CSE 155: Final Report

Author[1]

[1]Adrian Darian, Orion Johnson, Shivanshu Gupta

adarian@ucmerced.edu, ojohnson2@ucmerced.edu, sgupta35@ucmerced.edu

1. Introduction

Our project will be a tool that allows a user to link multiple chatting or messaging services together. For example, if your company uses Slack or hangouts to communicate, but you have some private chats with co-workers on facebook messenger. The user can now link all the accounts together and read or send messages in between. Starting a meeting via video call, will be as simple as selecting which platform the meeting will be on from your linked accounts and sending an invite to that group. We are not forcing users to subscribe to our service or anything, it would be free to use for as much and as long as the user wishes. The idea behind it, mainly came from teams onboarding new members and watching managers explain how they use this service for X and this other service for Y, while for all other forms of talk they use service Z.

2. Related Work

A ex-employee at the CatCard office informed us of chat services like SSHChat and Matrix that allow anyone to connect to a given server and message between each other from any device. The problem was these approaches only worked in the terminal and you had to send and receive messages from the terminal to each other. Which for non Computer Science Students would go way over their head and be very confused on how to get started.

3. Motivation

We wanted to bring about a solution to on boarding new members into a project, organization, or company in a clean and simple. There was a huge headache when trying to setup a new member or employee in a new team. This solution would cut back on one of many issues to be tackled in the on boarding process.

3.1. Problems

When working at a company you are typically introduced to a new platform to communicate with your team in or provided with one.

1. Older generation forced to adept to a new platform
2. Multiple platforms with similar channels to communicate
3. Different organizations that all would like their members to communicate together but do not want a new platform
4. Managers having to spend time teaching new employees or creating a "how to" doc for them

3.2. Why/How

Looking at this scenario, you are in a company that has an engineering team, logisitics team and a marketing team. You are tasked with communicating with individuals from another team. Normally you would need to join what ever platform they use on their team and then message each individual there. Now you have multiple platforms to check each day at work. With a bridge you would only need to be invited to a channel or group from that team and you can start communicating right away.

4. Design

We built a single server that acts as an entry point to bridge different platforms together. This allows a user to message with their personal account on any connected platform and all other members that can view or have access to that channel will receive the message and may reply. Because we can do all this with one server the whole project is actually quite a simple design. The only difference we would need to perform would be what the names and avatar icons are when new members join the new channels.

5. Implementation

To connect each platform together was a little more difficult than imagined. For example to link specifically Slack and Discord to this bridge we need to build a bot for each platform and invite it to each discord server and slack workspace. This is by design so the bot is the one forwarding the messages to the other platforms. While for Microsoft Teams we were able to load the bridge as an application on Microsoft Azure and have that forward messages through it to the deployed bridge server. As per Gitter and Telegram platforms we simply needed to create a temporary user and all messages would filter through them. Leaving it to just invite the user to each channel and everything was setup. As for Whatsapp that was a lot more difficult, due to the amount of encryption in Whatsapp we could not simply add a bot. We had to follow Whatsapp's online guide and build a Webapp host and configure a publisher/subscriber connection stream between that and the channels in Whatsapp.

It is to be noted that the whole service is a single privatized service on a server. This being said if we wanted to add another channel or another group to the list of connected chat rooms under the umbrella of this system then we would need to manually update the system. The only current way of updating the system would be to connect to the main server with some admin credentials and update the files themselves then restart the service. Which would then ping everyone online that the system will be doing a quick reboot. We were able to fix the reboot time to a less than a second per connected platform.

6. Evaluation

In our studies we decided that we wanted to focus solely on the user experience of the multiplatform chat application. This is strongly due to the fact that if we focused on some UI or some features it would feel more like we are selling a mobile or desktop application to the users and that would be painstakenly boring and exhausting resources to "become the best chat platform." Which takes away completely from our goals. By focusing on the user experience we can better modify and adjust the specifics of the service to meet user's interests and demands and provide customizability to each platform they choose to use.

6.1. Apparatus

In this case study we need not code any front end graphical designs because we are relying on the users to pick their favorite platform from our test platforms and use that one. These handpicked platforms are actual company platforms and it is to be noted we are not placing any bias on either of the platforms, we are just using them to showcase the possibility of connectivity between chat services. All the code written and engineered in this case study ended up on a Digital Ocean Droplet which is a cloud Ubuntu server that is hosted in the cloud. By setting it up this way we can asynchronously connect each platform to one central messaging broker and establish a publisher and subscriber system of connections.

6.2. Participants

Adrian was able to bring in several friends and collegues that he has in his network to help test our this project. They are all from different age groups ranging from 12 years of age to 67 years of age. The reason for this wide range was due to the wider range one might see on a team or a company project. We also made sure to make sure for every male there was at least one female and we even were able to acquire some non-binary people to help test the service.

6.3. Procedure

Just like any other chat platform you do not want to limit or restrict the freedom of speech between people in a platform. So we invited an initial group of people and told them to just talk to each other. As more people joined the group chats and conversation groups we were able to see conversations pinging off one another as normal. This allowed us to then add some spice and allow them to send images, files, and links. The biggest part of this test was that we set it up and let them go at it for hours. Where they could stop at any point in time and resume, because that would simulate someone pausing and going to a meeting or class or even an emergency came up. Then they would rejoin when they got back.

6.4. Metrics

This is without doubt the smallest section, because the only data we collected back was just user happiness and how well they enjoyed it on that specific platform. As shown in our presentation. Since the data was more in line with how people would rate a movie or a video game. I.e. a 4 out of 5 star rating. We did not end up making any graphical results for it. However, we found that the Discord service would ping back and repeat messages making the feel too uncomfortable to be used. Which will need some work in the future. However, the Slack, Telegram, and Gitter test platforms worked like a charm. Participants could share any form of media and exchange messages back and forth in real time. The only common concern brought up was the design of the message header that was being broadcasted. But in a production environment we would exclude the platform from which the message would come from and just say the name of the sender.

7. Results

We tested the deployed application out with a few friends and coworkers from different clubs and companies and got an overwhelming positive feedback. Not only was it more conveinent to just open one app of choice and start talking to anyone on your team (test group), but the accessibility to even start a private conversation with someone when they were on Discord and the other was on Telegram, was seemless.

However there were many concerns raised about the security of opening the platforms up to one communication channel. This does lead to a single point of failure for a breach that leads to all of a user's messaging history. This is something that we found no solid answer for given how the project is set up at this time. We feel the ultimate answer to this would be to have 'OneStopShop' to be its own standalone application with high encryption.

8. Future Work

In conclusion with more time to figure out how to connect more platforms together and researching more into encryption and security this could be a actually company requirement or neccessity. Right now it just lacks the breath of how many platforms can be connected and especially how secure communicating to a channel on another platform is. We would also want to streamline the auto deployment of new channels, group chats, and platforms into said service. We would then want to incorporate the ability to host voice and then video calls. Where people may start a call and join from either device seemlessly.

References