RFC 215 APPROVED: Improving User Feedback Channels

Editor: qkeast@sourcegraph.com
Status: Review
Requested reviewers (please review by EOD 2020-08-28): Christina, Pooja, Rob, Alicja, Quinn, Bunny, Felix, Aida, Garo, Julia
Approvals: Eric, Dan

Background

We're collecting feedback from our users from a variety of sources today. Some of these sources include:

- NPS survey, shown as a pop-over within Sourcegraph after three days of use, then every 30 days after that.
- Tweets and emails.
- Github issues.
- Sales feedback.
- Browser extension uninstall feedback.

The NPS survey is one of the only in-product sources of user feedback. This is working well for providing us with open-ended qualitative feedback—particularly from customer instances.

However, users can only provide this feedback when the NPS pop-up is displayed, which isn't a good way to get in-context feedback when it matters most. As well, some of our users find the pop-ups intrusive.

The challenge with NPS

NPS isn't a great metric for measuring the user experience, for a few reasons:

- It focuses on predicted behaviour, rather than immediate experience or actual behaviour. People are generally bad at predicting their future behaviour.
- The way it's measured doesn't really reflect the results of our efforts to improve the experience of using Sourcegraph. If six people respond with "0" one month, and the same six people respond with "6" the next month, we've clearly made a collective improvement. But as far as NPS is concerned, our score started at -100 and remained at -100.
- It results in a single number that can't be contextualized without specific open-ended feedback, which makes it difficult to use as an actionable metric in our work.

While it isn't necessarily "harmful" to use NPS as a metric (and in fact, it offers a lot of value through the open-ended feedback users provide along with their ratings—which is our chief benefit today), there are

other ways to collect quantitative and qualitative feedback from our users that are more actively useful in our work.

With these challenges in mind, how can we improve how we collect user feedback?

How do we want to use user feedback?

- As a "key experience indicator," particularly for "happiness" outcomes, so that we can evaluate the impact of our work on the user experience.
- As a qualitative data source that we can use to better understand our users and apply their language and thoughts to our work.
- As a reflection of pain points in our product, in the context in which they occur.
- As a signal of engagement with product features.
- For specific feedback in response to new ideas or features.
- As a trigger for customer outreach.

Principles for useful feedback

- **Affective** – Capture "happiness" as a reflection of the experience.
- **Contextual** – Capture and report feedback in relation to specific product features or resources.
- **Meaningful** – Tied to specific user outcomes in context (e.g. "This was **helpful**").
- **Reflection over prediction** – Prioritize feedback about what users **have done**, not what they **predict** they will do.

Proposal

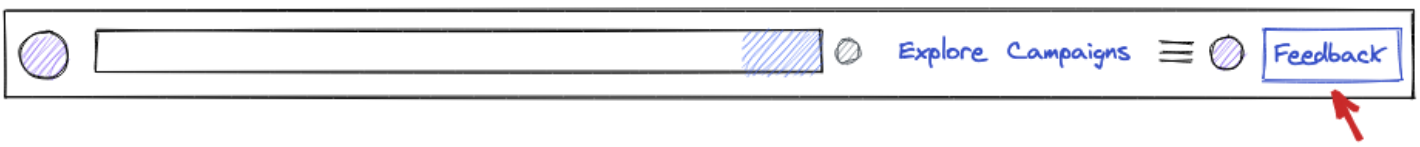
There's a few ways we can very quickly start collecting better user feedback within the current user journey, and a few things we can plan to explore in the future.

Short term actions

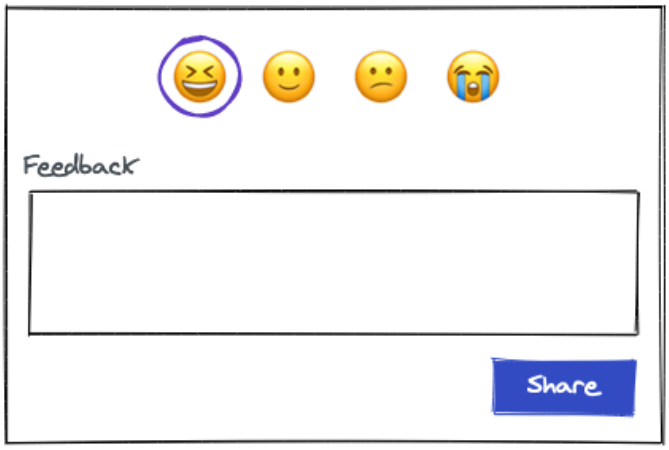
1. Collect ongoing contextual feedback in Sourcegraph

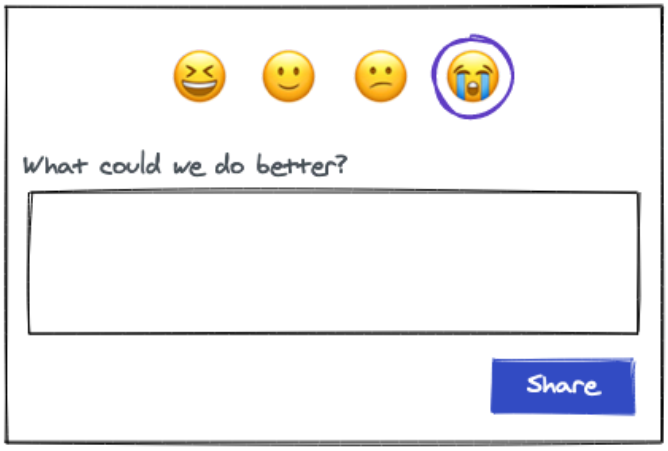
- Introduce a new, "omnipresent" feedback point in Sourcegraph Cloud, Server, and in the docs.
- Collect open-ended responses along with affective feedback (unhappy, happy, etc.), and attach contextual data to the responses, including the current view.

This would be exposed in an unobtrusive way in the Sourcegraph UI (consider these visualizations as almost a breadboard), and can be triggered by the user at any time.



This would collect two data points from the user directly: the “happiness” response, and open-ended feedback. Other data we could save in the stored response includes the user’s email/username (when appropriate, e.g. on Cloud or when Server users opt-in to sharing their contact info with their feedback), the user’s organization (if appropriate) and the current URL (so that we can contextualize the feedback).





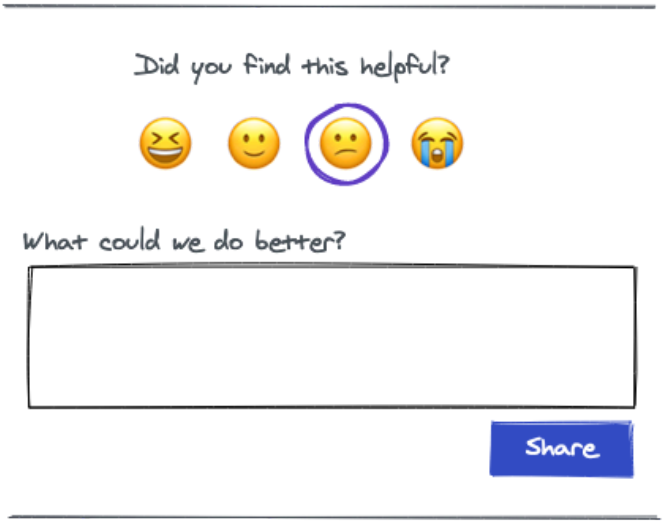
GitHub Issue: https://github.com/sourcegraph/sourcegraph/issues/13667

2. Collect specific contextual feedback in Documentation

- Introduce a feedback point in the Sourcegraph documentation.
- Similar to the omnipresent feedback in Sourcegraph itself, but contextualized to the documentation.

This would be exposed at the end of every documentation page.

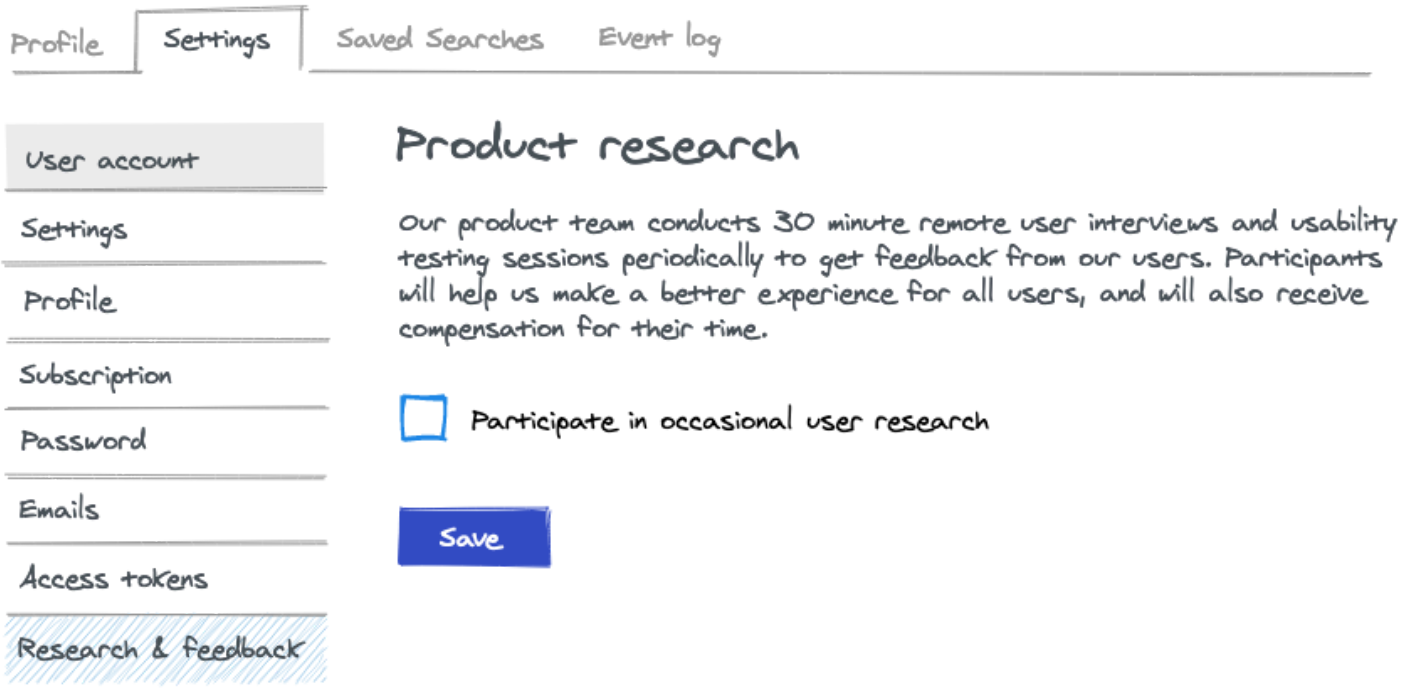
Since this is more contextual than the omnipresent feedback, we can be more purposeful with collecting open-ended feedback. When the user responds to the initial question (Did you find this helpful?), we can save that response immediately. If their response is negative, we follow up with a request for open-ended feedback.



Initial design in Figma:
https://www.figma.com/file/9FprSCL6roIZotcWMvJZuE/Improving-user-feedback-channels?node-id=300%3A4617

3. Create a pool of research participants for ongoing outreach and activities

- Introduce a “user research opt-in” to user settings.
- Create a ready pool of research participants that we can reach out to for conversations and user testing, simplifying the recruitment process.
- If we want, we can also use this to create a semi-automated pipeline of user interview participants. Every week, we can invite selected participants to self-schedule for an open-ended conversation about their experience with Sourcegraph.



Note that this is also an opportunity to build goodwill for the Sourcegraph brand: instead of offering compensation directly to participants, we can let them choose a nonprofit organization (potentially from a preselected list) to receive a donation.

GitHub Issue: https://github.com/sourcegraph/sourcegraph/issues/13668

Later term actions

Since we can't do the kitchen sink all at once, there's a few longer-term things we can also plan to do in the future to improve how we collect user feedback.

1. Tie qualitative feedback to stages in the user journey

- Based on the outcomes of the customer journey mapping project, identify specific stages of the customer journey and use these stages to tag and organize feedback in ProductBoard (and/or other sources).

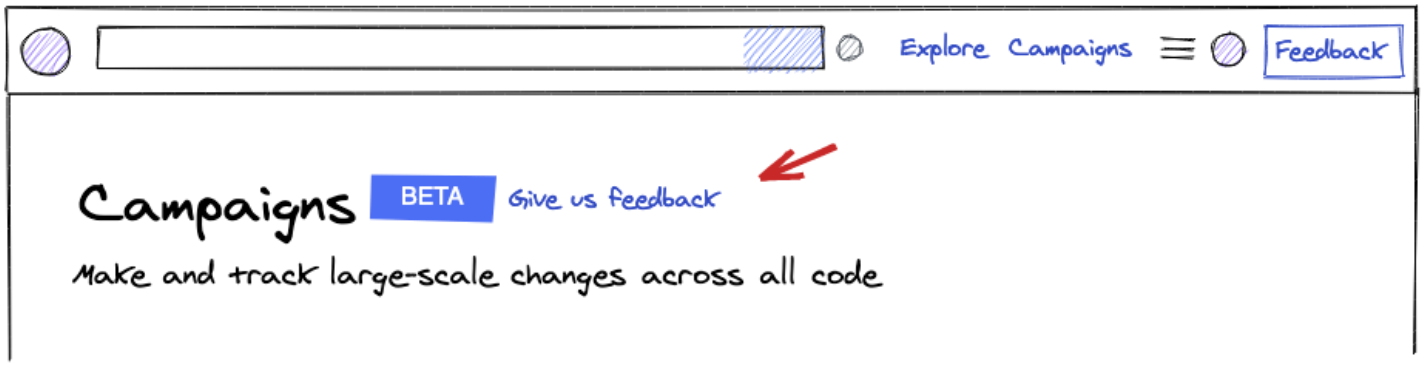
2. Collect specific contextual feedback in Sourcegraph and other touchpoints

- Introduce feedback points at key moments in specific user flows (e.g. after completing a Campaign), touchpoints (e.g. in "search notifications" emails), and actions (e.g. feedback for code intel results).

3. Create triggers for more detailed feedback

- For product initiatives where we want more specific feedback, introduce triggers that can lead to more extensive surveys.

For example (drawing from GitHub's approach):



We might also systemize emails in response to specific actions (or failures to complete funnels), similar to Rob's recommendation to prompt re-engagement with users who fall out of the installation funnel.

4. Create triggers to request access to ideas

- For new product ideas where we want to evaluate demand, add "future features" to the product where they would actually be found if they existed. (While this is predicted behaviour, it's triggered by real action on the user's part, so it can be valuable feedback.
- From @beyang on Slack.

5. Revisit how we expose "dev happiness" as a value proposition in Sourcegraph Server for site admins

- Slack thread for reference: https://sourcegraph.slack.com/archives/CN4FC7XT4/p1597415563370700

Okay, so now we have the feedback, what do we do with it?

Feedback would likely be stored in HubSpot as well as shared to other tools like Slack and Looker.

Since we would use the same format for all affective responses (essentially "Happy, Okay, Meh, Unhappy"), we can use user happiness as a consistent indicator of the experience. As an example, in Looker, we can create a dashboard that captures:

- The percentage of "happy users:"
    - Overall
    - Overall for Sourcegraph
    - Overall for documentation
    - Specific to features, based on the URLs attached to the feedback

Then, we can use this to identify pain points as well as use as a key experience metric for product initiatives and improvements. **This metric would ~~ultimately replace NPS~~ become the first quantitative metric that we use as a measure of the quality of experience.** We won't remove the NPS survey in the short term before validating the quality and quantity of feedback through alternate sources.

Definition of success

We'll know we're succeeding if:

- We're collecting more feedback from users than before.
- We use happiness metrics as project metrics and feel confident that they are actionable and meaningful to our work.
- We have a set of users who opt-in and are willing to participate in user research sessions.

Potential side effect: more negative than positive feedback?

We should also pay attention to whether feedback tends to be mostly negative rather than positive—since it may be that people are more inclined to provide feedback if they have a poor experience, and may only provide positive feedback when things are going well if their feedback is explicitly requested.

However, if we explicitly request feedback, we're adding a trigger that doesn't necessarily collect feedback in context, which kind of goes against our goals.

Our first implementation can give us some insights around how people provide feedback. If we discover we're getting much less positive feedback, then we can use a starting point to iterate on how we request feedback.