1. **Technical Stack**

|  |
| --- |
| **.NET Core Web API** |
| **SQLite database** |
| **IIS** |
| **OS (Windows)** |

* Source Control: Git
* CI/CD Tool: Jenkin, Gitlab
* Test tool: Nunit, Selenium

1. **Design**

* Git repositories
* CI/CD process

[**Source stage**]

* + - Each developer work on their repository.
    - When changes are made on these branches, pull the latest source from the Development branch before code compilation and unit testing will be triggered automatically.
    - Once all of the unit tests are passed, changes on these branches will be committed to the Development branch.

[**Build stage**]

* + - Each time a commit is merged to the Development branch, Jenkins triggers a build event. If there is no error, commit go to the test stage, otherwise, roll back to the previous version.

**[Test stage]**

* + - Integration testing, Functional testing, and UI testing will be triggered after the build stage. These tests ensure that no bug can reach end-users.
    - Once all of the tests are passed, changes will be committed to the Master branch.

**[Deployment stage]**

* + - Every time changes are merged on the Master branch, a new release branch with a version number will be created.
    - The deployment stage’s output includes infrastructure provisioning, configuration, and containerization, so it’s ready to be released to the target system.

1. **Critical spots**

* Although CI/CD aims at automation, developers are still responsible for self-review their source code and doing unit testing.
* Before triggering any automated task, synchronize changes on the base repository to the current branch.
* Test cases and configurations need to be carefully prepared to avoid any problem during the process.

1. **Potential area of complexities**

* Decide the need for automation of certain tasks: Do CI/CD make the process faster than doing it manually?
* Design test cases for an automated pipeline is not easy.