## VS Code Extension Evaluation Guide

### Evaluation Criteria

#### 1. Functionality

- \*\*What to Check\*\*:

- Review the extension's description on the [Visual Studio Code Marketplace](https://marketplace.visualstudio.com/vscode).

- Test core functionalities in a local VS Code instance.

- Gather feedback from users who requested the extension.

- \*\*Steps\*\*:

1. Go to the extension’s page on the VS Code Marketplace.

2. Read through the description and feature list.

3. Install the extension in a test environment.

4. Verify that all advertised features work as expected.

#### 2. Compatibility

- \*\*What to Check\*\*:

- Ensure compatibility with our current VS Code version.

- Verify it works with other commonly used extensions.

- \*\*Steps\*\*:

1. Check the "Version History" tab on the Marketplace page for compatibility details.

2. Test the extension with our current VS Code setup.

3. Document any compatibility issues encountered.

#### 3. Check for Existing Extensions

- \*\*What to Check\*\*:

- Verify if a similar extension with the same functionality is already available in our private marketplace.

- \*\*Steps\*\*:

1. Search our private marketplace for extensions with similar functionality.

2. Compare the requested extension with the existing one to determine if the new one offers any additional benefits.

3. Decide if the new extension is necessary or if the existing one suffices.

#### 4. Check for Alternative Extensions

- \*\*What to Check\*\*:

- Look for alternative extensions that offer similar functionality.

- Compare alternatives based on the same evaluation criteria.

- \*\*Steps\*\*:

1. Search the [Visual Studio Code Marketplace](https://marketplace.visualstudio.com/vscode) for extensions with similar functionality.

2. Verify the alternative’s publisher is verified.

3. Compare support and maintenance details.

4. Choose the best option based on overall evaluation.

#### 5. Integration Issues

- \*\*What to Check\*\*:

- Verify the extension integrates well with our existing systems and tools (e.g., Jira, Checkmarx).

- Test functionalities specific to our environment.

- \*\*Steps\*\*:

1. Install the extension and perform integrations with relevant systems.

2. Test key features, such as creating issues in Jira or viewing reports in Checkmarx.

3. Document any integration issues and assess their impact.

#### 6. Usability

- \*\*What to Check\*\*:

- Assess the user interface and ease of use.

- Look for intuitive design and clear instructions.

- \*\*Steps\*\*:

1. Use the extension and navigate through its features.

2. Note any usability issues or areas for improvement.

3. Gather feedback from a small group of users.

#### 7. Performance

- \*\*What to Check\*\*:

- Monitor system performance before and after installation.

- Test for any slowdowns or increased resource usage.

- \*\*Steps\*\*:

1. Install the extension in a test environment.

2. Use performance monitoring tools to track system metrics.

3. Document any negative impacts on performance.

#### 8. Security

- \*\*What to Check\*\*:

- Scan for known vulnerabilities.

- Verify the publisher's credentials.

- \*\*Steps\*\*:

1. Use tools like `npm audit` for Node.js-based extensions.

2. Check the extension's publisher on the Marketplace.

3. Review any available security reports.

#### 9. Maintainability

- \*\*What to Check\*\*:

- Verify the extension is actively maintained.

- Check for a history of timely updates and bug fixes.

- \*\*Steps\*\*:

1. Review the "Version History" tab on the Marketplace.

2. Check the extension's repository (if open source) for recent activity.

3. Evaluate the responsiveness of maintainers to issues.

#### 10. Documentation

- \*\*What to Check\*\*:

- Ensure there is sufficient documentation for installation, usage, and troubleshooting.

- \*\*Steps\*\*:

1. Review the documentation available on the Marketplace and the extension's website.

2. Check for tutorials and FAQs.

3. Evaluate the clarity and completeness of the documentation.

#### 11. Community Support

- \*\*What to Check\*\*:

- Look for active community forums or discussion boards.

- Assess the level of community engagement.

- \*\*Steps\*\*:

1. Search for forums or community pages related to the extension.

2. Check the number of stars, forks, and contributors on the extension’s repository.

3. Evaluate user feedback and support availability.

#### 12. License

- \*\*What to Check\*\*:

- Verify the extension's license is suitable for our use.

- \*\*Steps\*\*:

1. Review the license information on the Marketplace.

2. Ensure it complies with our organization’s policies.

3. Document any usage restrictions or obligations.

#### 13. Test Coverage

- \*\*What to Check\*\*:

- Verify the extension has adequate test coverage.

- Identify any known bugs or issues.

- \*\*Steps\*\*:

1. Check for test coverage reports on the extension’s repository.

2. Review the issue tracker for reported bugs.

3. Document the stability and reliability of the extension.

### Evaluation Process

1. \*\*Initial Request\*\*:

- Users submit a request for a new extension through the designated request form or platform.

2. \*\*Preliminary Review\*\*:

- Perform a quick review to filter out any obviously unsuitable extensions.

3. \*\*Detailed Evaluation\*\*:

- Conduct a thorough evaluation using the criteria outlined above for the requested or identified alternative extension.

- Document findings and make a recommendation based on the evaluation.

4. \*\*Decision Making\*\*:

- Based on the evaluation results, decide whether to accept or reject the extension.

- Communicate the decision to the requesting user and provide feedback if the extension is rejected.

5. \*\*Onboarding\*\*:

- If accepted, download the extension from the Microsoft Marketplace.

- Perform any necessary integrations and prepare the extension for deployment.

- Add the extension to the private marketplace and notify users of its availability.

6. \*\*Lifecycle Management\*\*:

- Regularly review and update the extensions in the private marketplace.

- Monitor for any new updates, compatibility issues, or security vulnerabilities.

- Maintain documentation for all lifecycle management steps and processes in Confluence.

---

By following this guide, we ensure that only high-quality, secure, and compatible extensions are onboarded to our private VS Code marketplace. This thorough evaluation process helps maintain a safe and efficient development environment for all our users.

## VS Code Extension Evaluation Guide

### Evaluation Criteria

#### 1. Functionality

* **What to Check**:
  1. Review the extension's description on the [Visual Studio Code Marketplace](https://marketplace.visualstudio.com/vscode" \t "_new).
  2. Test core functionalities in a local VS Code instance.
  3. Gather feedback from users who requested the extension.
* **Steps**:
  1. Go to the extension’s page on the VS Code Marketplace.
  2. Read through the description and feature list.
  3. Install the extension in a test environment.
  4. Verify that all advertised features work as expected.

#### 2. Compatibility

* **What to Check**:
  + Ensure compatibility with our current VS Code version.
  + Verify it works with other commonly used extensions.
* **Steps**:
  1. Check the "Version History" tab on the Marketplace page for compatibility details.
  2. Test the extension with our current VS Code setup.
  3. Document any compatibility issues encountered.

#### 3. Check for Existing Extensions

* **What to Check**:
  + Verify if a similar extension with the same functionality is already available in our private marketplace.
* **Steps**:
  1. Search our private marketplace for extensions with similar functionality.
  2. Compare the requested extension with the existing one to determine if the new one offers any additional benefits.
  3. Decide if the new extension is necessary or if the existing one suffices.

#### 4. Check for Alternative Extensions

* **What to Check**:
  + Look for alternative extensions that offer similar functionality.
  + Compare alternatives based on the same evaluation criteria.
* **Steps**:
  1. Search the [Visual Studio Code Marketplace](https://marketplace.visualstudio.com/vscode" \t "_new) for extensions with similar functionality.
  2. Verify the alternative’s publisher is verified.
  3. Compare support and maintenance details.
  4. Choose the best option based on overall evaluation.

#### 5. Integration Issues

* **What to Check**:
  + Verify the extension integrates well with our existing systems and tools (e.g., Jira, Checkmarx).
  + Test functionalities specific to our environment.
* **Steps**:
  1. Install the extension and perform integrations with relevant systems.
  2. Test key features, such as creating issues in Jira or viewing reports in Checkmarx.
  3. Document any integration issues and assess their impact.

#### 6. Usability

* **What to Check**:
  + Assess the user interface and ease of use.
  + Look for intuitive design and clear instructions.
* **Steps**:
  1. Use the extension and navigate through its features.
  2. Note any usability issues or areas for improvement.
  3. Gather feedback from a small group of users.

#### 7. Performance

* **What to Check**:
  + Monitor system performance before and after installation.
  + Test for any slowdowns or increased resource usage.
* **Steps**:
  1. Install the extension in a test environment.
  2. Use performance monitoring tools to track system metrics.
  3. Document any negative impacts on performance.

#### 8. Security

* **What to Check**:
  + Scan for known vulnerabilities.
  + Verify the publisher's credentials.
* **Steps**:
  1. Use tools like npm audit for Node.js-based extensions.
  2. Check the extension's publisher on the Marketplace.
  3. Review any available security reports.

#### 9. Maintainability

* **What to Check**:
  + Verify the extension is actively maintained.
  + Check for a history of timely updates and bug fixes.
* **Steps**:
  1. Review the "Version History" tab on the Marketplace.
  2. Check the extension's repository (if open source) for recent activity.
  3. Evaluate the responsiveness of maintainers to issues.

#### 10. Documentation

* **What to Check**:
  + Ensure there is sufficient documentation for installation, usage, and troubleshooting.
* **Steps**:
  1. Review the documentation available on the Marketplace and the extension's website.
  2. Check for tutorials and FAQs.
  3. Evaluate the clarity and completeness of the documentation.

#### 11. Community Support

* **What to Check**:
  + Look for active community forums or discussion boards.
  + Assess the level of community engagement.
* **Steps**:
  1. Search for forums or community pages related to the extension.
  2. Check the number of stars, forks, and contributors on the extension’s repository.
  3. Evaluate user feedback and support availability.

#### 12. License

* **What to Check**:
  + Verify the extension's license is suitable for our use.
* **Steps**:
  1. Review the license information on the Marketplace.
  2. Ensure it complies with our organization’s policies.
  3. Document any usage restrictions or obligations.

#### 13. Test Coverage

* **What to Check**:
  + Verify the extension has adequate test coverage.
  + Identify any known bugs or issues.
* **Steps**:
  1. Check for test coverage reports on the extension’s repository.
  2. Review the issue tracker for reported bugs.
  3. Document the stability and reliability of the extension.

### Evaluation Process

**Initial Request**:

* + Users submit a request for a new extension through the designated request form or platform.

**Preliminary Review**:

* + Perform a quick review to filter out any obviously unsuitable extensions.

**Detailed Evaluation**:

* + Conduct a thorough evaluation using the criteria outlined above for the requested or identified alternative extension.
  + Document findings and make a recommendation based on the evaluation.

**Decision Making**:

* + Based on the evaluation results, decide whether to accept or reject the extension.
  + Communicate the decision to the requesting user and provide feedback if the extension is rejected.

**Onboarding**:

* + If accepted, download the extension from the Microsoft Marketplace.
  + Perform any necessary integrations and prepare the extension for deployment.
  + Add the extension to the private marketplace and notify users of its availability.

**Lifecycle Management**:

* + Regularly review and update the extensions in the private marketplace.
  + Monitor for any new updates, compatibility issues, or security vulnerabilities.
  + Maintain documentation for all lifecycle management steps and processes in Confluence.

By following this guide, we ensure that only high-quality, secure, and compatible extensions are onboarded to our private VS Code marketplace. This thorough evaluation process helps maintain a safe and efficient development environment for all our users.