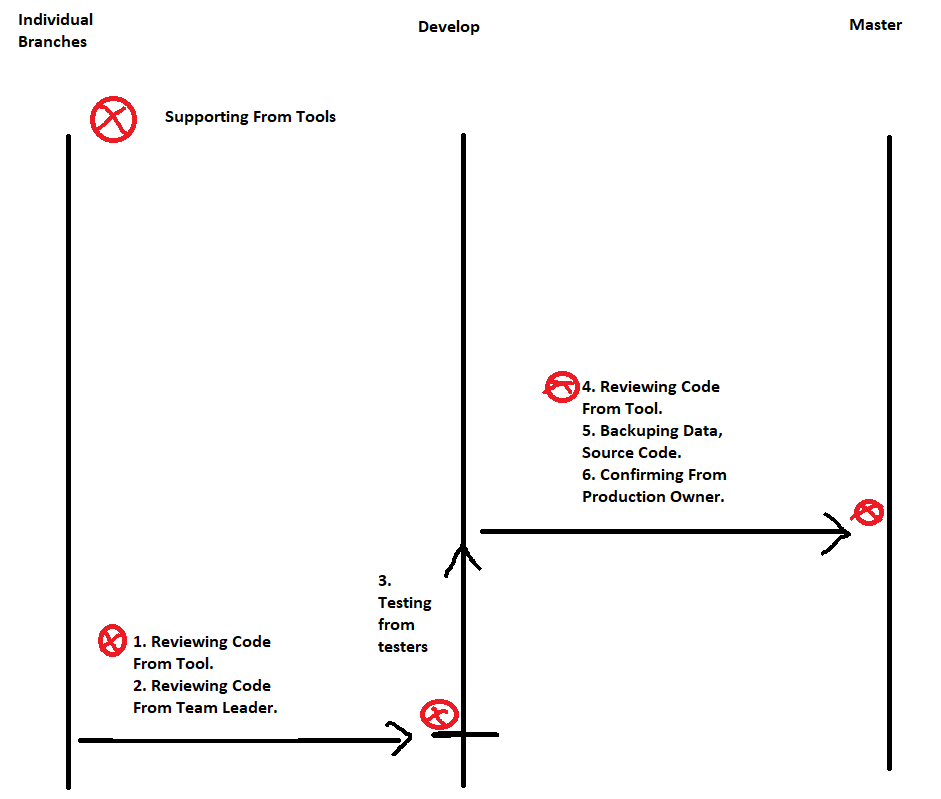
**Table of contents**

[1. Technology Stack 1](#_Toc11392)

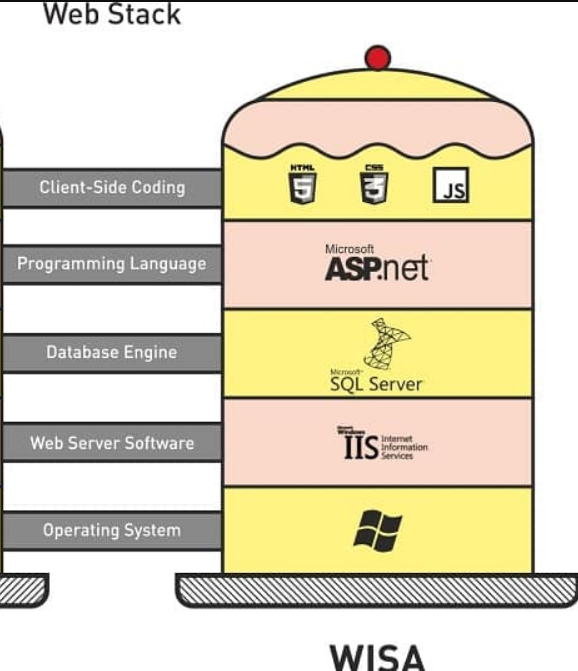
[2. Design. 1](#_Toc6879)

[3. Critial Spots: 2](#_Toc5791)

[4. Potential Area Of Complexities: 2](#_Toc21433)



1. **Technology Stack**



- OS: Window

- Cloud: Azure

- Server: IIS

- Database: MSSQL

- Framework: ASP.Net Core

- Front end: HTML, CSS, JavaScript, JQuery, Razor, Bootstrap, Angular, React, Vue,…

- Code reviewing tool: SonaQuebe

- CI/CD Tool: Jenkin, Gitlab

1. **Design.**

**- Git Branch**

+) master

+) develop

+) individual branches

**- Server**

+) Production

+) Development

**- Overall Steps**

1. Developers execute the code and push the code to the Individual Branches.
2. The developer makes a merge request into the “develop” branch.
3. Team Leader reviews the code for syntax, logic, optimization, then performs a confirm merge code into the “develop” branch.
4. The system does a review of the code for syntax, and if possible, checks for logic and optimization.
5. The system automatically deploys the source code from the "develop" branch to the development server.
6. Tester executes test on development server.
7. Owner creates a merge request from branch "development" to branch "master".
8. The system performs source code review. If pass, enable to allow merge and deploy, wait for the owner of the production server to confirm.
9. Owner performs a confirm merge request from branch "develop" to branch "master". The current source code on the production server is backed up, the new source code is merged into the master branch, then the system automatically deploys to the production server.
10. Tester on production server to perform UAT and confirm.
11. (OPTIONAL) If an unexpected error occurs, the Owner can revert the version code to the previous version.
12. **Critical Spots:**
13. The self-review of the source code of the developers.
14. The team leader's source code reviewing.
15. The source code reviewing of the source code check tool.
16. Tester's testing needs to have high logic coverage.
17. Choosing when to perform deploy. Deployment cannot take place at peak times of usage, affecting customers' work.
18. Checking the logic when deploying, the deployment is easy to affect the data of the production server, so a suitable backup strategy is needed.
19. **Potential Area Of Complexities:**

- The implementation of the design and writing of unit test test cases by developers is quite complicated, requires a lot of experience and time to learn as well as time to implement.

- The execution of design and writing of system test cases by testers to get business logic coverage is complex, requiring a deep understanding of the customer's business.

-The automatic code review tools are quite new and often change and update, installing these tools requires many complicated and configuration steps.