InterConn

Cause we interconnect

Mahesh Kukunooru, Maheedhar Gunnam, Rohila Gudipati
Department of Computer Science, Old Dominion University
Norfolk, Virginia

***Abstract*—InterConn is a project that helps users to collaborate with each other either as a group being in channels or individually. The content shared through InterConn can be text, code snippets, images, and files. In addition, users can react to the messages and can create a thread for the posted messages as well. A neat feature called Live Collaboration is also implemented. In this paper we made an attempt to elaborate on this feature to its depths.**

Keywords—Live Collaboration; WebSockets; Firebase

I. Introduction

InterConn is a website we developed, works like the very popularly used collaborative tool called Slack. This tool allows the user to post messages in channels, directly to another user, makes it possible for the user to like and dislike the messages posted, allows the user to search for other user profiles. Along with these features, live collaboration satisfies an effective use case for the users to be able to collaborate live by looking at, editing each other changes at real time like Google Docs. Third party APIs like Gravatar for standardize avatars across the platforms, GitHub API for Authentication through GitHub, No Captcha ReCaptcha for solving bot issues and finally Firebase to get the Websockets implemented toward accomplishing live collaboration were used. LAMP is technical stack used for implementation of this project.

II. Problem

As far as the current state of InterConn goes, the communication over channels, direct messages isn't actually a live one, meaning there isn't actual live chatting kind of environment implemented, a user can't actually see the message posted by others until the page is refreshed. A quick fix would something be like implementing a code that auto refreshes the page for every 10 seconds or so, but the problem is that there is a heavy burden on the network, cause the same resources (HTML, CSS, Javascripts) are being downloaded again and again and ofcourse, the reloaded content has to be re-rendered everytime after the download. Even after implementing this quick fix, the actual problem still persists, the message doesn't appear instantly, the only advantage here is the user doesn't have to refresh the page manually.

III. Solution

The above problem can be resolved by implementing a protocol called WebSockets. The WebSocket protocol enables interaction between a browser and a web server with low overheads, facilitating real-time data transfer from and to the server. This is made possible by providing a standardized way for the server to send content to the browser without being asked by the client, and allowing for messages to be passed back and forth while keeping the connection open. In this way, a two-way (bi-directional) ongoing conversation can take place between a browser and the server.

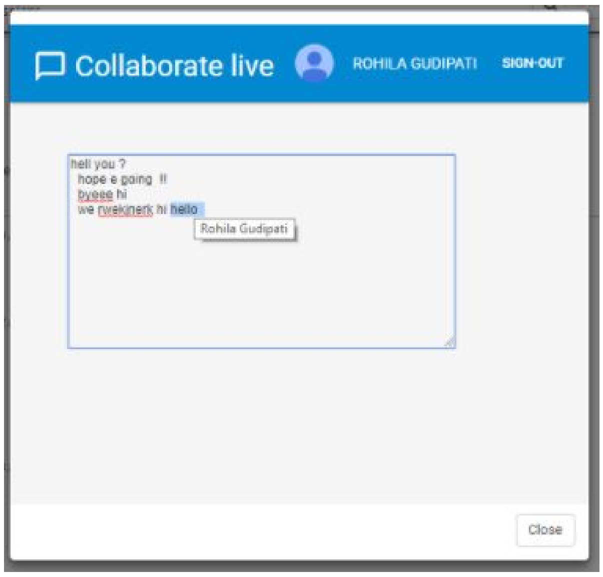
In layman's terms, traditional web apps (even the ones that use AJAX) talk to servers by sending separate messages everytime without the context being preserved, and as a solution WebSockets are a way to establish permanent connection between a web browser and the server.

IV. Implementation

In our preliminary research to address this problem, very quickly we arrived at WebSockets to be the better solution. And there are two ways to get this implemented, the first way is to implement the WebSocket protocol itself and the second one is to integrate any of the third party APIs(Google Realtime API, Google's Firebase, HackMD) that provide readymade WebSocket way of communication. We chose the second way of implementations for various reasons.

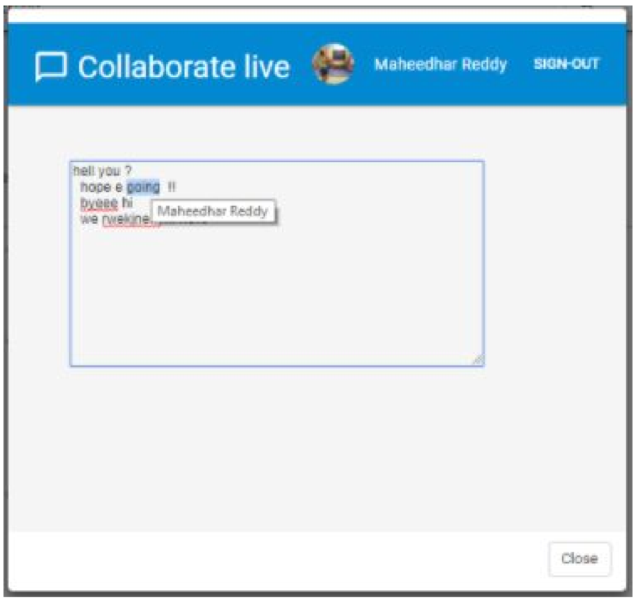
The reason behind choosing the latter way is that the whole project is developed on LAMP Stack, Websockets are usually implemented using Node.js and Tornado web server. And due to the fact that PHP is developed as a programming language for web pages which take the request-response format, there isn't much support for WebSockets and which then would require us to write these WebSocket services in a different language and ultimately require us to deploy the same in a dockerized environment too, this made us to follow the second way for this project. However, a further research made us learn that there exists libraries like Ratchet would simplify the whole process of implementing WebSockets for PHP.

Going ahead, the top choices available were Google's Realtime API and HackMD. HackMD is a real time, multi platform collaborative markdown note editor. Using this service one can write notes with other people real time, though HackMD provides many neat features it is in early stage and is provided with a Git Repository which is required to be hosted on any other server for the usage which then can be integrated with InterConn.



hell you ?
hope e going !!
byeee hi
we rwekinerk hi hello

Fig. 1.a Live collaboration UI



hell you ?
hope e going !!
byeee hi
we rwekine

Fig. 1.b Live collaboration UI

So the other best choice available was Google Realtime API, a preliminary research proved that it is apt for our requirement and with a POC that we implemented made it look promising to provide a matured Google Docs kind of feature to our InterConn upon integration. But on the day we sat for an actual implementation we came to know that this service is recently deprecated and can't be used for any of the new projects since November 28th 2017, which was of a huge shock to our team, but we quickly learnt that Google's focus is shifting more towards it Firebase API and also would be an apt fit to our application. So our destiny was Firebase and we explored and analysed a sample project from their documentation(given in references) which has basic chat like features implemented, we then extended this basic one to make it a real time collaboration tool and fit into our Interconn. Fig1.a and 1.b shows this live collaboration.

V. Future Enhancements

Our live collaboration tool is usable enough, but of course not as mature as the Google Docs itself, certain edge cases like drag selection on the text for deletion would not work, updating the last word of a line would not work, these are on our future enhancements list. To utilize this feature one has to login through their Gmail account, to bypass this and making a GitHub authentication available is one of our future enhancement. This would then enhance the usable of this feature as our site comes with GitHub authentication as well.

Acknowledgment

We are very thankful to Dr. Justin Brunelle for the wonderful guidance and also being available all time fixing our issues on the spot. We very much appreciate the other groups for sharing the various useful links on the slack channels. Never the least we wouldn't have finished our project on time if there isn't help from the online web resource, we highly respect their presence.

References

[1] https://hackmd.io/features
[2] https://internetdevels.com/blog/implementing-websockets-using-php-ratchet-library-or-tornado-web-server
[3] https://codelabs.developers.google.com/codelabs/firebase-web/#0
[4] https://developers.google.com/google-apps/realtime/overview
[5] http://socketo.me/
[6] https://firebase.google.com/docs/database/web/start