Practical Course:
Contributing to an Open-Source Project
Project Report – Visual Studio Code

Goldstein, Leonard

16.03.2021

1 Introduction

In the context of the Practical Course – Contributing to an Open-Source Project, during the last semester I learned about Free/Libre Open-Source Software (FLOSS). Then I contributed to the Project Code-OSS, the Open-Source project for Microsoft's popular code editor Visual Studio Code (VS Code). In this report, I will outline my contributions and my interaction with the community and team members. I will also describe what I have learned and suggest which parts of the project should be handled differently.

1.1 Project structure

The VS Code editor runs in Electron and is therefore written using web technology, especially TypeScript. Most of the Communication happens through the main repository *microsoft/vscode*, in its issues and pull requests (PRs). Many editor components, such as the editor itself, Monaco, are located in separate repositories. Technically, VS Code refers to the compiled product, which has a proprietary license, and Code-OSS refers to the Open-Source project, although due to the repository name, often the whole project is referred to as VS Code. My preliminary report[3] contains more details about the project structure.

2 Contributions

2.1 In the Main Repository

After the theory part of the course, I started working with the code in the vscode repository on December 1 by compiling it and familiarizing myself with the structure. I then continued working on multiple features and bugs in that repository. On February 10, I also started working on a feature in Eric Amodio's GitLens repository, which contains the code for the VS Code extension of the same name.

My first feature, which I started working on December 13, adds an option to the Editor's settings: VS Code has so-called preview editors, which automatically close when another file is opened - they usually open when a file is clicked once and can be recognized due to their italic tab title font variant. With the November 2020 update, the behavior has changed, so that opening a file directly from the preview editor, for example, by using *go to definition*, would not close the existing editor anymore but convert it to a non-preview editor. In an issue that he opened, the user Sarper Soher requested a feature that would allow him to switch back to the old behavior of always closing preview editors because the new behavior would be counterintuitive to them[16]. In coordination with the assigned team member Benjamin Pasero, I added an option to the application settings that allow users to switch to the old behavior. The PR was merged on December 15[4].

Secondly, I started working on working on a feature regarding the names of diff editors on December 13 as well. VS Code usually only shows the name of the file in a label in the tab, without their path. When multiple files with the same name, but different file paths are opened, the tabs have an additional, differently styled label, which shows the relevant parts of the path. So-called diff editors, which compare two files, show the complete paths, when the file names are equal, which, as the user Shasiru Anuhara explains in their issue about this topic, can cause very long tab names[2]. After discussing multiple approaches, I implemented two solutions, one that keeps the current format but shortens the paths, and one that splits path and file name like for usual editors, requiring two labels per file. I pushed both versions and created a PR for the second version on February 9th, but Pasero noted that he won't have time to review the PR until October[6]. This was the issue that was most work for me, because of the two separate implementations and the necessity to refactor parts of the application for this, resulting in changes in over 400 lines of code across 9 files.

Then, I started to work on some smaller bug issues: VS Code got a new feature that allows tabs to be wrapped across multiple lines and stretched to always fill the available space in January 2021. According to an issue by team member Ladislau Szomoru, in some cases, the tabs flicker on editor resize[17]. I found the origin of the bug; in some special cases, the editor was switching between the behavior of two edge cases for each rendering. Subsequently, I submitted a fix on January 27, Pasero merged the PR on January 28[5].

I also found the cause for a visual bug regarding the display of hover labels in test suites on January 28. I explained what was causing the issue and offered, but the team member Alexandru Dima thanked and declined because he was planning to refactor the respective files anyways[7].

2.2 In the GitLens Repository

Lastly, because of the lack of meaningful, bigger issues and problems with communication that I will be elucidating in a later section, I then decided to work on an issue in the project and VS Code-extension GitLens, which is led by Microsoft Employee Eric Amodio. I implemented the internationalization feature so that the extension can be translated into other languages. Translation files that can be edited with user-friendly tools will be converted and bundled at build time. I started with this issue on February 10th[1], and while I have almost finished the code, I expect that Amodio will request some changes depending on how he plans the translation workflow to work. In this issue, a lot of refactoring was necessary - over 900 strings needed to be moved into a translation file, most of them manually.

3 Communication and other interactions

3.1 General observations

In general, the communication with team members was straight-forward, friendly, and helpful. I personally did not notice that interactions would always take too long: in six out of the nine cases in which a comment of mine directly required an answer from the team, I even got an answer in one day or less. Many practices happened in a helpful manner. In most issues, the responsible team members supported contributions by pointing to relevant code fragments, which was, especially in the beginning, very helpful.

Already at the time of the preliminary report, I noticed that it is common practice that development team members do smaller refactorings to external contributor's PRs without asking them previously to do so. That takes some getting used to and for me in my PRs sometimes irrationally felt like a negative comment on my code's quality, but ultimately I understand that this is not a bad idea and decreases the time a PR needs.

While communication in basic cases worked well, I would have wished that there were other communication platforms than the GitHub Issues. Issues require formal texts and don't leave much space for asking and answering technical questions, because they are usually kept in a way that everyone can understand them. There is already a *VS Code Development Community* Slack workspace[15] which contains channels for this specific use case but it is not linked at any prominent place in the repository and thus less active than it could be. Also, most team members are barely active on the Slack workspace.

Contributing to VS Code gave me a deeper understanding of parts of the application, and I was able to use that to contribute to discussions. For example, I answered an issue by wesinator, a community member who requested to change back the default behavior for preview tabs[20]. Because I had implemented a feature that is connected to his request and understood the background of the change, I was able to explain why in my opinion, the change should not be reverted.

3.2 Missing transparency regarding community involvement

Despite some positive interaction experiences, I experienced slow response times of up to 23 days and vague answers in some cases. A good example is the diff editor name feature[2]: Initially, Pasero was highly available and supported the discussion about possible implementations. He answered messages by me and another community member in one day. When I started working on the feature and drafted two possible implementations, he did not give feedback on which version would be better for the project that would lead to an immediate result, so I implemented both. After full implementations, the answers took longer: I initially pushed the initial versions on January 18th, but only got an answer after pinging again on February 9th. Ultimately, he signaled that he won't be available to review the PR before October when the next issue grooming iteration would happen[6]. I am dissatisfied with that decision due to multiple reasons. First, the respective files would likely change a lot in that time frame, so more work on resolving merge conflicts will be necessary. Some changes to these files already happened while developing the feature, and even I changed one of the files in the fix for the flickering tabs bug[5]. Second, this signaled me that this feature has a low priority at a point in time that was too late. In my comments, I made it obvious that the implementation would change large portions of how the tab system works. If I would have been told that the probability of a successful PR in near future is low in that case, I would have allocated my time to more important issues and wouldn't have implemented two solutions.

This example illustrates two problems in the field of communication. First, as soon as communication and reviews get lengthier, answers take more time. I noticed that answers and reviews for less code-intense issues happen more quickly, for example, the flickering bug issue was reviewed and merged in less than a day. This should not be the case in my opinion. I understand that lengthy reviews take more time, but a quick comment about whether it will be done and how long that will take is written fast and is sufficient.

Michael Plainer, another participant in the practical course and contributor to VS Code, made similar experiences regarding the lack of useful responses in reasonable time and suggested updating the project documentation for more clarity in an issue on February 15[12]. Because I saw many parallels to my concerns, I agreed and added my perspective and possible ways to solve them. The response of team member Kai Maetzel to this issue was diplomatic but evasive. Instead of being open to addressing the core issue, he explained general reasons for unavailability and asked for a specific case in which this problem happened. Until now, we haven't got a reaction to our last comments or an explanation on how a contributor should react when team members don't answer.

Secondly and most importantly, the team does not communicate clearly in which cases and to what extent they want to involve the Open-Source development community.

Code-OSS is an Open-Source project, but not every Open-Source project is developed in a similarly open fashion: In my understanding, open development requires an Open-Source project but also incorporates the possibility to voice feedback and opinions openly, to get insight into future plans for the project, to allow contributions from external parties and to decentralize decision-making. The more open a project is developed, the fewer developers and customers could be distinguished.

VS Code does not meet all these criteria: The project is Open-Source and feedback and opinions are being received. The results of the team's plans are available publicly in form of iteration plans and yearly roadmaps, but the reasoning behind decisions can't be comprehended completely because the internal communication that leads to them is not made public. Community contributions exist, but as the previous example illustrates, processes such as reviews are not geared towards encouraging and incorporating bigger contributions, although they are officially possible. Decision-making, such as planning how the project should evolve, is not opened to the public. There are some mechanisms that create the appearance that decisions are not centralized and users can decide which features get implemented, but in reality, the chance that they influence the decision is very small. For example, there is the backlog candidate mechanism[9]: Some community issues are marked as backlog candidates. When these get 20 thumbs-up reactions (upvotes) in 60 days, they will be moved into the backlog, otherwise, they won't be implemented. This sounds like the community can decide, but practically, issues almost always fail to succeed because there are apparently not enough community members that browse open issues. At the time of writing, there are 1929 issues that were at some point backend candidates and used the upvote mechanism[10], of which only 16 of the remaining issues (about 0.8% of all upvote mechanism issues) are linked to a merged PR [11]. The team must be aware of this because they often mark issues as backlog candidates and almost never merge linked PRs, but apparently they still this mechanism. The user scscgit complained about the arbitrarity of the 20 upvote metric in an issue in August 2020 but didn't receive an answer from the team. [14] This is notable, especially because it happened before the last issue grooming iteration, where overlooked issues would be revisited.

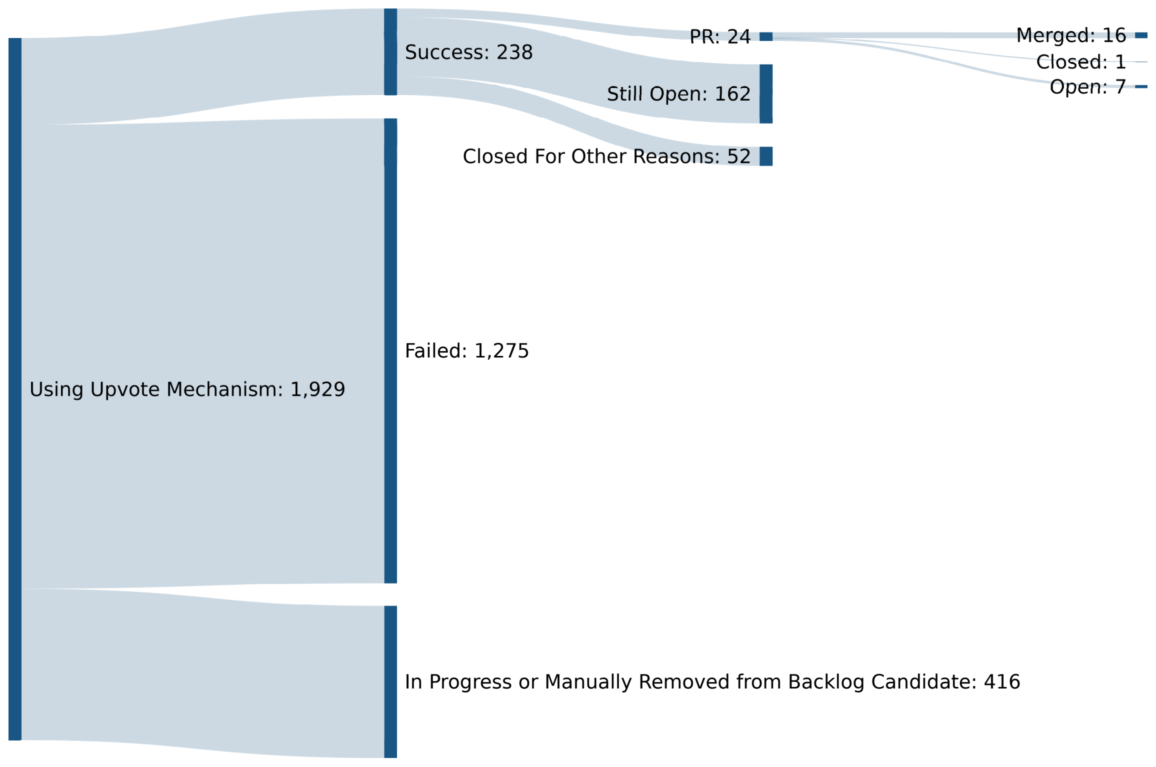

Figure 1: (Lack of) success of upvote mechanism powered backlog candidate issues

I conclude that VS Code is Open-Source, but is not developed openly. This is in itself not a problem, there are valid reasons for that, such as decreased interdependence with the community, more control, and less community management effort. However, a big problem is that the details of openness of the project are not communicated precisely to avoid confusion. Knowing where contributions, both in form of code and suggestions, are wanted saves time not only for contributors but also for team members, who currently have to spend a whole month yearly to clean the repository from community issues and PRs.

These observations show that the project is development team-driven and not community-driven and the team knows that, but for some reason is not willing to publicly explain the extent of that. I can only assume that this is because leading individuals want to keep as much control over the project as possible without appearing too possessive to third parties. Although it does not match the ideas behind Open-Source, I can understand that they might want to stay able to control the project, but being dishonest is harmful to the project.

4 Suggestions for the project

4.1 Updating the wiki

In his issue, Plainer has argued that the documentation should be updated to include the process that one can follow when team members don't respond[12]. While I agree, I think that alone and the internal discussion that this might or might not cause are not enough to tackle all problems. I described previously that slow answers not dependent on external contributors dependent on external contributors might be connected to different priorities. This leads me to believe that adding more details in regards to responsibilities and aims would be helpful: Being able to read on who in the team works on what would help finding the correct contact person and would give more context to the role of a community contributor. Also, the documentation should be extended in the means of what kind of contributions are appreciated most, so both sides can be useful and don't waste time.

4.2 Opening up communication

As I mentioned in the preliminary report[3], there is a separate internal Slack workspace for team communication. While I understand that team members want to have a space to communicate efficiently with other team members, I think that not doing so publicly seems too secretive for an Open-Source project. The team could instead use dedicated channels on the workspace that is already open to the community, which they could make read-only for non team members if they deem that to be necessary.

4.3 Development structure

Apart from the discussion about transparency in communication, I think that the team could profit from a more open development approach. In his talk for the practical course, Stephan Kulla[8], who works on the Serlo project, explained that it takes a lot of time and frustration tolerance to introduce new people to the project, but is extremely helpful in the long run. In the case of VS Code, the situation is different: Communication on technicalities takes time for team members, but there are already many people who don't need any more motivation and would like to contribute. What stuck with me most from his talk is that he described how he plans that all the motivational and technical work he puts into other developers would pay back someday. Encouraging more meaningful community contributions by communicating better would certainly still be a time investment, but eventually would result in more reliable and frequent community contributors. Empowering the community would not likely result in the loss of control: Important decisions would still be made by the team. Successful forks of the project are, in my opinion, improbable, due to the size of the project it would still be hard for the community to completely continue the project on its own in a similar quality, let alone to compete with the original project. Development speed might decrease due to more necessity to review additions. This could be avoided by employing more team members, possibly even from the community. Some could concentrate on development as they do right now, others could focus on community coordination work. However, I know the latter proposed changes are improbable because they would require a lot of restructuring.

4.4 Community engagement

Even though I have argued that the VS Code team doesn't have to motivate contributors, but just give them space and support for the coding part, it would be helpful to connect and strengthen a network between contributors. There are other ways to involve the development community than enabling them to make bigger contributions. VS Code already has many approaches to community building measures, such as so-called release parties[13] and the VS Code Day[18], which are online streams that report on the development progress. However, these are geared mostly towards the general user community and the insiders' community, who test changes before the release. Interest in more continuous development from community members would increase if these events would get more technical and interactive. When I participated in the VS Code Days, there were few things that I learned, because I already knew about the current development efforts due to GitHub issues, so for me as a member of the FLOSS developer community, it would be of great interest to instead get more context behind decisions. I also didn't feel involved, because the event, apart from the fact that some questions were answered from time to time, didn't feel interactive. Structuring it more like a conversation with community members could increase engagement because members would feel more valued and listened to.

4.5 Extending technical documentation across repositories

Orienting myself inside the codebase and understanding which part does what took me most of the time during development. Many parts of the code are written in a way that may be easy to understand to someone who knows the project, but not for someone new. For example, I came across a class `Disposable`, which many other classes extended. Even though I am familiar with the dispose pattern, I had a hard time understanding what exactly happens during disposal. It took me a long time to understand it because the code behind such general classes is relatively abstract. Now, after having used it a few times, I would probably not need documentation, but it would certainly be useful for new developers to have at least a short jsdoc description for every class and interface, which is not always the case.

The situation is, in my experience, even worse in other repositories of the project. For the GitLens internationalization feature, I am using some tools that were created specifically by Microsoft for VS Code plugins, such as *vscode-nls-dev*[19]. It seems like the user is just supposed to copy the example and built upon that. The example, however, doesn't cover all functions that are available to the user, for example, I almost missed the fact that the tool contains a webpack loader, which I could make use of for GitLens. To understand how exactly it has to be used, I had to browse through barely documented code, which should not be the case in a project that should be used as a black-box-like helper tool. In this case, at least the README file should be updated. I am thinking about doing that after the internationalization issue is fixed.

5 Personal Learnings

5.1 Personal understanding of Open-Source projects

5.1.1 Communication

Some of my biggest learnings were in the field of communication. Before contributing, I expected that being friendly is the most important aspect. I now know that being open, transparent, and honest in communication is more important. This is of course important to follow as a team member, but also as a contributor. For example, one should always communicate availabilities. When I worked on the diff tab label issue, I initially was not sure if I was to work during the holidays. Because of that, I didn't write that I'm unavailable, even though I didn't work on the issue for some time. This leaded another user to ask if I was still working on the issue[2]. Because I responded to that quickly, I don't believe that I did any harm, but still, it would have been better if I would have communicated that I might be slower in that time period to save others from confusion. When structured correctly, more information is better than less.

I also learned that I am partially responsible for the clarity of the responses I get. When I don't explain what exactly I am planing, I cannot expect to get an answer on whether that is the right thing to do. Asking direct questions helps to get direct answers.

An important aspect is to be being aware of and clear about the message's recipients. In one message in an issue, I mentioned two persons via GitHub handles and expected an answer from both of them[2]. One of them answered but the other one did not; possibly because he thought that the answer of the first person was sufficient to me. When confronting multiple people with multiple questions, it is better to split the questions into multiple distinct messages.

Some of the issues that I worked on, such as the diff tab label and the internationalization issue, resulted in a lot of changes. I don't think that in those cases, it would have been possible to keep the changes smaller, but I noticed that this makes it harder to get good feedback and increases the risk of not being what the other party expected it to be. When the issue and project structures allow doing that, it is useful to split issues into smaller issues and do multiple PRs.

5.1.2 FLOSS projects

All previous learnings apply to software projects with written communication in general, not just FLOSS. But I learned that there are some differences to normal projects:

A thought that is central to my understanding of Open-Source contributions is that one gives something, usually, time, away for free, so that other people can profit from it. An Open-Source developer should be aware of the fact that the community and maintainers are not obligated to use one's contribution in the way that one expected it. The flickering tab bug that I fixed only changed few lines of code. Those lines of code were, for style reasons, refactored, so that no line of my commit was left as I committed it originally[5]. The main work in this issue was to find the bug at all, as team members unsuccessfully had already spent multiple days to find the bug's origin before I found it. Although I didn't say so in the PR, I was slightly frustrated that the end results in the code were not representative of that fact. To prevent that, I should have been aware of my motifs and should have understood that there is no such thing as ownership of one's contribution. Open-Source contributions most of the time, and in that example too, result in appreciation, but should not be made expecting it. In the end, the only thing that counts is that one was helpful.

FLOSS can come in many forms. The one that comes to mind when thinking about FLOSS for me is a community-driven project form that depends on the individual contributors because that seems to be what fits best to the philosophical ideas behind FLOSS. This is, especially with corporate projects like VS Code, not necessarily the case. There are other motifs to sharing something openly than just goodwill, and in the same way, there are other ways to develop FLOSS than through a community. When coming to a new project, one should not assume a specific role for him-/herself.

5.2 Changes in my personal relation to VS Code

Since my initial research for the preliminary report, my relation to the project has changed. I originally concluded that VS Code would, despite its size, be a good place to start getting into Open-Source Development. After having taken part in contribution processes, I have changed my opinion about that: VS Code does have some good issues to get into the project, but many aspects that one would have in other project are not possible in VS Code, such as taking responsibility and working on an important part of the project. That does not mean that I would not recommend or not like to contribute to VS Code in the future: I am still a fan of the product and will at least want to continue with the open diff editor label PR in October. I would, however, not start working on larger issues in that repository anymore, due to the risk of huge delays and insecurities in the review process.

Even though in the preliminary report I stated that VS Code consists of more parts than the vscode repository, most contributors only look out for issues in the main repository, as many other repositories don't seem to get similarly much attention. This results in a situation where there is more demand than supply in help-wanted issues - in the beginning, I refreshed the issues list multiple times per hour to make sure to be the first one who offers help. That is suboptimal because someone else would have been able to do the same work, so one could say that my contributions have a lower worth. Also, in other projects surrounding VS Code, help is, although not being as visible as in the main repository, needed as well. This is why I think of VS Code not as the vscode repository anymore, but as a product that consists of many repositories, and started working on GitLens, which is technically not even a core component of VS Code, but an extension by a Microsoft developer.

5.2.1 Ideas from VS Code that should be adopted in other FLOSS

If I was leading an Open-Source project of a similar scale, I would, despite the criticism in this report, copy many adopt ideas from VS Code. First and foremost, I would implement a similar sprint system: monthly iterations seem to be a good fit for Open-Source projects of that size. It doesn't in itself puts on too much pressure to include ideas outside of what is planned, but puts on just enough structure to control progress, which to me seems to be slower in other big Open-Source projects.

I also appreciated the modularity of the project and that many parts are located in different repositories, so they can be developed rather independently. This enables other projects to profit from VS Code because they can build upon technology that was created as a byproduct of VS Code. I think this modularity could even be increased in the content that is now in the main repository and their organization could be a bit more split.

6 Conclusion

After taking part in the development of VS Code for four months, I was able to successfully contribute to the project. Although I had the chance to make some code contributions, I partially felt held back by the way the development for this project is structured. The biggest problems that occurred were due to the lack of timely and meaningful feedback and sometimes even the complete absence of explanation. I and other contributors found various approaches that the team behind VS Code could apply to work against the problems. However, based on reactions to previous issues, it is improbable that the team is willing to change something.

Despite knowing much more about FLOSS after the practical course from the theory side and after my time developing from the practical side, I would not say that I now fully understand FLOSS development: My biggest realization was that there is not one go-to way to develop FLOSS - and as long as all aspects are communicated clearly, different ways can lead to a successful project.

References

[1] Eric Amodio. Issue: Help wanted: Add internationalization (i18n) support to gitlens. `https://github.com/eamodio/vscode-gitlens/issues/577`, December 2018. Accessed: 2021-03-15.

[2] Shasiru Anuhara. Issue: Allow to shorten the diff editor title. `https://github.com/microsoft/vscode/issues/110694`, November 2020. Accessed: 2021-03-15.

[3] Leonard Goldstein. Lab course contributing to an open source project - preliminary report: Visual studio code. `https://www21.in.tum.de/teaching/osp/WS20/assets/pr-goldstein-vscode.pdf`, December 2020. Accessed: 2021-03-15.

[4] Leonard Goldstein. Pull request: add option for preview navigation behavior. `https://github.com/microsoft/vscode/pull/112389`, December 2020. Accessed: 2021-03-15.

[5] Leonard Goldstein. Pull request: fix 115050 flickering tabs when wrapped. `https://github.com/microsoft/vscode/pull/115273`, January 2021.

[6] Leonard Goldstein. Pull request: Multiple tab labels for diff editors. `https://github.com/microsoft/vscode/pull/116178`, February 2021. Accessed: 2021-03-15.

[7] Daniel Imms. Issue: Test hover lacks padding. `https://github.com/microsoft/vscode/issues/114925`, January 2021. Accessed: 2021-03-15.

[8] Stephan Kulla. Motivation in open source projects. Personal Communication, `https://www21.in.tum.de/teaching/osp/WS20/index.html#stephan-kulla-motivation-in-open-source-projects`, December 2020.

[9] Kai Maetzel et al. Issues triaging. `https://github.com/microsoft/vscode/wiki/Issues-Triaging`, July 2020. Accessed: 2021-03-15.

[10] Issues - microsoft/vscode - is:issue "this feature request is now a candidate for our backlog. the community has 60 days to upvote the issue". `https://github.com/microsoft/vscode/issues?q=is%3Aissue+%22This+feature+request+is+now+a+candidate+for+our+backlog.+The+community+has+60+days+to+upvote+the+issue%22+`, March 2021. Accessed: 2021-03-15.

[11] Issues - microsoft/vscode - is:issue ”this feature request received a sufficient number of community upvotes and we moved it to our backlog” linked:pr. https://github.com/microsoft/vscode/issues?q=is%3Aissue+%22This+feature+request+received+a+sufficient+number+of+community+upvotes+and+we+moved+it+to+our+backlog%22+linked%3Apr+, March 2021. Accessed: 2021-03-15.

[12] Michael Plainer. Issue: Community question regarding issue handling. https://github.com/microsoft/vscode/issues/116726, February 2021. Accessed: 2021-03-15.

[13] Vs code release live stream. https://code.visualstudio.com/livestream. Accessed: 2021-03-15.

[14] scscgit. [meta] feature request that’s a candidate for backlog has no way to gain visibility and uses arbitrary metric. https://github.com/microsoft/vscode/issues/103784, August 2020. Accessed: 2021-03-15.

[15] Visual studio developer community. slack workspace. https://aka.ms/vscode-dev-community. Accessed: 2021-03-15.

[16] Sarper Soher. Issue: Ability to switch to previous behavior on 1.52.0 ’preview editor improvements’. https://github.com/microsoft/vscode/issues/112348, December 2020. Accessed: 2021-03-15.

[17] Ladislau Szomoru. Issue: Tab wrap - flicker when there is not enough room for wrapping tabs to display. https://github.com/microsoft/vscode/issues/115050, January 2021. Accessed: 2021-03-15.

[18] Vs code day 2021 live event. https://code.visualstudio.com/vscode-day. Accessed: 2021-03-15.

[19] microsoft/vscode-nls-dev. https://github.com/microsoft/vscode-nls-dev. Accessed: 2021-03-15.

[20] wesinator. Issue: Change workbench.editor.enablepreview default to false. https://github.com/microsoft/vscode/issues/115231, January 2021. Accessed: 2021-03-15.