GitHub Contributions

Branching Strategy

We will be making use of a **GitFlow-based branching strategy**, with a layout that looks as follows:

```
-> main
    -> dev
        -> ui
        -> docs
        -> api
        -> docker-container
            -> derived branch A
        -> ai-bot
            -> derived branch B
```

Follow the Chain (VERY IMPORTANT)

Please:

```
> ❌ DO NOT COMMIT TO MAIN/DEV
> ❌ DO NOT MERGE TO MAIN/DEV
> ❌ DO NOT BRANCH FROM MAIN/DEV
```

Explaining main and dev:

- **main** is our production branch that should always be in a working condition. Direct changes that do not follow our branching strategy can introduce issues on the version that other people might see.
- **dev** is the branch where all other branch additions are combined into an almost working state. In this branch all *minor* bugs and integrations are fixed before sending to main.

Our **dedicated DevOps team-member** will be managing these two branches and taking all new changes from the respective **feature branches** through dev to main.

Creating Branches

Each person is allowed to create personal/feature branches, as long as:

- They branch off one of the **feature branches**.
- The branches are created from an **Issue**. See the Issues section below if you are unsure how to do this.
- They **use issues to create the branch**. This will help the DevOps team manage everything.

Deleting Branches

When you're done with an issue and it has been merged, you can select the option to delete the branch along with closing the issue.

Otherwise, if you wish to unassign yourself from the issue, you may delete the branch and update the project board **without closing the issue** to do so.

Merging and Pull Requests

When to Merge

Before making a pull request, you should **merge the feature branch** you are branching from into your current branch.

Use the command:

```
git pull origin <branchname>
```