CS179I Networks

Final Report

Project Name: Hobby-ever

Team Name: Whatever

Ziyi Huang	**X665335**	**huangzi_yi96111@163.com**
Yinna Wei	**X665715**	**yinnawei63@gmail.com**
Zhengbo Zhou	**X665862**	**a1023478824@gmail.com**

	Reading	T%	C%	D%
Ziyi Huang	100%	35%	20%	80%
Yinna Wei	100%	35%	20%	80%
Zhengbo Zhou	100%	30%	25%	75%

1 Abstract

1.1 Hobby-ever

Hobby-ever is a web app aims at helping individuals learn skills and interests freely. It is an Interest-oriented community, skill-exchange system, and self-improve platform.

1.2 abstraction

we have established an Interest-oriented community helping individuals share their skills and learn new skills with strangers with talents.

In our visual community, we will provide these functions:

1. Login and Registration:
2. User Profile
3. Calendar
4. Matching Paired-Users
5. Searching Paired-Users
6. Rating Paired-Users
7. Displaying Course Lists
8. Uploading and Watching Short Videos
9. Online Chatting Room

The main challenges of the project are implementing the online chatting function and Interacting with the backend database and the backend server. And what is gratifying is that we finally solved these problems.

The innovation of the project is that we have used visual currency to attract more users to show their talents and communicate with others. And we have also used short videos to make our system more attractive and diverse.

Our project mainly developed based on Html + CSS + Javascript + Firebase. There are about 20,000 lines of codes in total.

2 Introduction

2.1 Problem

2.1.1 Online chatting

1. Our entire system should be scalable, it should be adaptable to a different number of online users.
2. Our entire system should be stable and robust. When the system of one of the two communicating parties breaks down, the other one should not be affected
3. The server must be stable so that users can connect to the chat interface and chat at any time

2.1.2 Matching system

1. Get the best results quickly
2. Can accurately identify the user's search conditions and extract keywords
3. Design accurate and reliable data structure and algorithm, realize optimum matching according to user's various limitations and conditions

2.1.3 Video displaying and uploading

1. Minimize lag when playing video
2. sort the videos uploaded by users in a category, so that it will be more convenient for users to view their uploaded videos
3. Select videos from users' computers and upload them to users' personal homepage.

2.1.4 Virtual currency

1. Calculate the sum of each user's virtual currency correctly
2. Implementate the circulation, increasing or decreasing of virtual currency accurately

2.1.5 Real-time video

1. Minimize latency while maintaining audio and picture quality.
2. Our entire system should be scalable, it should be adaptable to a different number of online users.
3. Our entire system should be stable and robust. When the system of one of the two communicating parties breaks down, the other one should not be affected.

2.2 Previous work

Before deciding to implement our system, we have already tested some existing websites for finding a tutor. However, many existing platforms have many shortcomings and need to be improved.

2.2.1 Most of those websites only provide a platform to find a tutor, after that, users have to meet each other offline. Obviously, this would not be a convenient way for most people. For instance, some adults and most teenagers are not able to drive a car by themselves, so that they can not have that much available time to go to a particular place and meet with somebody else. In our system, however, we allow users to choose to meet offline or online. And we even support offer multiple methods to chat online, like real-time video chatting, audio chatting, or simply texting messages.

2.2.2 When users need to find a tutor, almost all existing websites will need the users to choose a tutor on their own. However, most users are not familiar with available tutors, so it would be almost impossible for them to find the most appropriate one that fit their requirements. So ours will match users automatically based on their owned skills and time availabilities.

2.2.3 Almost all the current systems are ‘one-side’ tutoring system, they separate users into tutors and students. But we believe that everyone has his own specialty, if all the paired users can learn from each other, the efficiency of learning will be much higher.

2.2.4 Many previous works only allow users to introduce themselves by writing down their abilities in the profile. However, this could be not that convincing.What’s more, if a user thinks he is better then other users in a particular field, it will be hard for him to describe in words that he is better than others, so we need other ways to prove his ability. So we allow users to upload short videos to attract others. By watching these videos, other users will know who they are and how great their abilities are.

2.3 Solution

Our users are those Teenagers who need a learning platform and a social platform.

Main modules of our system:

2.3.1 log in and sign up

Every user who uses our system will have an account of his/her own. We will provide users with the account registration, login, and exit interface.

2.3.2 User Profile

User Profile will display the current user's basic personal information, which can be modified in the later stage. Our searching system and matching system will use the information here to find the most suitable partner for the user.

2.3.3 Calendar

The calendar will show the user's free time and all his/her appointments, which will make it easier for the current user to see his/her own arrangement. What's more, other users will be able to choose the best appointment time by looking at his paired users' calendars.

2.3.4 MyCourse List

This will show all the courses that the current user has booked, and will also record the paired-user, teaching subjects and teaching forms of each class, so that the user can check his/her own course arrangement conveniently.

2.3.5 Short Videos uploading and playing

In order for users to better show their talents, we allow users to upload short videos about themselves, so that users can have a better understanding of each other.

2.3.6 Searching

The searching system can help users find the specified users or the specified courses.

2.3.7 Matching

The matching system will match the most suitable partner for each user through personal information such as their hobbies and interests. The matching partner should be complementary, that is, they can teach each other what they want to learn.

2.3.8 Online Chatting

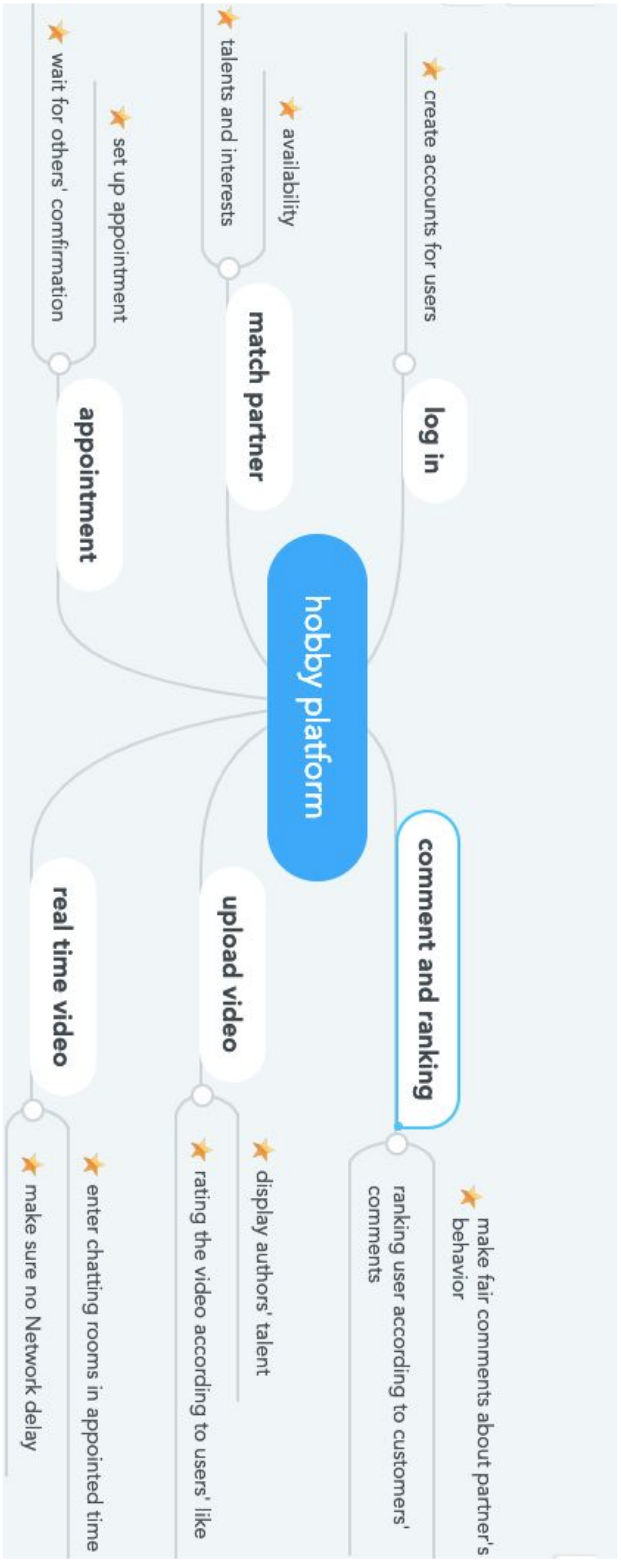
When the teaching method chosen by users and their partners is online teaching, we will provide online chatting function for users. Current online users will be free to discuss, learn from each other, and see whether their paired-users are online or not in our chatting room.

2.3.9 Rating System

Every time a user finishes a lesson, he or she will be able to use a grading system to grade his/her paired user. The grading system will encourage the user to improve his or her ability and the quality of his/her lesson.

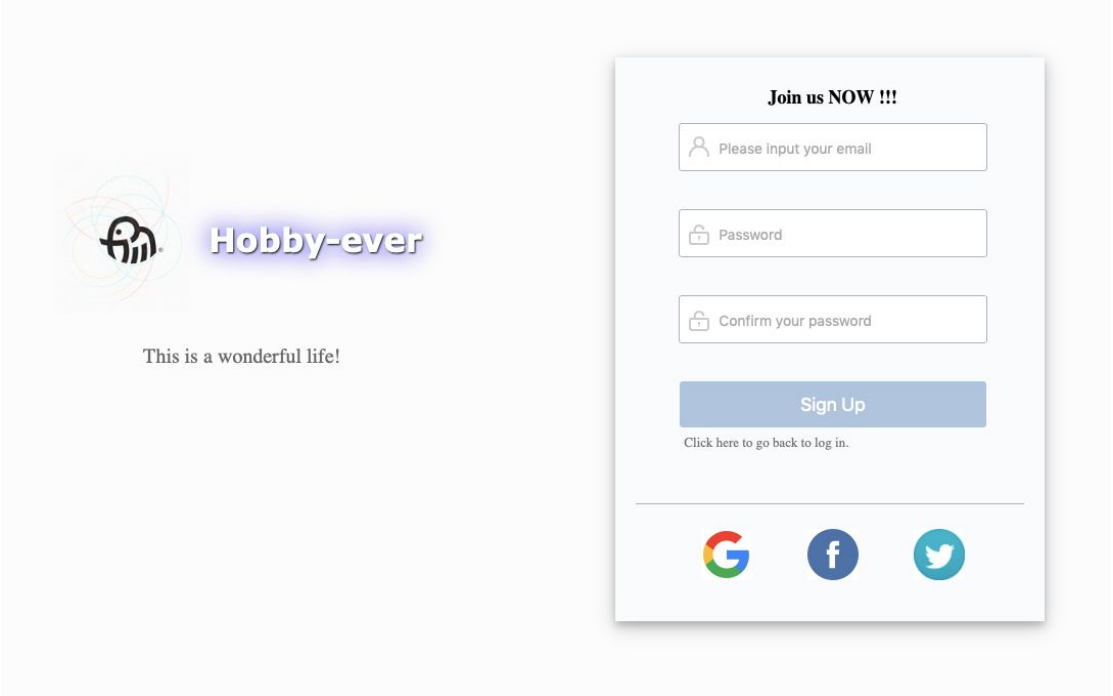
3 Implementation

3.0 Mind Map

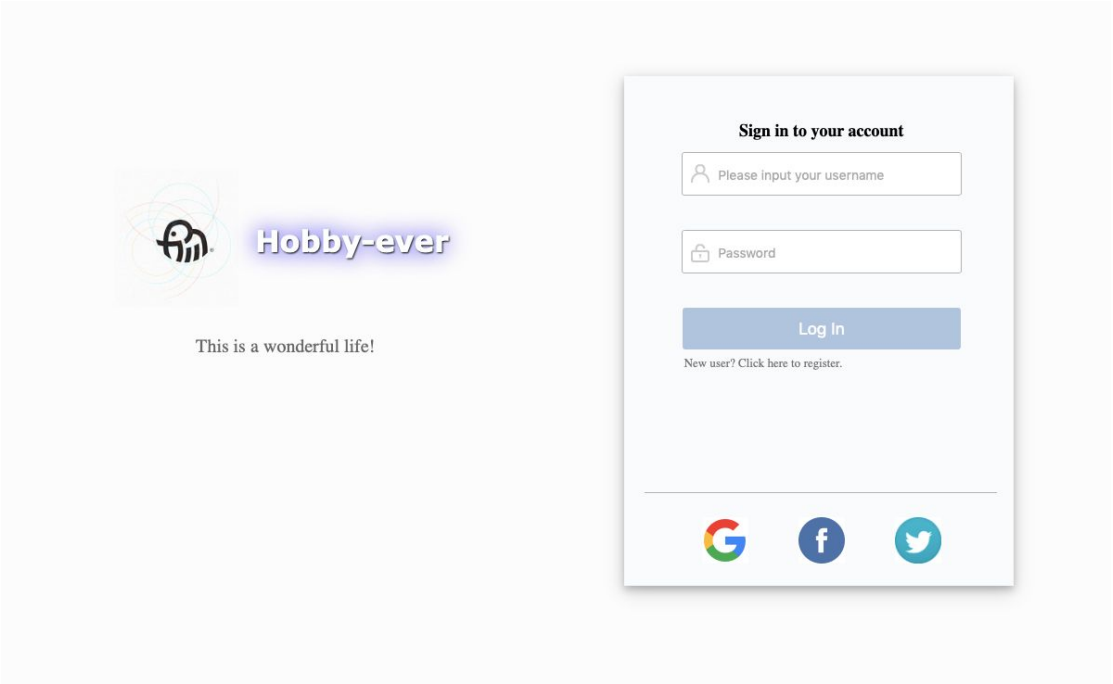


3.1 Login and sign up

In the signup page, we ask users to input his/her email and password, then make use of firebase's createUserWithEmailAndPassword function to insert a new user with the information user entered and assign this user a unique uid automatically. Later, all the other functions will base on this uid to find this exact user.



Then, after a new user signs up his/her new account, he/she will be directed to our homepage. However, the next time when this user wants to access to our system, he/she need to sign in with the correct email address and password, and we will use firebase's signInWithEmailAndPassword function to check user's sign in information. If it is incorrect, the system will alert a message to help user correct the information.

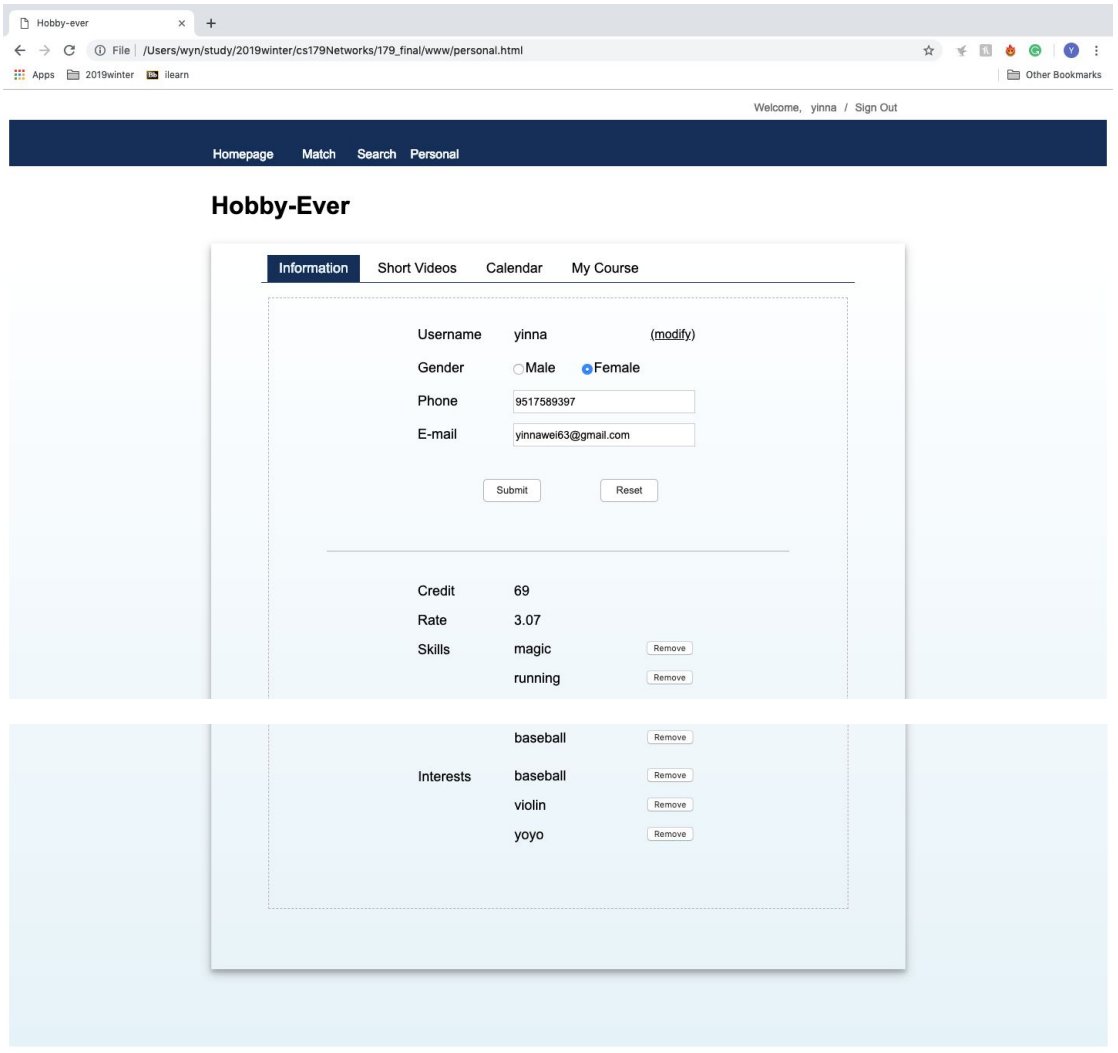


3.2 User Profile

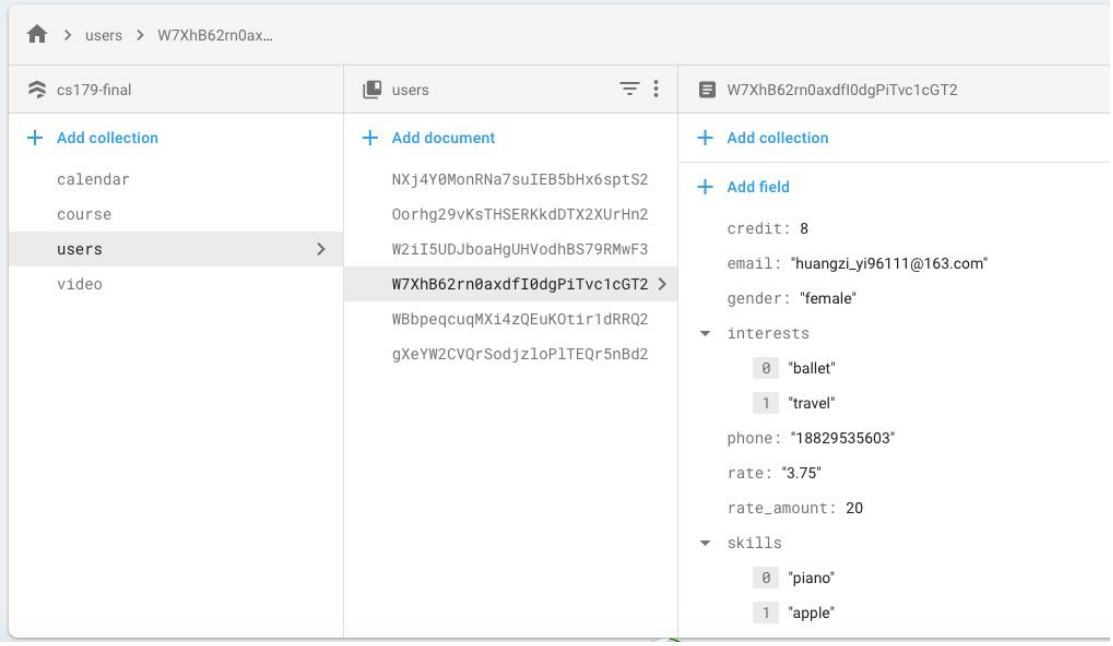
This part is for modifying users' personal information, including the user's phone number, username, skills, interests and so on. Once the update button is clicked, an update function will be called. And the information modification is separated into three different parts. The first one is username update, this is because username should be unique, and in a future system upgrade, we will also use the username as one of our user id to identify users. And the other two parts are basic information and hobbies.

While updating the database, firstly, we find the current user using '`var user = firebase.auth().currentUser;`'. And got all of its information shown in the page, this is because, for basic information, we make changeable information field as input type, users can change their content directly; as for interest and skills, users can add three of them at most, and they are added or removed once the user input a new one and click the add button or click a remove button of a specific item. That means, all the information shown in the current page are the new information that the user

wants to change to. Then, we find the current user's data collection in a database using `firestore.doc("users/" + uid)`. And set all the information with the new one.



And below is an example of a user's profile data in our database.



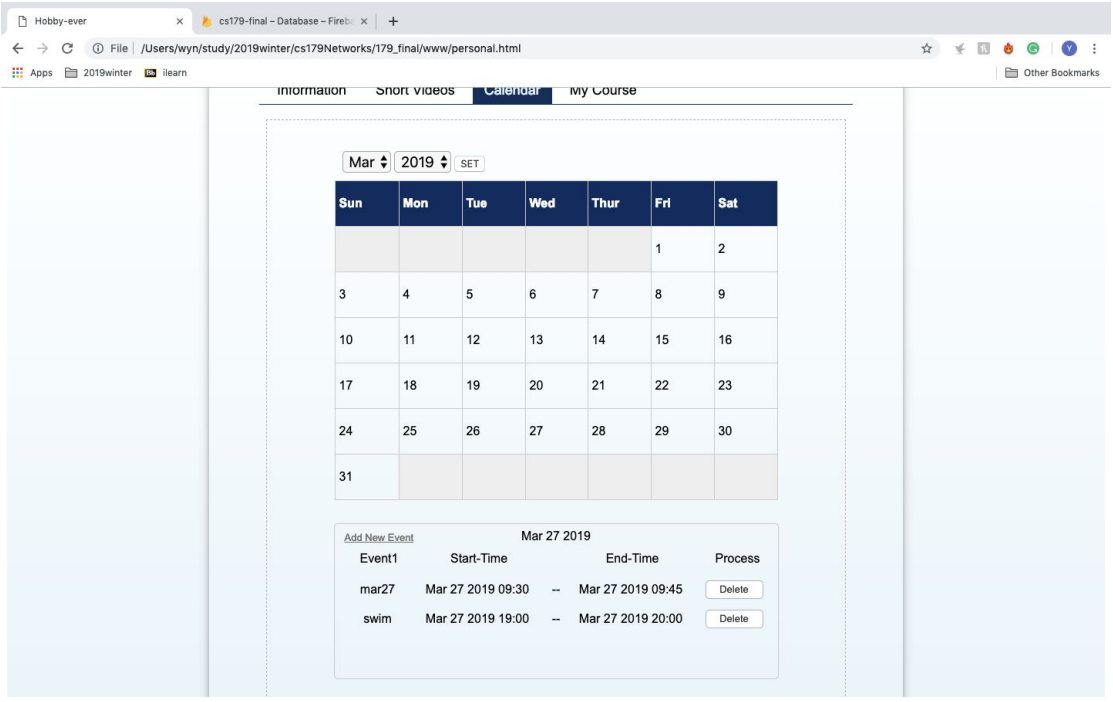
3.3 Calendar

In the calendar part, when the user first comes into this tab, the date chosen will be default as the current date. And there would be event list of the chosen date below the calendar, and it will show the event name, start time and end time of this event and also there would be a button enable users to cancel this event from his/her calendar. And users can also set the calendar to another month or year and click on the date they want to check or add a new event on. Then the event list will change to the date chosen.

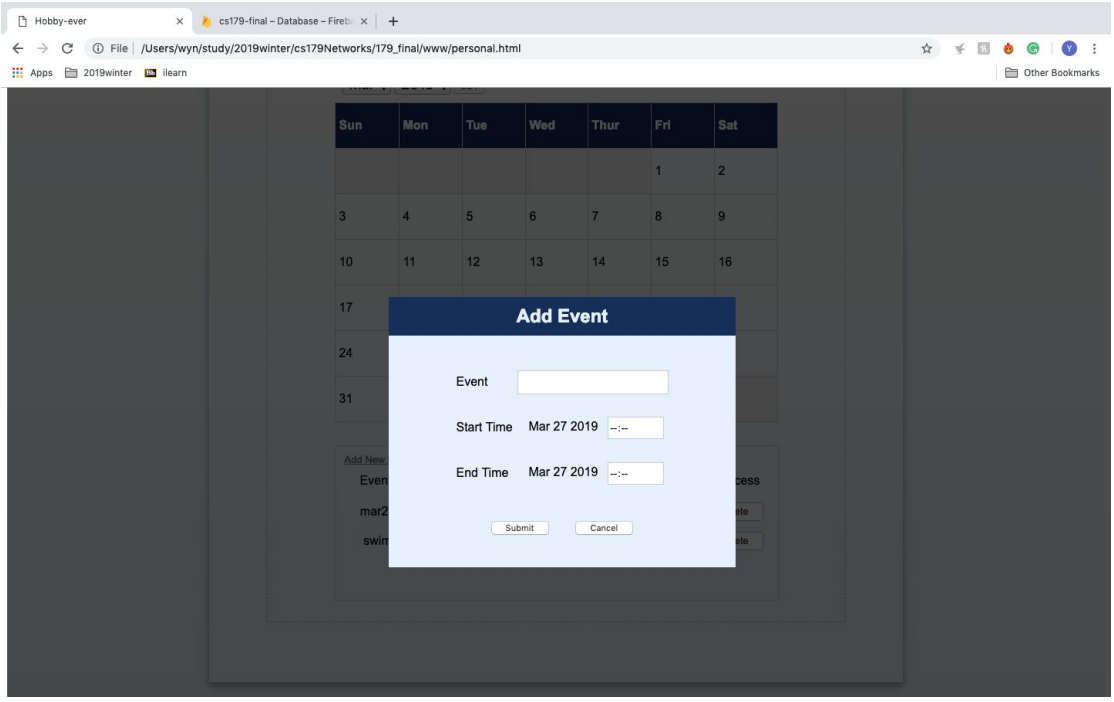
Here we get current date from system date to show as default. And for the whole calendar part, we set is as a table and find out which table item is clicked to set the chosen date.

To implement the add and show list functions, we first need to have the current user's uid, which we can easily get from firebase.auth().currentUser.uid. Since when we store a new event, we will save uid of the user who owns this event. When we need to show the event list, we use `firebase.firestore().collection("calendar").where(" user_id", "==", uid).get()` to find all the event of this user first. Then we did a loop looking

up each event, if it's event_date is equal to the chosen date, then we will display it, otherwise we will do nothing and move to the next event.



While adding a new event, users need to enter the detail of this event, and we will save them all as a whole item into the calendar document in firebase, and since we use add function from the firebase, it will be assigned a unique id automatically.

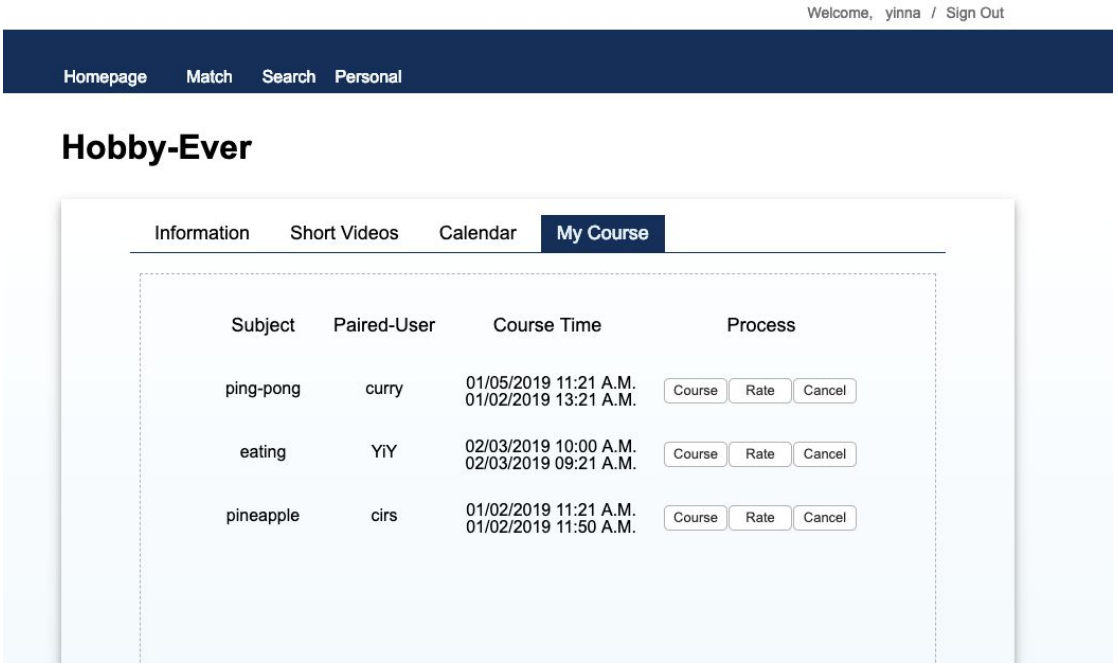


Then, since we saved each event's id while loading the event list, we can also make use of it while deleting. Refer to the specific document of the event_id and use firebase's delete function to remove this event. Update the event list once the data has been saved.

3.4 My Course List

The last tab of the users' personal page is the user's course list. This part contains all the courses that this user has attended and will attend. And this list contains some useful details of each course. Similar to the event list, we make use of the current user's uid to look up the 'users' collection in the database. And find out all the courses that the user1_id is the current uid or the user2_id is the current uid (This is because in our system, each user is both student and tutor at the same time in each course, and each course will always contains and only contains two users. So no matter which user of this course is the current user, it means the current user is in this course). And then doing a loop within all the founded courses and display them in the course list one by one. And there will be three options for users to do with each course, which are go to the online course page and rate the paired-user and cancel a specific course. Users can also click on their paired-user's username here to go to the paired-user's personal profile.

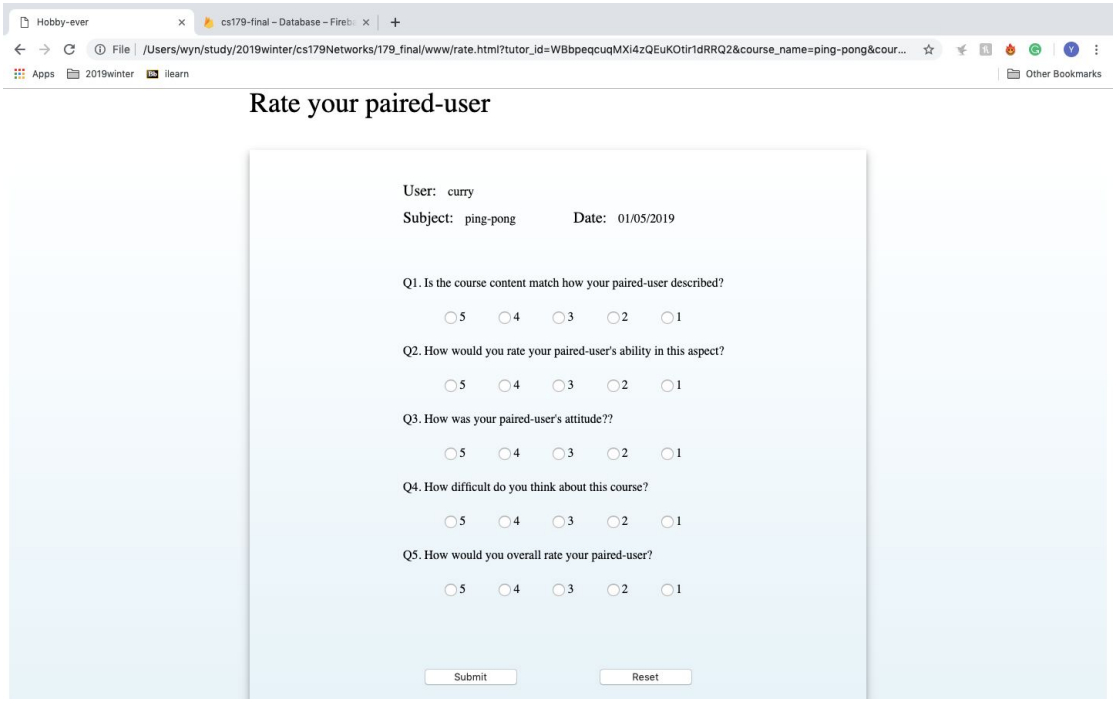
To cancel a course, it is also similar to remove an event from your calendar, while loading the course list, we add courses one by one and we will transmit the course_id to delete_course function. Then we make use of firebase's delete function to remove this course from the course document.



3.5 Rating

After click rate button of a specific course in the user's own course list, the system will go to a rate page and let the user to rate his/her paired-user. Users can choose to answer any amount of questions here and the score will be counted based on the number of questions this user has answered. We will count the average score of this review and find the original score of the rated user and the number of users that have rated him/her. Then count the new score and update the paired-user's data.

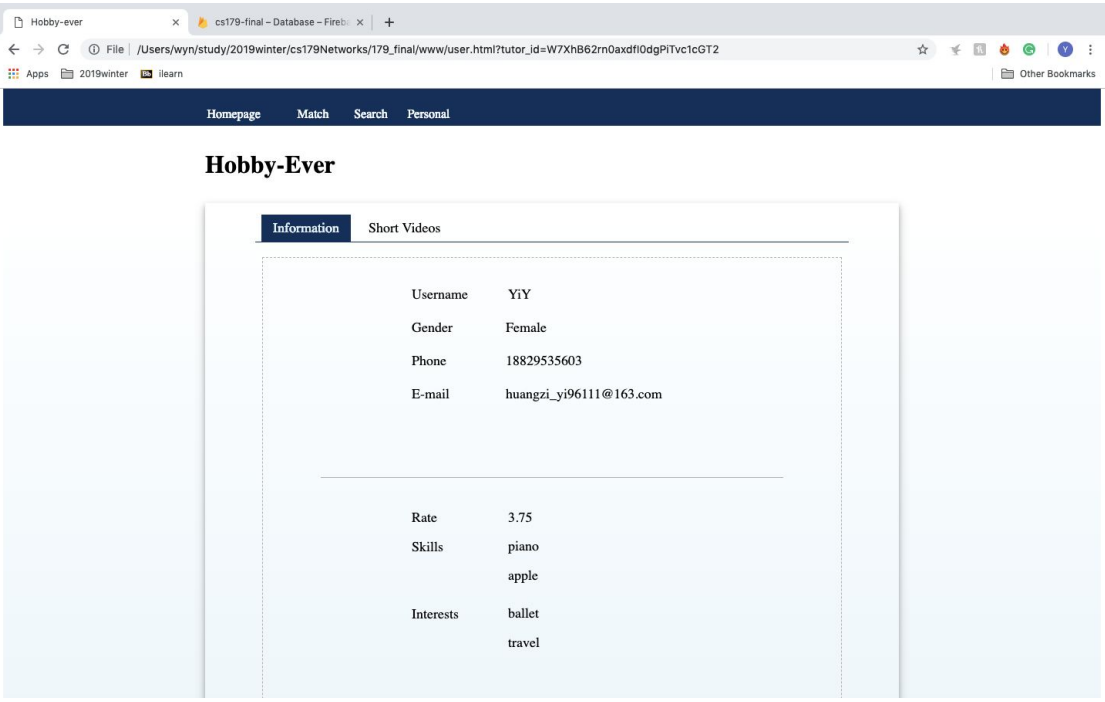
While jumping to the rate page, we will transmit the paired-user's uid, username, and course subject and course date from the previous page. Then display the later three data in the rate page and us paired-user's uid to find his/her collection in the database when updating his/her rate.



3.6 Paired-User's profile

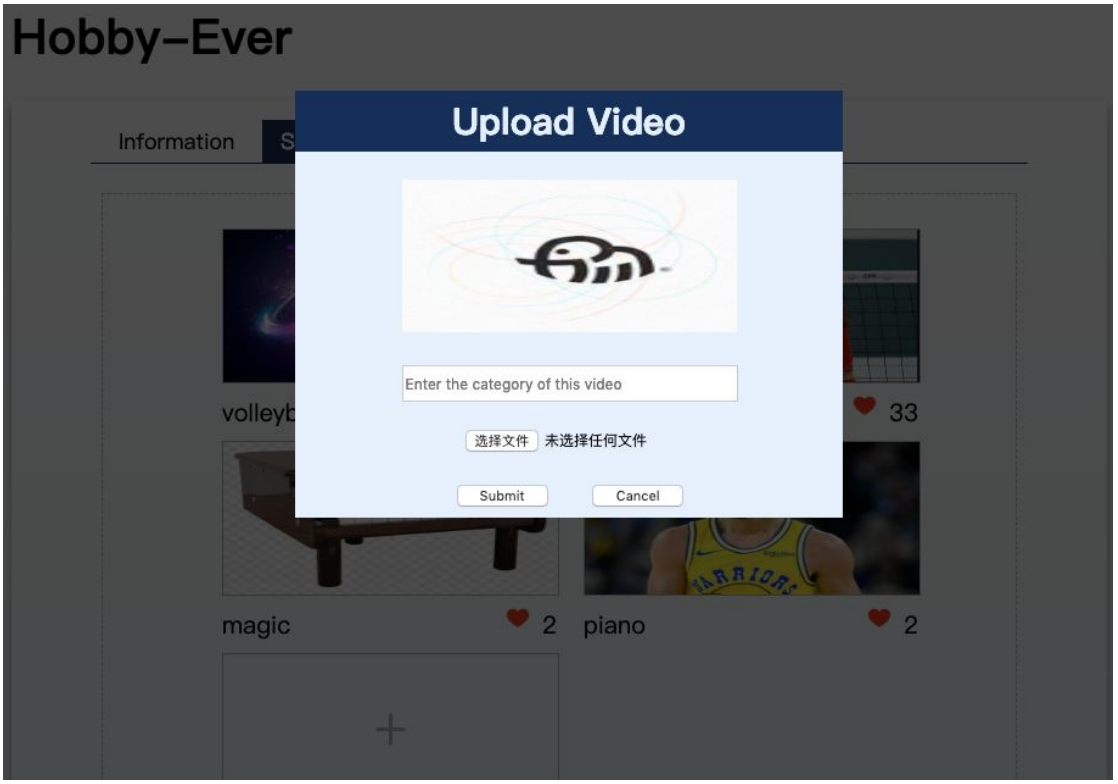
While users want to go to their paired-users' profile page, it will be different from what the paired-users themselves can see. This is because we will not allow other users to change your information or see your private calendar details.

To realize this, we only need to set the calendar tab and course tab's display style as none and change the text type of information field.

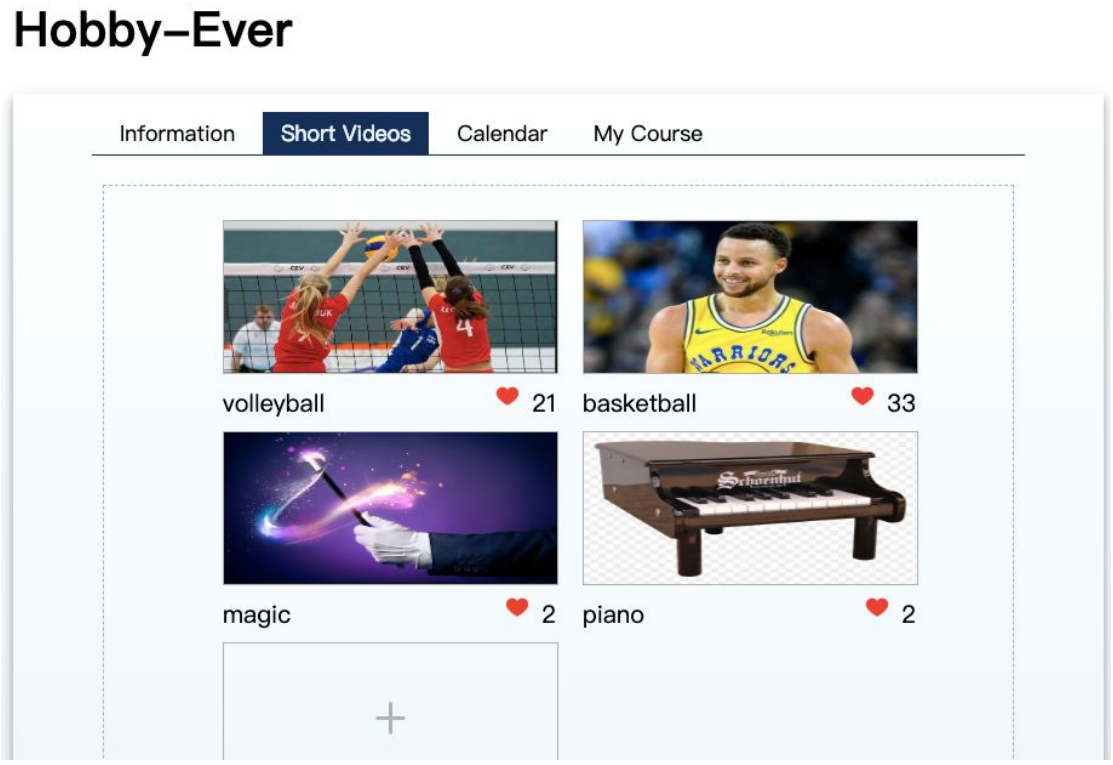


3.7 Short Videos

When the users click the upload module, the users are able to upload the videos from their own computer and input the categories of videos in order to help us sort the videos. And after that, the system would store such videos on Firebase according to a category.

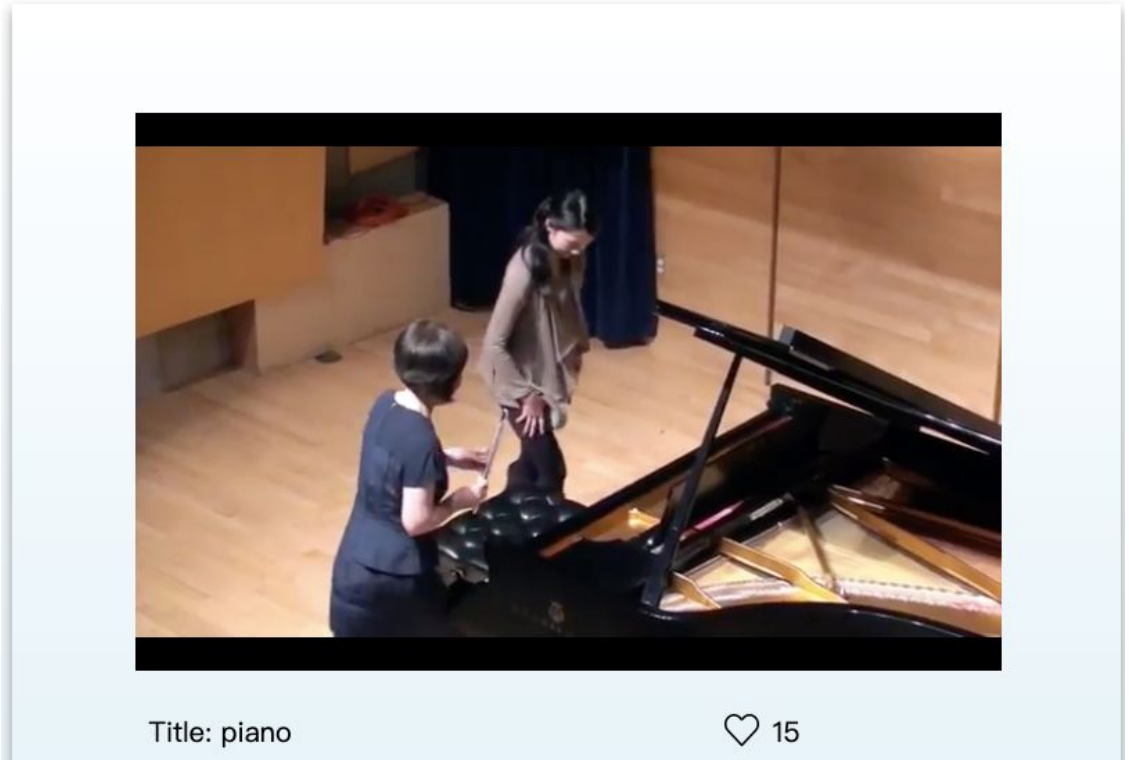


Each user can see all the short videos he/her uploaded in his/her personal homepage:



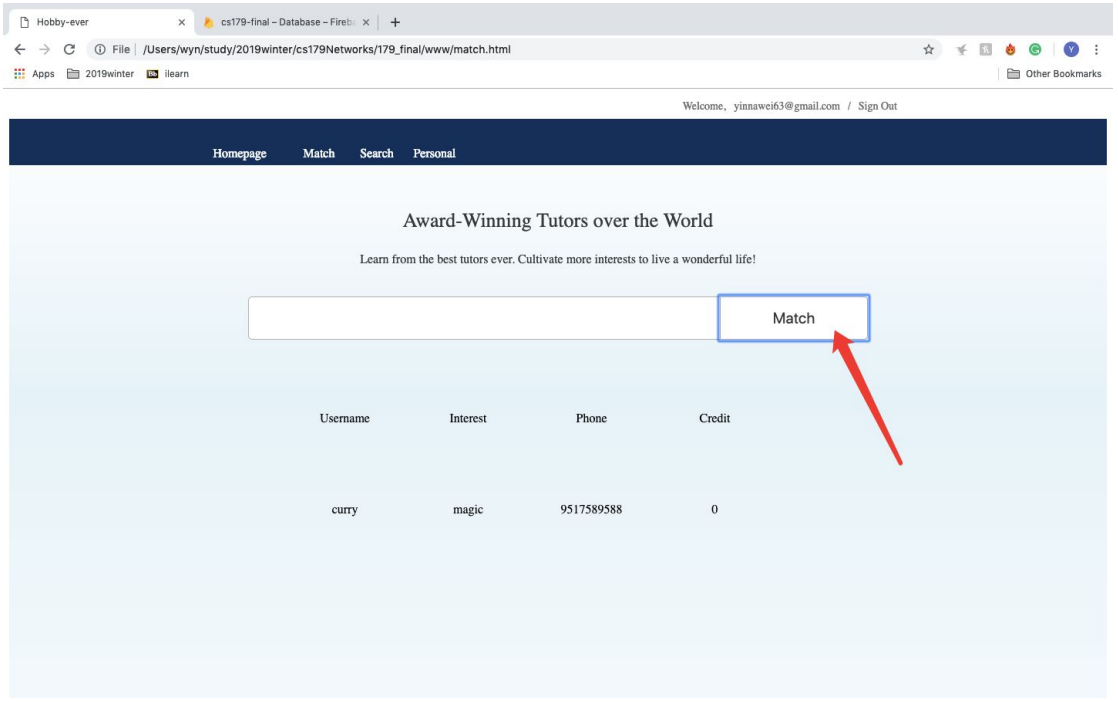
3.6 Video Playing

In the personal page, users can display their videos in their video-pages. The platform would automatically extract the file source from firebase according to the uid of users and display them in order. Other users would also visit such pages and click the likes if they are attracted by the talents of the partners. And the number of likes could be stored in the databases indicating the popularity of those users.

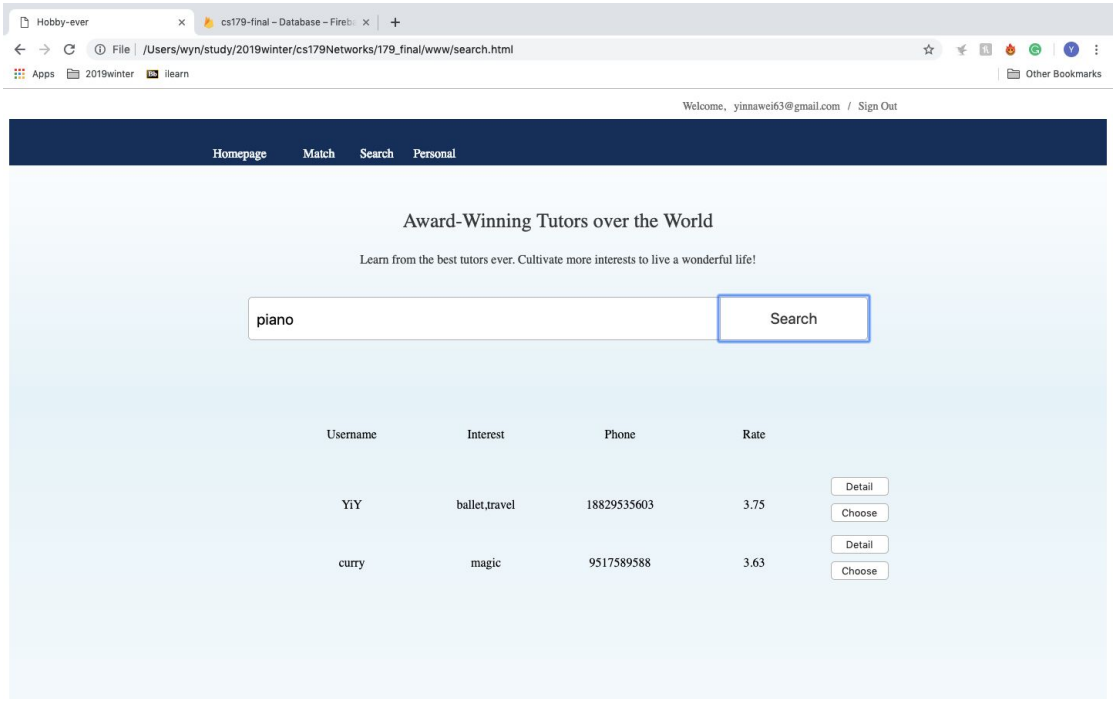


3.7 Search and Match

For matching function, after clicking the match button, based on information in the personal profile, the platform could automatically extract the information of users' interests. Then the platform could return the useful information of targeted users. Besides, two buttons could appear at the same time, choose button would help users choose the courses. And detail button would assist users to check the information in detail, such as time availability of their potential partners.



And in searching, users need to input some keyword first, then similarly, the system will use those keywords to find appropriate courses and users.



3.8 Online Chatting

In this part, we will have a server to allow users to communicate with their paired-users. Firstly, while jumping to this page, we also transmit the current user's uid and paired-user's uid here. Then the server will obtain the current user's uid automatically and judge whether the paired-user is online or not.

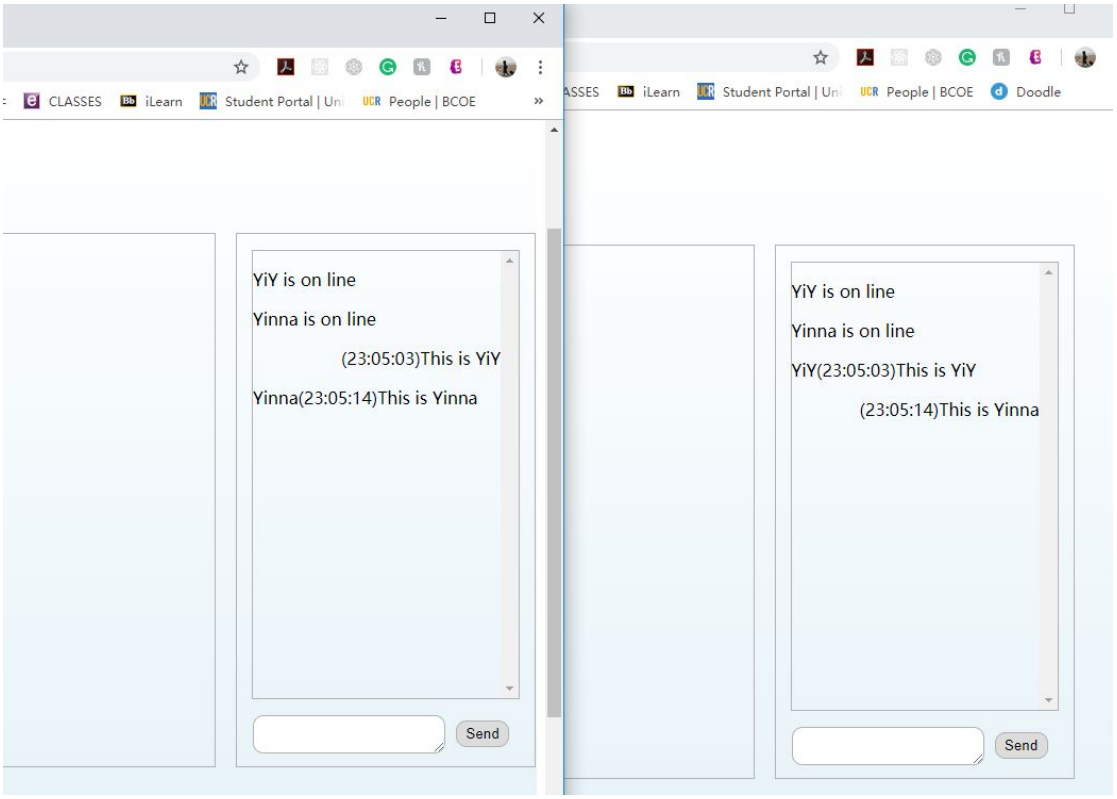
We mainly use node. js, express, and socket.io to implement this part. Node.js is used to set up the server; Express is a module in node.js that manages the routing and response of requests, it is used for the interactions between HTML, background, and clients.

The front-end logic consists of four parts: establishing a connection with the server, sending messages, receiving messages from other users, and receiving messages from the system. Background logic is mainly divided into establish a service, establish a connection with the front end, user login, accept the message sent by the user and broadcast this message, and process the system message. WebSocket is used to establish a persistent connection between the client and the server. Using WebSocket also enables the server to proactively push messages to the client. The front-end and back-end communicate with each other mainly through sockets.emit() and sockets.on().

What we also want to achieve is one to one private chatting rooms and group chatting rooms, the core of the implementation method is to store each user's socket id. If the current user wants to chat with someone privately, the id of the other party and the message need to be sent will be sent to the server. After receiving the message, the server will find the corresponding socket id of that recipient, and forward the message to that socket, then the message will be displayed in the interface of that particular user.

However, We met with some difficulties in the implementation of this part: when a private chat is processed, every time after the sender sends a message, although the other party can receive the messages correctly, the message sender will automatically disconnect with the server and then establish a connection again, which causes the sender repeatedly landed and disconnected. We suspect that the reason for this problem may lay in the client code, maybe there is a lack of some mechanism to keep the connection in the private message sending module, but unfortunately, this problem has not been solved yet.

This module still has a lot of parts that can be improved, for example, we can rent a large cloud server to implement the online chat server, but not using the server on our own computer, in this way, the server can stay online for a long time, users can use our system at any time they want, at the same time, use the cloud server can also make our system more stable and scalable.



3.9 Road Map

1. Currently, our system can only match users based on their skills and interests.
   In the future, we will realize to match users based on their calendar availability as well.
2. Currently, we can only search by interests to find users who own this as a skill.
   In the future, we will also allow users to search by username or gender or some other detail information to make the result meet the user's requirements at most.
3. Currently, we only realize a chat-room for online chatting, which means all the users connect to the server will be able to chat together.
   In the future, we should have a private chatting, which will only happens between the current user and his/her paired-user, so the course could be more private and would not be influenced by others.
4. And we will also have real-time video chat in the future, this will help users watch how their paired-users behave more clearly.

Google Drive Link:
https://drive.google.com/open?id=1so6zEUssI6HlhSHTUKgXNr6tPTljo7cu