DSCI551 Project Final Report – USC Whispers

Team member

Chia-Hsuan(Sharon) Lee, Jiajin Luo

Topic

USC Whispers: An emulated Firebase and a real-time chat application

Scripts and Documentation Link

https://drive.google.com/drive/folders/1_pw7VH3JA_mrWR66JyFgXhO19y7l6SH7?usp=sh
are_link

Implementation

Concepts and Theories

In this project, our aim is to develop a platform exclusively for USC students where they can unwind and socialize. Our platform will allow students to use any username and share their thoughts freely, visible to all users of the platform. Our app comes with an administrator interface that allows us to moderate the content by editing or deleting inappropriate messages, sensitive information, or private data. Additionally, we can save the messages in the record during high message volume.

In our application, we utilized a combination of Flask, Socket.IO, and MongoDB to create a powerful and flexible solution. Firstly, we used Flask to create a web server with endpoints for clients to connect and interact with. Next, we set up our MongoDB database with four collections to handle further requests, including storing usernames and messages. This enables users to access their message history and continue conversations even after refreshing the page.

To structure the webpage for the client's usage, we created an HTML file. Socket.IO was then used to connect the server and the client, allowing real-time communication. With Socket.IO, messages can be sent and received instantly. Whenever a message is sent from a client, the client sends an event to the server and passes data along with it, and the server sends an event to all connected clients, enabling instant messaging.

Overall, our chat application is a robust and scalable solution that can handle real-time messaging and store message history. By leveraging Flask, Socket.IO, and MongoDB, we were able to create a reliable chat application that meets the needs of modern communication.

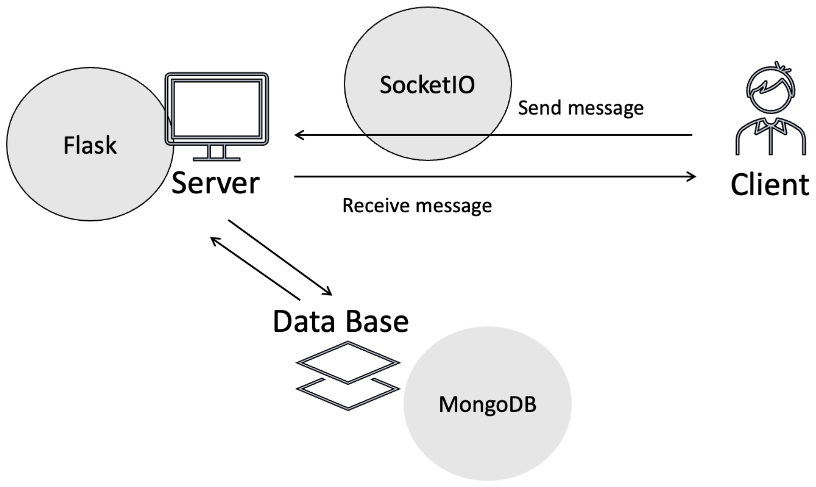


Image A: Structure of the application

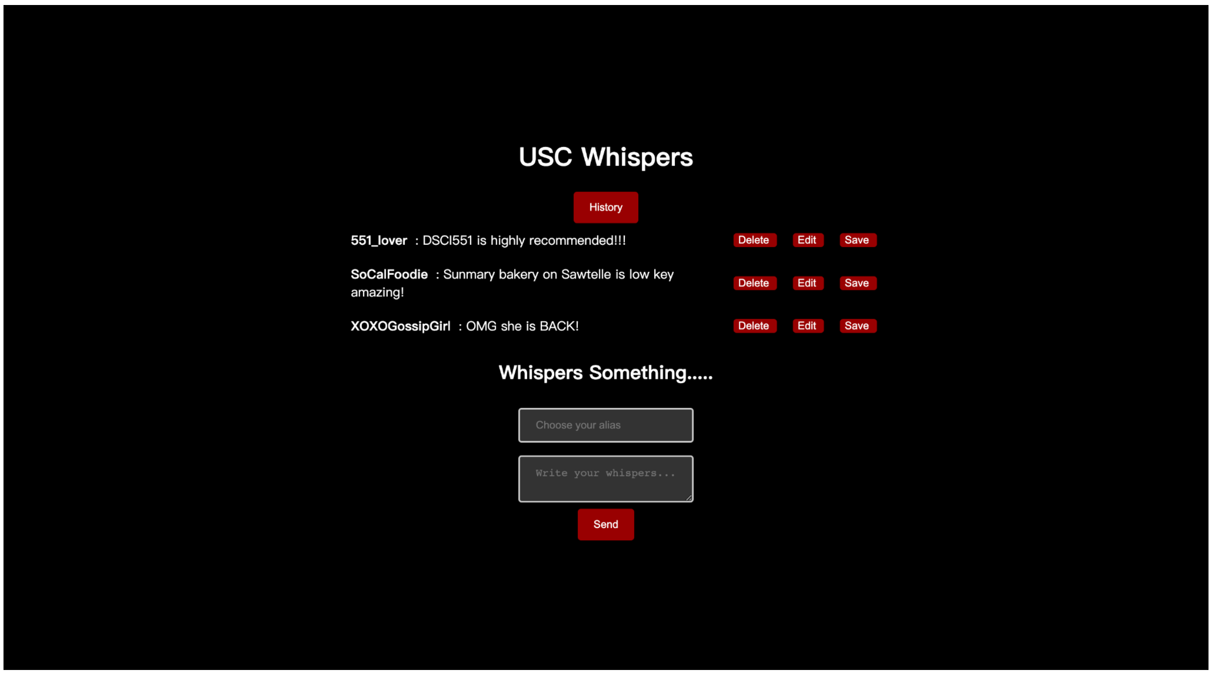
Emulating Firebase: Support curl command

- Create indexes for each element
  We have defined a "create_index" function that generates a numeric index to replace the original messy index from MongoDB each time an input is made. The index begins at 10001 and increments by one for every subsequent input.
- RESTful API and filtering functions
  To manage incoming requests and perform actions on collections in MongoDB, we use the "requests" module in the Flask library and functions in the PyMongo library, such as insert_one, delete_one, update_one, and replace_one. This module enables us to handle CRUD (GET, PUT, PATCH, POST, DELETE) and filtering functions (orderBy= $key/$value, limitToFirst/Last, equalTo, startAt/endAt) for curl operations effectively. By leveraging this functionality, we can ensure that the server can intercept responses and effectively manage incoming requests. Please refer to the appendix for the screenshots of these curl commands.
- Store JSON data in collections on MongoDB. Each time the server receives a request, it turns the data to json format and operates on MongoDB using the functions in PyMongo. This allows us to interact with the database and perform various operations.

Build a real-time chat app – USC Whispers

- Build a pretty and user-friendly UI
  Image B shows the application interface in the administrative view. It features a clean and user-friendly design that we made with HTML and Javascript. Users can easily input usernames and messages, which will appear above.
- Allow real-time syncing of the data
  Taking advantage of WebSockets, the app can communicate with our database to store and grab data in real-time. So whenever a user inputs their names and whispers, it will appear on the page immediately, allowing users to chat in real time.
- Various functions for administrators to manage the app

The app offers a range of options for each message sent. You can edit, delete, or save it to your message history. This makes it easy to keep track of your message and refer to them later. Refer to the appendix for more screenshots of the interface.



551_lover : DSCI551 is highly recommended!!!

SoCalFoodie : Sunmary bakery on Sawtelle is low key amazing!

XOXOGossipGirl : OMG she is BACK!

Image B: application interface

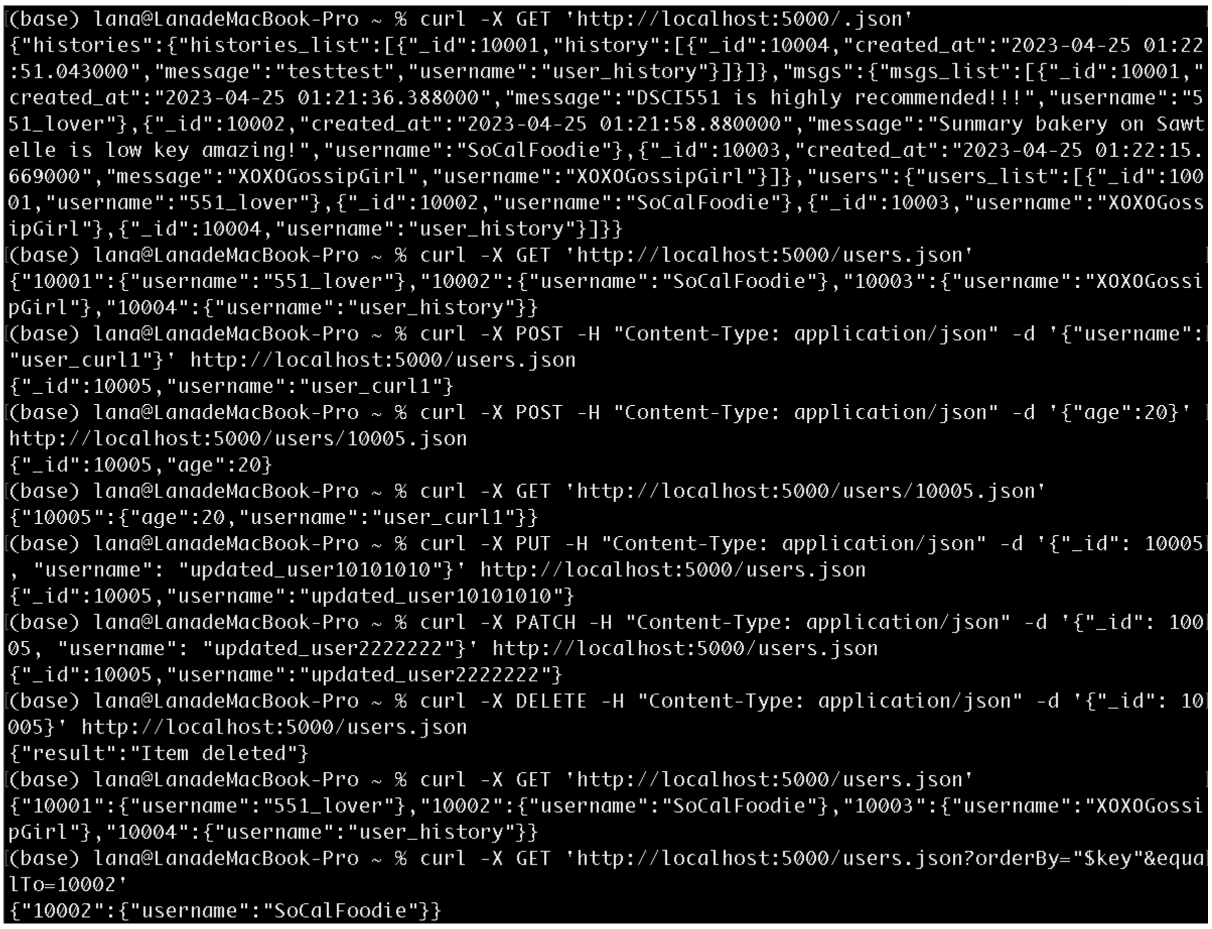
Learning Experiences

It was really tough work for our group, as neither of us was a CS student. Throughout the whole project, we acquired tons of knowledge in different fields. As our work was focused on using Flask, Websocket, and MongoDB to emulate Firebase and then build a real-time chat application, we were able to learn more about the NoSQL database and the front/back-end of a web application. Flask allowed us to develop a web server with client endpoints, Socket.IO facilitated real-time communication between the server and client, while MongoDB provided a reliable database for storing and grabbing user data and messages.

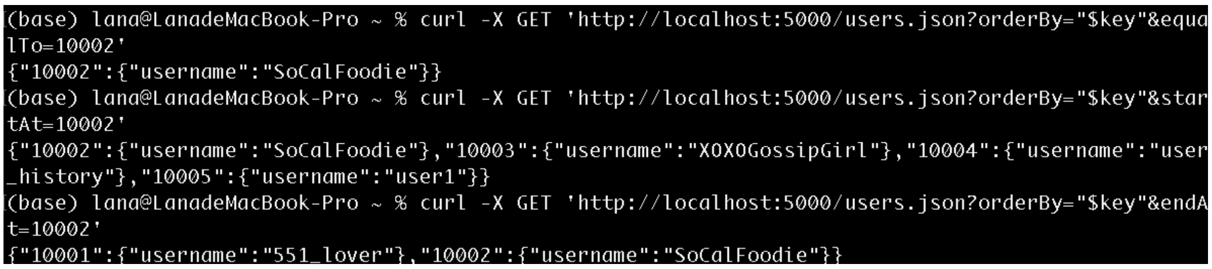
We found setting up the database and allowing curl commands the easiest part, as it only requires basic knowledge of Python and database, and we then spent some time learning how Flask and requests work. The most tricky part was connecting the script with the interface using Socket.io, it did not work whatever we tried, so it took us tons of time going through the documentation, StackOverflow and Youtube Videos. Luckily we made it finally and the UI looks pretty cute.

Appendix

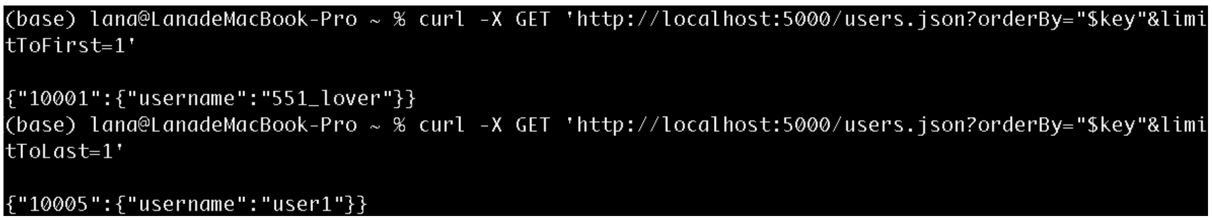
1. Trying curl commands in terminal (GET, PUT, POST, PATCH, DELETE, orderBy)



```
(base) lana@LanadeMacBook-Pro ~ % curl -X GET 'http://localhost:5000/.json'
{"histories":{"histories_list":[{"_id":10001,"history":[{"_id":10004,"created_at":"2023-04-25 01:22
:51.043000","message":"testtest","username":"user_history"}]}]},"msgs":{"msgs_list":[{"_id":10001,"
created_at":"2023-04-25 01:21:36.388000","message":"DSCI551 is highly recommended!!!","username":"5
51_lover"},{"_id":10002,"created_at":"2023-04-25 01:21:58.880000","message":"Sunmary bakery on Sawt
elle is low key amazing!","username":"SoCalFoodie"},{"_id":10003,"created_at":"2023-04-25 01:22:15.
669000","message":"XOXOGossipGirl","username":"XOXOGossipGirl"}]},"users":{"users_list":[{"_id":100
01,"username":"551_lover"},{"_id":10002,"username":"SoCalFoodie"},{"_id":10003,"username":"XOXOGoss
ipGirl"},{"_id":10004,"username":"user_history"}]}}
(base) lana@LanadeMacBook-Pro ~ % curl -X GET 'http://localhost:5000/users.json'
{"10001":{"username":"551_lover"},"10002":{"username":"SoCalFoodie"},"10003":{"username":"XOXOGossi
pGirl"},"10004":{"username":"user_history"}}
(base) lana@LanadeMacBook-Pro ~ % curl -X POST -H "Content-Type: application/json" -d '{"username":
"user_curl1"}' http://localhost:5000/users.json
{"_id":10005,"username":"user_curl1"}
(base) lana@LanadeMacBook-Pro ~ % curl -X POST -H "Content-Type: application/json" -d '{"age":20}'
http://localhost:5000/users/10005.json
{"_id":10005,"age":20}
(base) lana@LanadeMacBook-Pro ~ % curl -X GET 'http://localhost:5000/users/10005.json'
{"10005":{"age":20,"username":"user_curl1"}}
(base) lana@LanadeMacBook-Pro ~ % curl -X PUT -H "Content-Type: application/json" -d '{"_id": 10005
, "username": "updated_user10101010"}' http://localhost:5000/users.json
{"_id":10005,"username":"updated_user10101010"}
(base) lana@LanadeMacBook-Pro ~ % curl -X PATCH -H "Content-Type: application/json" -d '{"_id": 100
05, "username": "updated_user2222222"}' http://localhost:5000/users.json
{"_id":10005,"username":"updated_user2222222"}
(base) lana@LanadeMacBook-Pro ~ % curl -X DELETE -H "Content-Type: application/json" -d '{"_id": 10
005}' http://localhost:5000/users.json
{"result":"Item deleted"}
(base) lana@LanadeMacBook-Pro ~ % curl -X GET 'http://localhost:5000/users.json'
{"10001":{"username":"551_lover"},"10002":{"username":"SoCalFoodie"},"10003":{"username":"XOXOGossi
pGirl"},"10004":{"username":"user_history"}}
(base) lana@LanadeMacBook-Pro ~ % curl -X GET 'http://localhost:5000/users.json?orderBy="$key"&equa
lTo=10002'
{"10002":{"username":"SoCalFoodie"}}
```

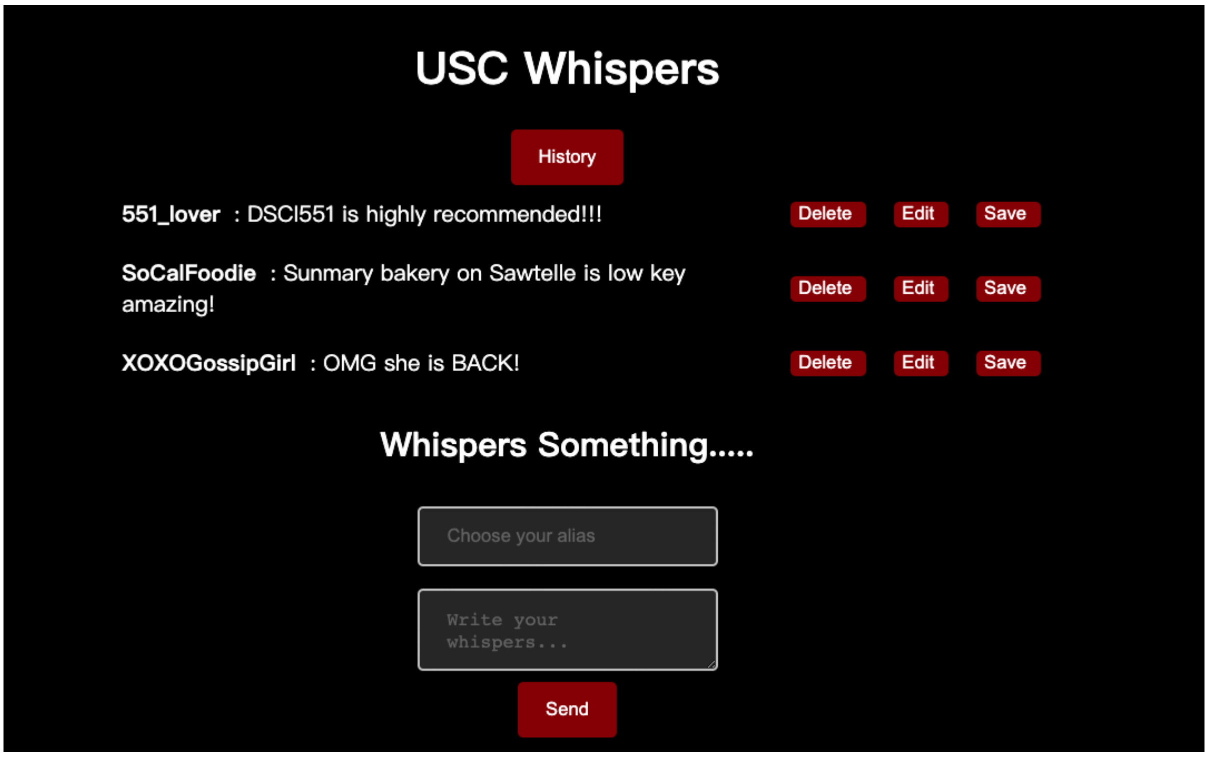


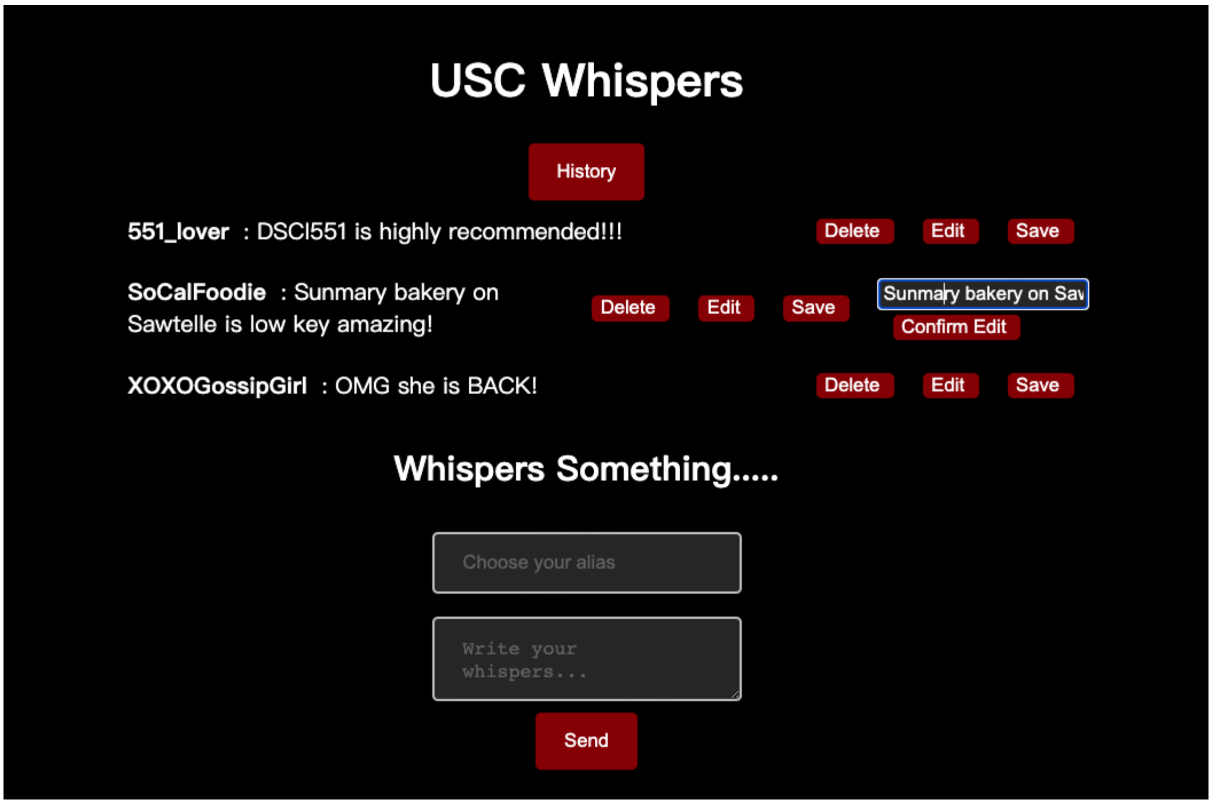
```
(base) lana@LanadeMacBook-Pro ~ % curl -X GET 'http://localhost:5000/users.json?orderBy="$key"&equa
lTo=10002'
{"10002":{"username":"SoCalFoodie"}}
(base) lana@LanadeMacBook-Pro ~ % curl -X GET 'http://localhost:5000/users.json?orderBy="$key"&star
tAt=10002'
{"10002":{"username":"SoCalFoodie"},"10003":{"username":"XOXOGossipGirl"},"10004":{"username":"user
_history"},"10005":{"username":"user1"}}
(base) lana@LanadeMacBook-Pro ~ % curl -X GET 'http://localhost:5000/users.json?orderBy="$key"&endA
t=10002'
{"10001":{"username":"551_lover"},"10002":{"username":"SoCalFoodie"}}
```

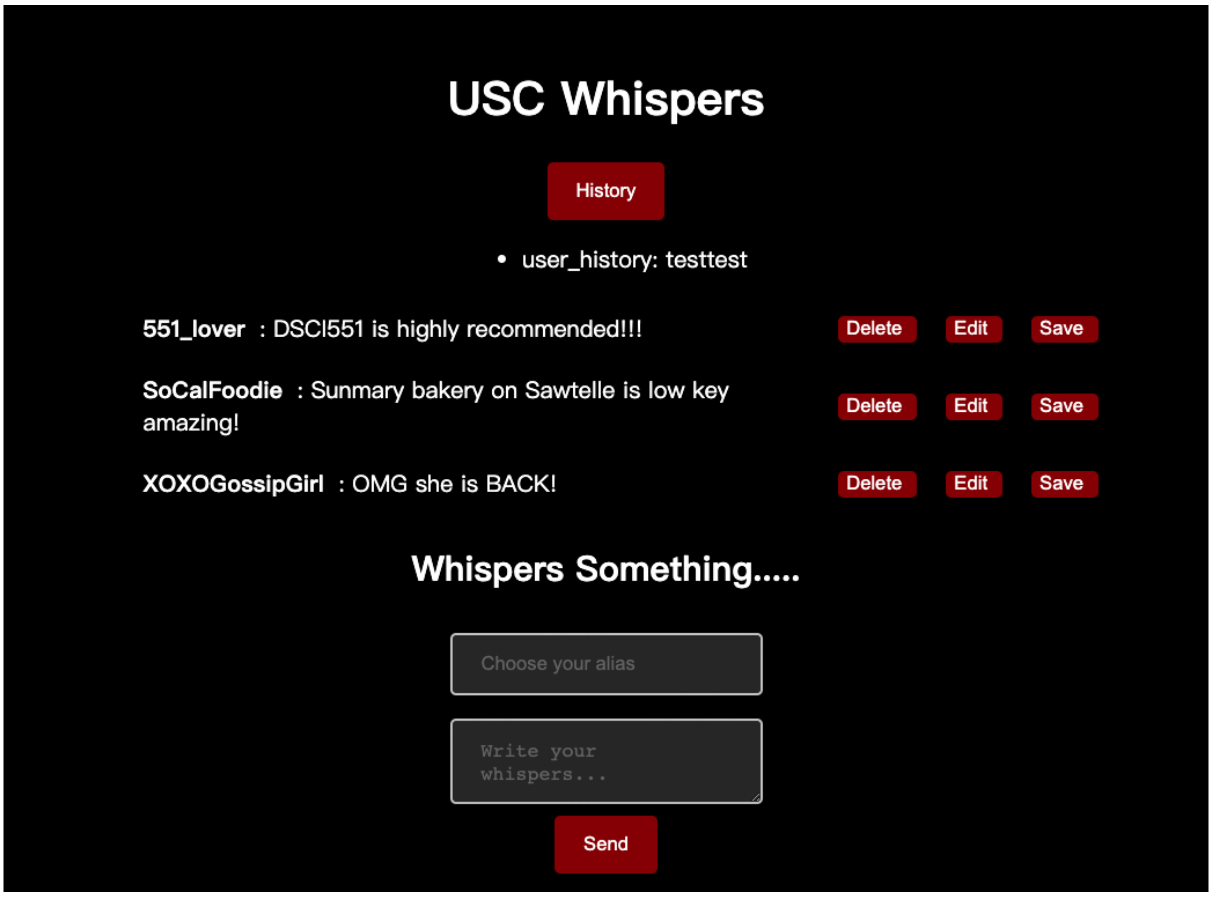


```
(base) lana@LanadeMacBook-Pro ~ % curl -X GET 'http://localhost:5000/users.json?orderBy="$key"&limi
tToFirst=1'

{"10001":{"username":"551_lover"}}
(base) lana@LanadeMacBook-Pro ~ % curl -X GET 'http://localhost:5000/users.json?orderBy="$key"&limi
tToLast=1'

{"10005":{"username":"user1"}}
```

2. Interface of the app


USC Whispers
History
551_lover : DSCI551 is highly recommended!!!
Delete
Edit
Save
SoCalFoodie : Sunmary bakery on Sawtelle is low key amazing!
Delete
Edit
Save
XOXOGossipGirl : OMG she is BACK!
Delete
Edit
Save
Whispers Something.....
Choose your alias
Write your whispers...
Send





USC Whispers

- user_history: testtest

551_lover : DSCI551 is highly recommended!!!

SoCalFoodie : Sunmary bakery on Sawtelle is low key amazing!

XOXOGossipGirl : OMG she is BACK!

Whispers Something.....