figure out:

- Is this something we want?

- What's the impact?

Answering those questions first in the request might help us make a decision.

## <a name="submit"></a> Submission Guidelines

### <a name="submit-issue"></a> Submitting an Issue

Before you submit an issue, please search the issue tracker, maybe an issue for your problem already exists and the discussion might inform you.

We want to fix all the issues as soon as possible, but before fixing a bug we need to reproduce and confirm it. Having a reproducible

scenario gives us wealth of important information without going back & forth to you with additional questions.

You can file new issues by filling out our [new issue form](https://github.com/vuesion/vuesion/issues/new).

### <a name="submit-pr"></a> Submitting a Pull Request (PR)

Before you submit your Pull Request (PR) consider the following guidelines:

- Search [GitHub](https://github.com/vuesion/vuesion/pulls) for an open or closed PR

that relates to your submission. You don't want to duplicate effort.

- Make your changes in a new git branch:

```shell

git checkout -b my-fix-branch next

```

- Create your patch, \*\*including appropriate test cases\*\*.

- Run the full test suite,

and ensure that all tests pass.

- Commit your changes using a descriptive commit message that follows our

[commit message conventions](#commit). Adherence to these conventions

is necessary because release notes are automatically generated from these messages.

```shell

git commit -a

```

Note: the optional commit `-a` command line option will automatically "add" and "rm" edited files.

- Push your branch to GitHub:

```shell

git push origin my-fix-branch

```

- In GitHub, send a pull request to `vuesion:next`.

- If we suggest changes then:

- Make the required updates.

- Re-run the test suites to ensure tests are still passing.

- Rebase your branch and force push to your GitHub repository (this will update your Pull Request):

```shell

git rebase next -i

git push -f

```

That's it! Thank you for your contribution!

#### After your pull request is merged

After your pull request is merged, you can safely delete your branch and pull the changes

from the main (upstream) repository:

- Delete the remote branch on GitHub either through the GitHub web UI or your local shell as follows:

```shell

git push origin --delete my-fix-branch

```

- Check out the next branch:

```shell

git checkout next -f

```

- Delete the local branch:

```shell

git branch -D my-fix-branch

```

- Update your next branch with the latest upstream version:

```shell

git pull --ff upstream next

```

## <a name="rules"></a> Coding Rules

To ensure consistency throughout the source code, keep these rules in mind as you are working:

- All features or bug fixes \*\*must be tested\*\* by one or more specs (unit-tests or e2e-tests).

## <a name="commit"></a> Commit Message Guidelines

We have very precise rules over how our git commit messages can be formatted. This leads to \*\*more

readable messages\*\* that are easy to follow when looking through the \*\*project history\*\*. But also,

we use the git commit messages to \*\*generate the change log\*\*.

### Commit Message Format

Each commit message consists of a \*\*header\*\*, a \*\*body\*\* and a \*\*footer\*\*. The header has a special

format that includes a \*\*type\*\*, a \*\*scope\*\* and a \*\*subject\*\*:

```

<type>(<scope>): <subject>

<BLANK LINE>

<body>

<BLANK LINE>

<footer>

```

The \*\*header\*\* is mandatory and the \*\*scope\*\* of the header is optional.

Any line of the commit message cannot be longer 100 characters! This allows the message to be easier

to read on GitHub as well as in various git tools.

### Revert

If the commit reverts a previous commit, it should begin with `revert:`, followed by the header of the reverted commit. In the body it should say: `This reverts commit <hash>.`, where the hash is the SHA of the commit being reverted.

### Type

Must be one of the following:

- \*\*build\*\*: Changes that affect the build system or external dependencies

- \*\*ci\*\*: Changes to our CI configuration files and scripts

- \*\*docs\*\*: Documentation only changes

- \*\*feat\*\*: A new feature

- \*\*fix\*\*: A bug fix

- \*\*perf\*\*: A code change that improves performance

- \*\*refactor\*\*: A code change that neither fixes a bug nor adds a feature

- \*\*style\*\*: Changes that do not affect the meaning of the code (white-space, formatting, missing

semi-colons, etc)

- \*\*test\*\*: Adding missing tests or correcting existing tests

- \*\*chore\*\*: Maintenance

### Scope

The scope should be the name of the module affected as perceived by the person reading changelog generated from the commit messages.

### Subject

The subject contains succinct description of the change:

- use the imperative, present tense: "change" not "changed" nor "changes"

- don't capitalize first letter

- no dot (.) at the end

### Body

Just as in the \*\*subject\*\*, use the imperative, present tense: "change" not "changed" nor "changes".

The body should include the motivation for the change and contrast this with previous behavior.

### Footer

The footer should contain any information about \*\*Breaking Changes\*\* and is also the place to

reference GitHub issues that this commit \*\*Closes\*\*.

[github]: https://github.com/vuesion/vuesion/issues