Redesigning Data Provision at Slack: The Mentions Tab

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Stephen Dolan

Spring, 2022

Technical Project Team Members

Stephen Dolan

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Rider Foley, Department of Engineering and Society
Rosanne Vrugtman, Department of Computer Science

Abstract
Slack, the corporate messaging application, decided to offer a more updated and intuitive version of the "mentions" tab in the iOS app by showing any updates to the screen without a manual refresh. I achieved this by changing the implementation from waiting for the next API call to a 2-way connection that streamed any new mention events. The solution involved two parts. First, I added the streaming of associated Message models by using updates to existing messages to change the corresponding existing mentions. Second, I used the existing 2-way connection for all messages to create a stream of new mentions not existent in the last API call. The main result was that I successfully implemented this feature with the help of my team and modernized data provision for the mentions tab in Slack's iOS app. This feature is currently being rolled out internally and will be available on the App Store version of Slack within the semester. In the future, to fully synchronize message and mention data, we could include all necessary activity data in message WebSocket events so that an API call is rarely needed. This implementation of using an existing 2-way connection to stream updates to a related database without creating a new 2-way connection could also be applied to other software companies looking to sync data from multiple streams.

1. Introduction and Background

Millions of users rely on Slack for extremely important parts of their lives such as work and hobbies. Ensuring that messaging data on Slack is accurate and updated is essential to these users getting the information they need for their daily lives. Beyond that, Slack has reshaped the way that people can interact with their work and hobbies through the expansion of remote work. Allowing for complex communication via messages, reactions, files, video calls, and document sharing allows work to be more accessible and more diverse in terms of location (which has racial diversity implications as well). As an industry leading messaging app like Slack improves its software, remote work becomes more feasible and more companies will implement remote work policies, leading to a net positive effect on the world.

2. Related works

Stoeckli, et al. (2018) discuss how Slack creates value in remote work within enterprise use. Specifically, it discusses this impact through the case study of chatbots. This is relevant as it helps show the impact of Slack's iOS app.

Richette, et al (2020) discuss the impact of remote work on accessibility (specifically in terms of physical and mental disabilities), both the negatives of an abrupt change but also the positives that remote work offers in the long term. This is relevant as it helps show the impact of Slack's iOS app.
Federici et al. (1970) discuss remote work's effect on accessibility specifically in terms of how accessible it is to different types of people and families. This is relevant as it helps show the impact of Slack's iOS app.

Stray et al. (2020) introduce a case study of how Slack is used in the tech industry and whether its impact is largely positive or negative. They also discuss differences in impact between remote and in-person users of Slack. This is relevant as it helps show the impact of Slack's iOS app.

3. Project Design

My project focused on redesigning data provision in the mentions tab of Slack's iOS app to be more synchronous and up-to-date.

3.1. Review of System Architecture

My project focused on the mentions tab in Slack's iOS app, referred to internally as the "Activities" tab. This tab displays all messages that mention the current user and any messages sent *by* the current user that other users have reacted to. The primary data object used in this case is the *Activity* object that represents one *mention* or *reaction* in the list of activities within the tab.

Currently, data is delivered to iOS devices in two ways: API (application program interface) calls and WebSocket events. API calls (within the context of this project) consist of the iOS device making a network request to a server which returns a list of *Activities*. WebSocket is a framework for two-way communication between an iOS device and a server. Unlike API calls, once a WebSocket connection is established, user devices do not have to make requests to receive data. This allows instant updates to data on the server to flow to end devices in real time through "WebSocket events".

3.2. Project Overview

In the current Slack iOS app, activity data is solely refreshed from an API call triggered by either manual pull down or a 60 second timer. In between these API calls activity data is not up to date, often leading to desynchronized data between the activities tab and other parts of the app. My task was to use the WebSocket framework to create true streaming of all activity changes, thus modernizing the mentions tab.

3.3. Existing Activities

The first part of my project concerned adding streaming of any changes to *existing* activities in the tab. This was accomplished by using updates to the *Message* data model. Every *Activity* model has an associated *Message* model because every mention is also a message, and every reaction is a reaction to a message. There are no WebSocket events for *Activity* changes. However, there are existing WebSocket events for *Message* model changes. I decided to use these WebSocket events to update *Activity* data since each *Activity* is associated with a *Message*. I did this by, first, converting each *Activity* ID to an associated *Message* ID, and then listening to any updates to those associated *Message* models. Upon relevant *Message* changes, associated *Activities* are now updated.

3.4. New Activities

Unfortunately, listening to associated WebSocket *Message* events as described in **3.3** still does not allow streaming of new or deleted activities. This is because *Activities* are currently streamed using the list of IDs from the last API. Even if we create or delete a new activity, its ID will still be present until the next API call. Thus, the second part of my project was to allow streaming of new and deleted activities.

Again, I used existing *Message* WebSocket events for new or deleted *Messages*. However, instead of updating existing *Activities*, for these new events we create new (or delete old) *Activity* models using relevant message data in the WebSocket event handler.

However, because the current *Activities* stream only listens for updates to models in the initial set of IDs, the final step needed was to redesign the *Activities* stream itself to include any new or deleted *Activities*. This was achieved by creating an additional, separate stream for new activities, and then combining this stream with the existing stream into one stream of all activities from all sources for the UI. This was done by creating a "scratchpad" stream of all activities with timestamps after the last activity from an API (in

other words, all activities created from WebSocket events). This stream was then combined with the stream of activities whose IDs were present in the last API call. Upon each API call, the scratchpad stream of WebSocket-created activities is wiped because the new API call's data has all the activities that were previously in the scratchpad stream.

3.5. Challenges

The biggest challenge in this project came in the final step: redesigning the data stream. There were many problems in synchronizing data from multiple sources (e.g. from both API calls and WebSocket events). For example, often the next API call after a new *Message* WebSocket event would add a duplicate *Activity* to the stream before the WebSocket event stream was reset. Other times, if the most recent *Activity* was deleted in a WebSocket event, the most recent timestamp would change and thus delete valid WebSocket-created *Activities*. To solve this, I had to draw out all the possible orders in which various events could occur, and have built-in checks take place in the appropriate data refresh points to avoid synchronization errors.

4. Results

I successfully implemented this project with the help of my team and modernized data provision for the mentions tab in Slack's iOS app. This feature is currently being rolled out internally and will be available on the App Store version of Slack within the semester. This will directly impact user trust in Slack as it makes activity data more up-to-date and makes the app simpler as less user manual refreshes are needed. Even small points of confusion or inaccurate data can cause companies to lose trust in mobile apps and therefore be less likely to approve remote work. Specifically, innovations in mobile software allow even greater accessibility within remote work. Many individuals cannot afford computers or have blue-collar jobs which require them to move around throughout the day. Mobile messaging apps like Slack's iOS app increase the number of people who can work remotely and therefore contribute to the accessibility, diversity, and social life benefits remote work has added to society. By extension, an outcome of this project was greater trust in remote work, thus increasing accessibility in the workplace.

5. Conclusion

The expansion of remote work has enabled people with both physical and mental disabilities to work more comfortably and even work at jobs that were otherwise not possible. One crucial element of this emergence of accessibility in the corporate world is the trust of organizations in remote work. Possibly the biggest factor for corporate trust in remote software applications is data consistency and availability. To increase trust in remote work, I redesigned the mentions tab in Slack's iOS app to be more synchronized and up-to-date through the two-way connection framework WebSocket. This implementation and its impact are being currently rolled out internally and will be in the app store version soon.

6. Future Work

To fully synchronize message and mention data, we could include all necessary activity data in message WebSocket events so that an API call is rarely needed. In other words, instead of having a scratchpad stream that combines with a "source of truth" stream, we could just have a single stream of *Activities* that is added to by WebSocket events. Furthermore, my method of using an existing two-way connection to stream updates to a related database without creating a new two-way connection could also be applied to other software companies looking to sync data from multiple streams. Specifically, companies looking to

reduce data sent for low-connectivity or cost reasons could borrow from my implementation as it uses one connection to provide data for multiple data models.

7. UVA Evaluation

UVA does a fantastic job at teaching core CS fundamentals such as time-complexity, debugging, and algorithms through classes like Algorithms and 2150. This is by far the strength of UVA's CS program.

However, UVA is pretty lacking in attracting top tech companies to recruit here and also does not provide the opportunity to create high quality real-world projects in classes. Basically my entire resume is self-taught projects and internships at companies that do not recruit at UVA. I loved my time at UVA but the path to working at a top tech company is much harder here than other similar schools like Michigan.

8. Acknowledgements

This project would not have been possible without my manager, Tom Witkin, and my team (especially Satoshi Nakagawa and Tracy Stampfli who provided key technical guidance).

References

Stoeckli, E., Uebernickel, F. and Brenner, W. (2018). Exploring Affordances of Slack Integrations and Their Actualization Within Enterprises - Towards an Understanding of How Chatbots Create Value. Retrieved September 20, 2022 from https://aisel.aisnet.org/hicss-51/dsm/social_is/2/

Pichette, J., Brumwell, S., Rizk, J. (2020) Improving the Accessibility of Remote Higher Education: Lessons from the Pandemic and Recommendations. Toronto: Higher Education Quality Council of Ontario from https://heqco.ca/wp-content/uploads/2020/10/Formatted_Accessibility_FINAL.pdf

Federici, S., Bifolchi, G., Mele, M. L., Bracalenti, M., De Filippis M. L., Borsci, S., Gaudino, G., Amendola, M., Cocco, A. and Simonetti, E (2022). Remote Working: A way to foster greater inclusion and accessibility? (January 2022). Retrieved September 20, 2022 from https://link.springer.com/chapter/10.1007/978-3-031-08645-8_23

Viktoria Stray and Nils Brede Moe (2020). Understanding coordination in global software engineering: A mixed-methods study on the use of meetings and Slack. (July 2020). Retrieved September 20, 2022 from https://www.sciencedirect.com/science/article/pii/S0164121220301564