Eatogether Final Report

Member

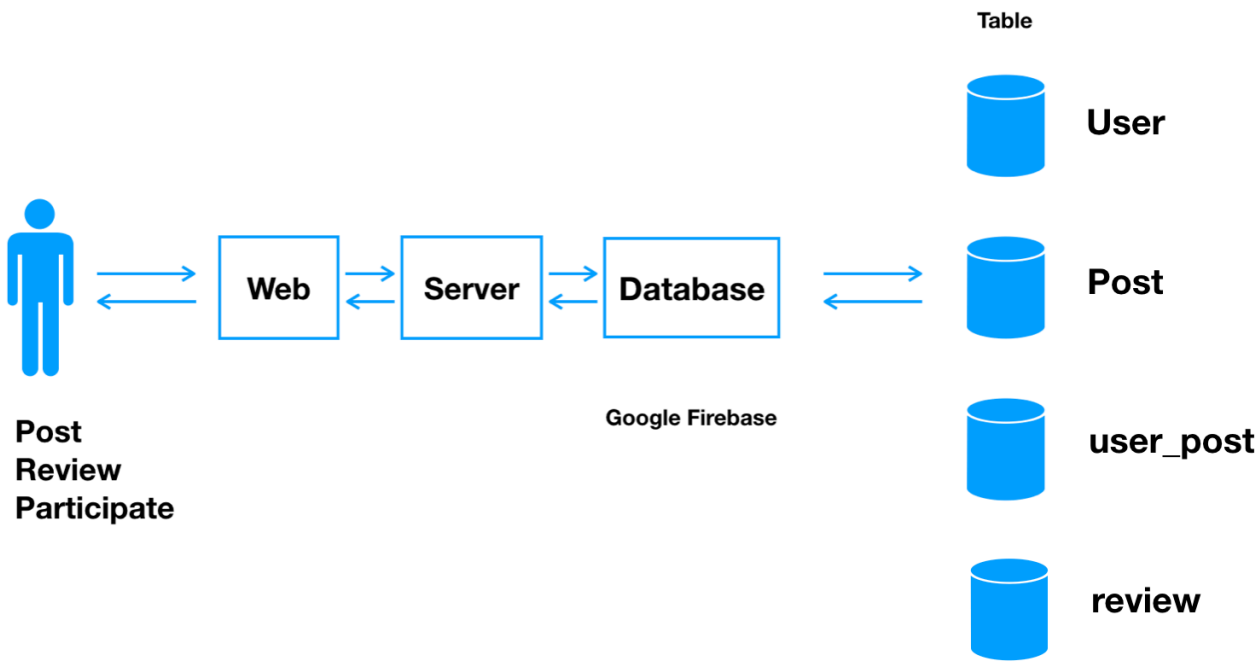
Anran Guo(USC ID: 8796-0029-64)
Lu Zhang(USC ID: 1778-7383-24)
Mi Youn Song(USC ID:8890-4523-05)

Project motivation and goal

The main idea of our project is to build a web-based online platform for student in USC to find a partner to have a meal together. Users can login with thier USC account, post their meal plan or join a meal initialed by others. To be more specificially, our project contains the following fucntions:

- Login & logout with USC gmail account
- Post meal plan
- Join a meal initialed by other user
- Get notification when someone join a meal
- View meal history
- Write review about experience

The project's front-end is implemented by javascript and html with stylesheet. To link with cloud database, google firebase, we use jacascript.The following diagram illustrates the architecture of the project



Architection of the application

Major Components

- Firebase:hold the record of our application
- Jacascrip:Implement the function in the web
- Html:User interaction
  *Server:home.ustc.edu.cn

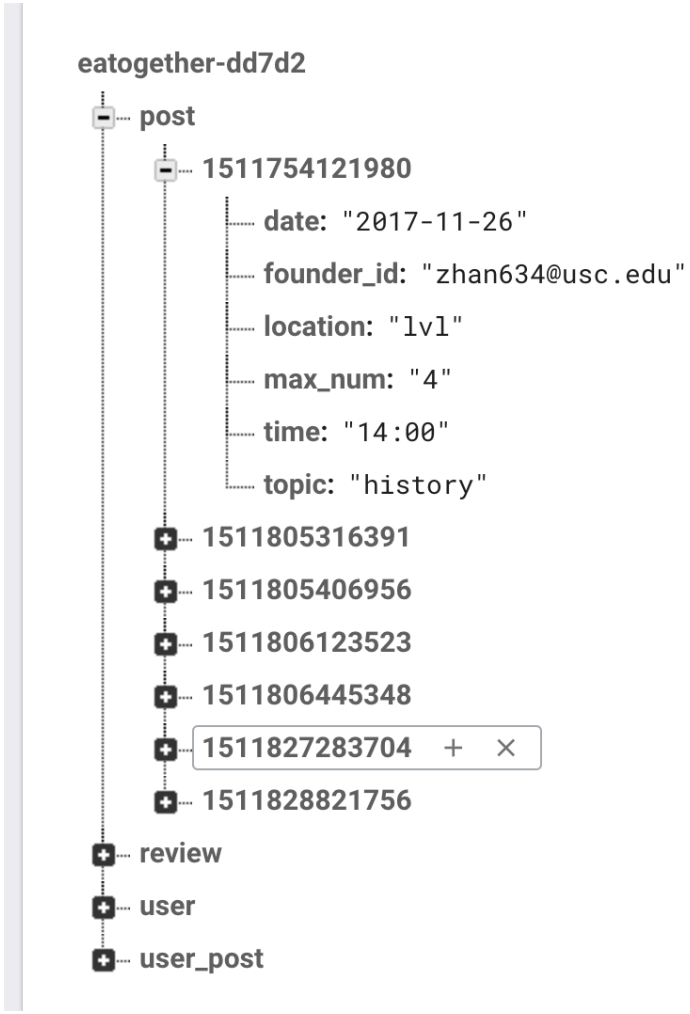
User Interface

- **login page**:use the usc email to login
- **post page**:launch a new post including the time ,location ,date
- **review page**: when an user participated in an activity,then the users can write the review to show their experience
- **history page**:this page shows the history of the user, which is a list of the activities.
- **Sign out**:the use can Sign out and go back to the index page

Data management

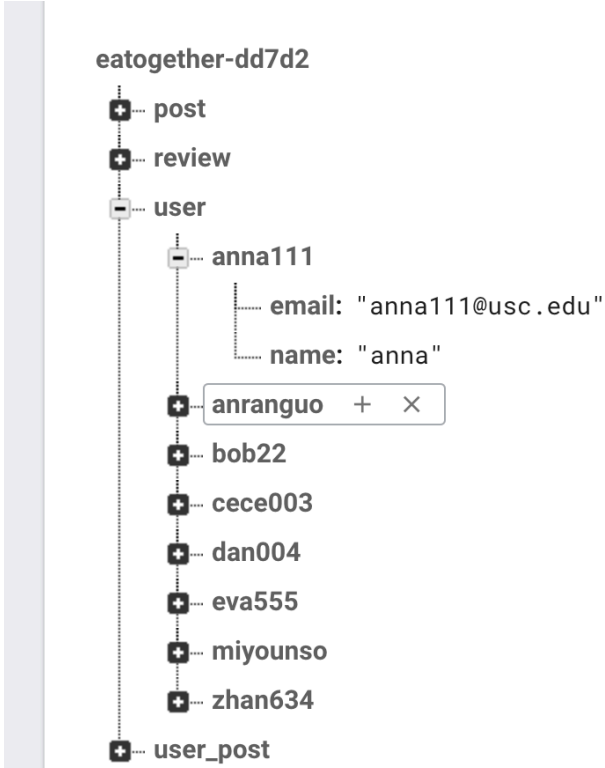
All our data is stored in the firebase. We use the retrieve function provided by the firebase to query.

Post table holds content of a post



Review table holds the content of a review

The user table get the userid and use's email and name from the google api.



Use_post contains all the users join in one activity

Description of implementation & structure of your code base of each function

Connect Firebase

Every page has code for connecting between Firebase and pages.

```
// Initialize Firebase
    var config = {
      apiKey: "AIzaSyCkSaYu8onQ5v6VioLcxJHGH3idSorGtu8",
      authDomain: "eatogether-dd7d2.firebaseapp.com",
      databaseURL: "https://eatogether-dd7d2.firebaseio.com",
      projectId: "eatogether-dd7d2",
      storageBucket: "",
      messagingSenderId: "210383482029"
    };
    firebase.initializeApp(config);
```

Navigation bar

Every page contains nav bar except landing page. Navigation bar enables users to switch between different functional pages. Besides, on the right side of the navigation bar, there is current user's name and sign out button. The current user name is obtained via session storage, which is created when user login via google api. Paramters will be pass to the next page, which includes name, user_id and etc.

When users clicks logout, they will be directed to landing page (with login).

Verify the user logging status

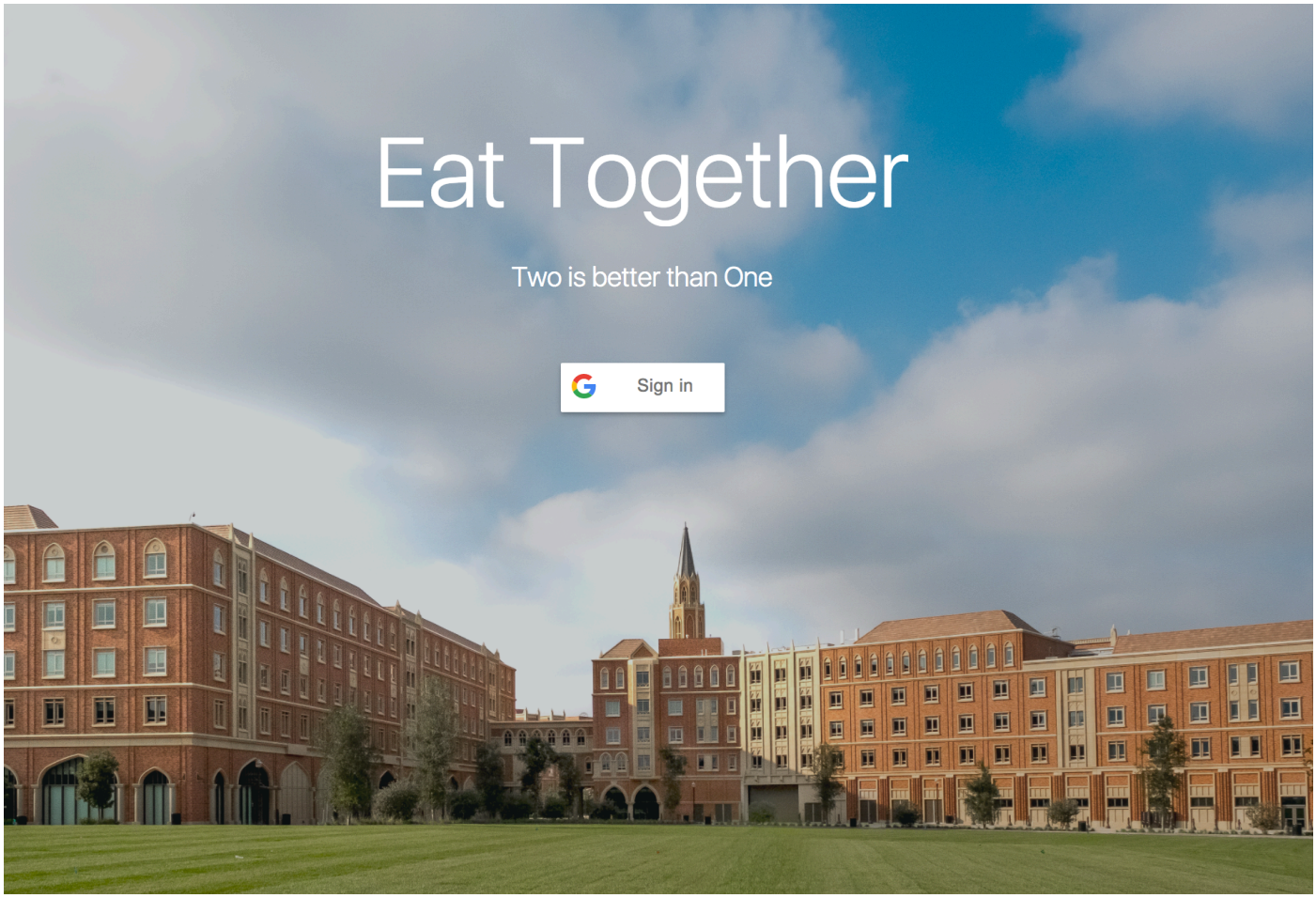
```
var userEntity = JSON.parse(sessionStorage.getItem('myUserEntity'));
    if (!userEntity){
      alert("not login");
       window.location.href="index.html"}
       else{
        console.log("Hi," + userEntity.Name);
        document.getElementById("username").innerHTML = "Hi," + userEntity.Name;
As we can see from the code, if the user hasn't logged in, the app will automatically jump
to the index page.
```

Sign Out Function

Click Sign Out Button in Navigation Bar, disconnect the user_id and clear the attribute.

```
function signOut() {
    console.log('User signed out.');
    //sessionStorage.removeItem('myUserEntity');
     sessionStorage.clear();
     localStorage.clear();
}
```

landing page and Log in



This page is the entrance to eatogether with log in function. Due to the concern of real world scenario, if we want to build a platform for USC student. Safety issue should be taken into consideration. Thus,the login gate only allow email account which ends with “@usc.edu” to enter. The functional part of login is implemented with google API.

```
function onSignIn(googleUser) {
```

```
  var profile = googleUser.getBasicProfile();
    console.log('ID: ' + profile.getId());
    console.log('Name: ' + profile.getName());
    console.log('Image URL: ' + profile.getImageUrl());
    console.log('Email: ' + profile.getEmail());
```

The code snippet above will prompt the google login window. Users can either login by enter their usc email and password through the entry of university or use existing cookie from browser. When the profile is obtained as the code snippet indicated, the current email account will be checked weather it ends with "@usc.edu". Therefore, normal gmail account will be prevent to enter eatogether with alert generate by javascript.

After successfully login, if it is the first time of the user to login, user data will be inseted into firebase. At the same time, user will be directed to post page(i.e. home page) Besides, session storage is created and passing the current user's information to home page via url (e.g. name, email...)

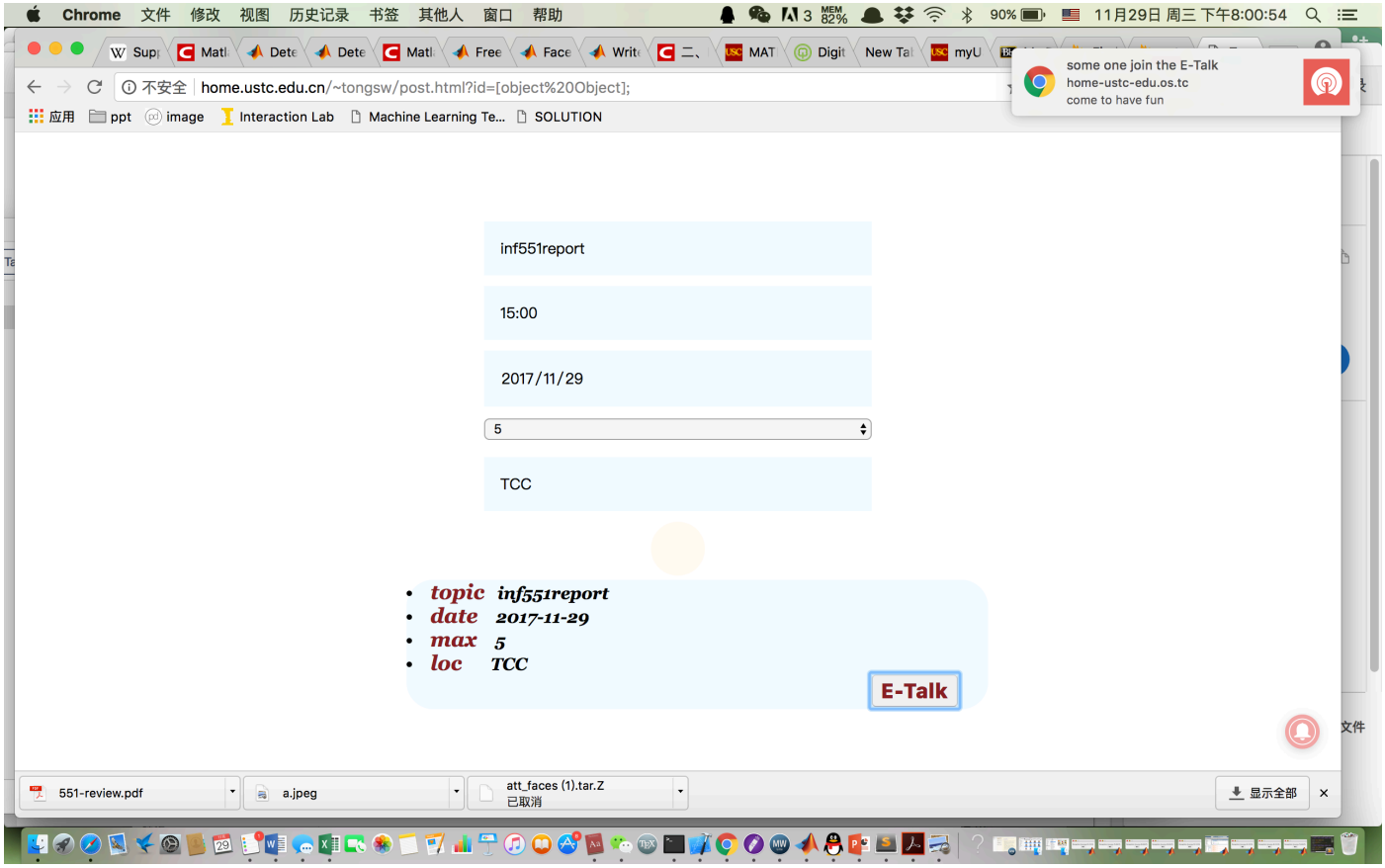
```
// creat session and store user entity
    myUserEntity = {};
    myUserEntity.userId = key;
    myUserEntity.Name = profile.getName();
    myUserEntity.email = profile.getEmail();
    sessionStorage.setItem('myUserEntity',JSON.stringify(myUserEntity));
    key = sessionStorage.getItem('myUserEntity');
    console.log(key);
    // direct to main page
    var url = "post.html?id="+myUserEntity+";";
     window.location.href = url;
```

Post Page

Get the data from the input and write data to firebase

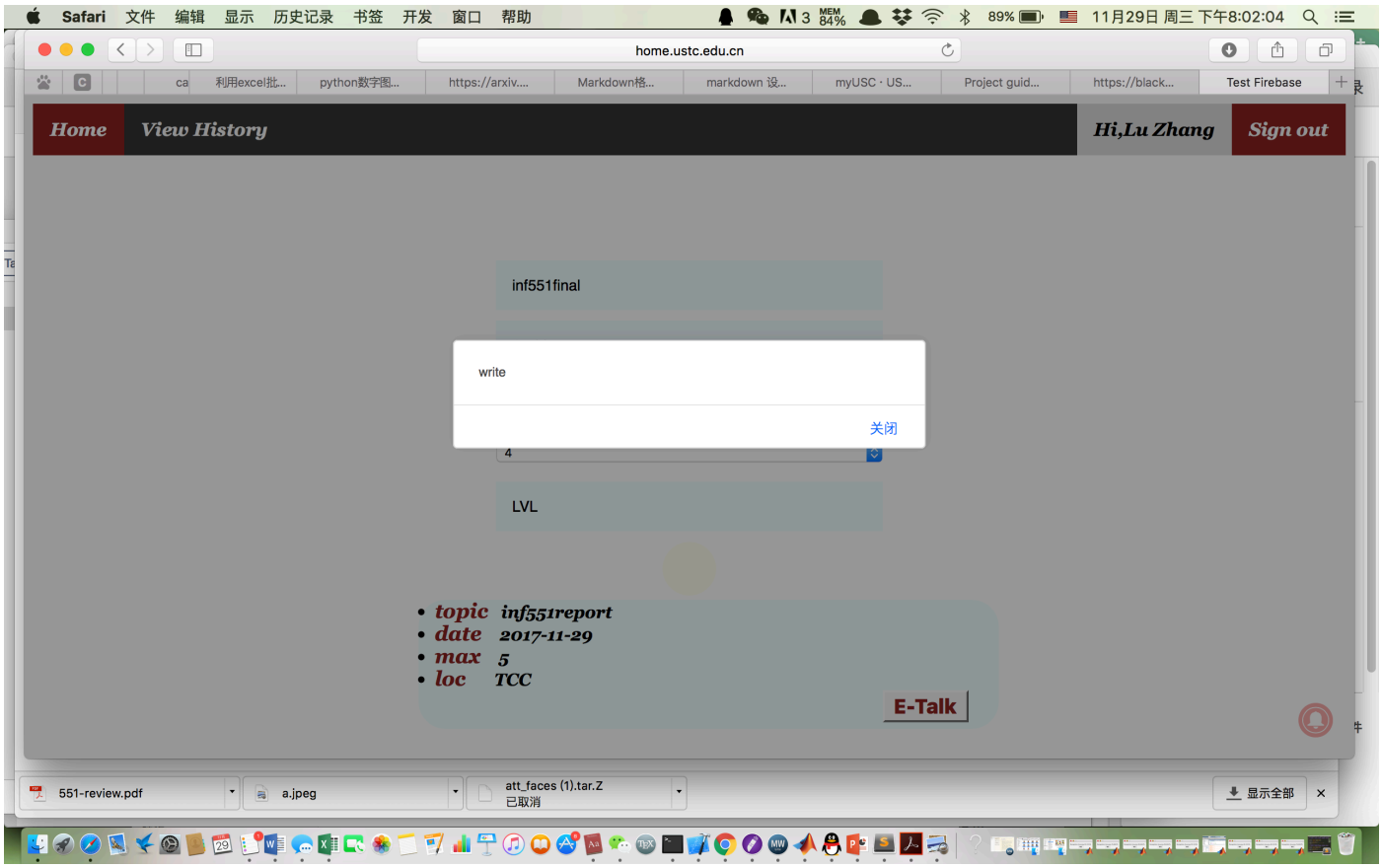
```
function writeUserData(){
    var userEntity = JSON.parse(sessionStorage.getItem('myUserEntity'));
    if (!userEntity){
     alert("not login");
      window.location.href="index.html"}
    var top=document.getElementById("topic").value;
    var t=document.getElementById("time").value;
    var dat=document.getElementById("date").value;
    var max=document.getElementById("max").value;
    var loc=document.getElementById("location").value;
  alert("write");
  var myDate= new Date();
  myDate.getTime
  var idtemp=myDate.getTime()
```

```
   firebase.database().ref("post/" + idtemp).set({
  topic:top,
  time: t,
  date: dat,
  max_num: max,
  location:loc,
  founder_id:userEntity.email,
  });
   firebase.database().ref().child("user_post/" + idtemp).once('value',function(snapshot){
    console.log(snapshot.numChildren());
    count=snapshot.numChildren();
    console.log([count]);
    console.log(userEntity.Name);
    var user_email=userEntity.email;
    var to_remove = userEntity.email.slice(user_email.length-8,user_email.length);
    var key = userEntity.email.slice(0,user_email.length - to_remove.length);
   firebase.database().ref("user_post/" + idtemp).update({
   [key]:userEntity.Name});
   });
}
<form  method="get">
        <div class="form-group">
          <input type="text" class="form-control" id="topic" placeholder="topic:music">
          <input type="text" class="form-control" id="time" placeholder="time:15:00-
15:30">
          <input type="date" class="form-control" id="date" placeholder="11/12">
           <select type="text" class="form-control" id="max" style="width:360px"
placeholder="3<number<6">
          <option>3 </option>
          <option>4 </option>
          <option>5 </option>
           </select>
          <p></p>
           <input  type="text" class="form-control" id="location"
placeholder="location:tcc">
        </div>
    <div class="submit-btn">
        <svg  onclick="writeUserData();notice();" width="50px" height="50px"
viewBox="487 475 50 50" version="1.1" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">

```



Write the join data

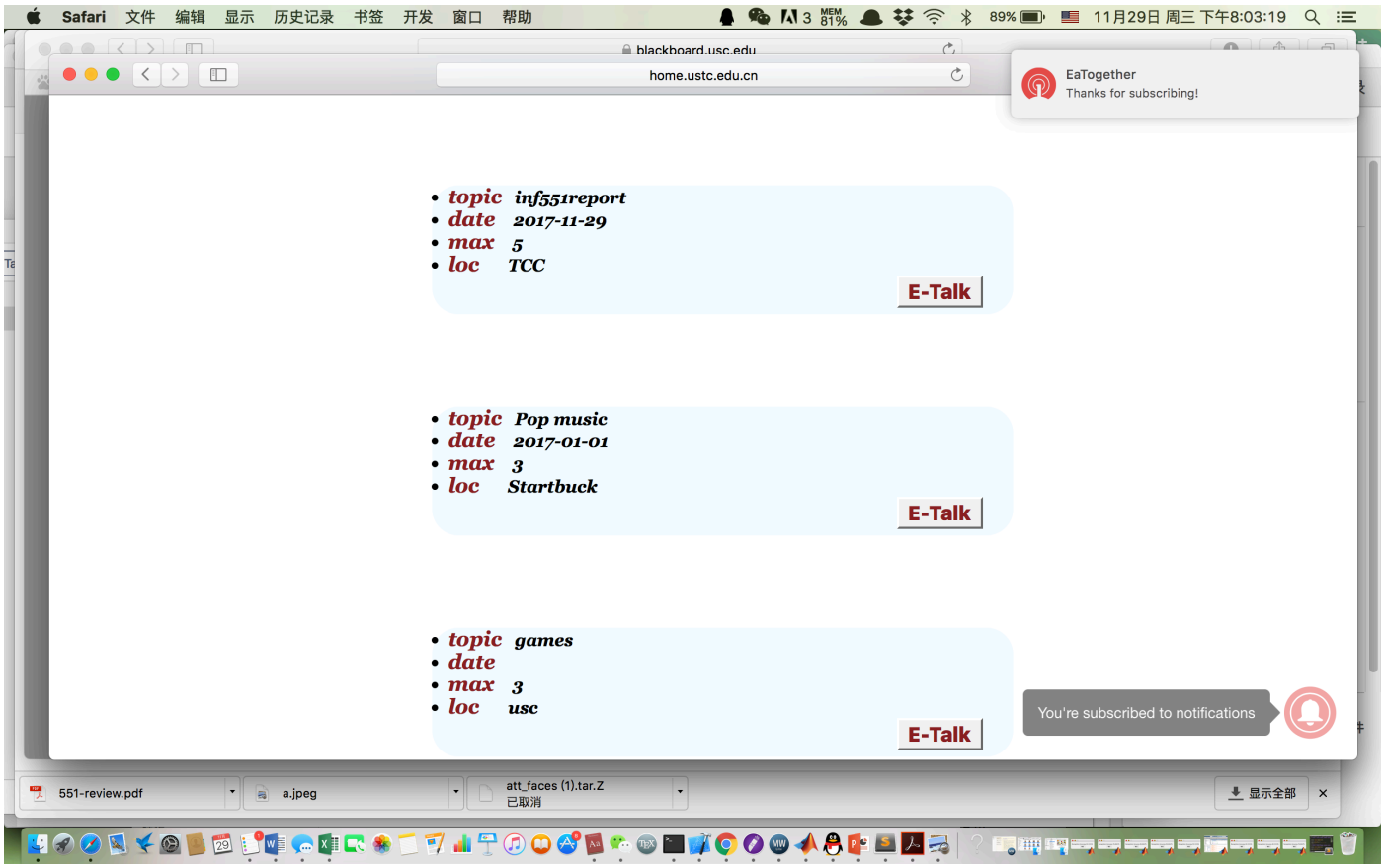
```
function writeJoin(click_id){
    alert(click_id);
    var myDate= new Date();
    myDate.getTime
    var count=0
   firebase.database().ref().child("user_post/" + click_id).once('value',function(snapshot){
    console.log(snapshot.numChildren());
    count=snapshot.numChildren();
    console.log([count]);
    console.log(userEntity.Name);
    var user_email=userEntity.email;
    var to_remove = userEntity.email.slice(user_email.length-8,user_email.length);
    var key = userEntity.email.slice(0,user_email.length - to_remove.length);
   firebase.database().ref("user_post/" + click_id).update({
   [key]:userEntity.Name});
   });
}

```



Notice function

```
function notice(){
OneSignal.push(function() {
  OneSignal.sendSelfNotification(
  /* Title (defaults if unset) */
  "some one join the E-Talk",
  /* Message (defaults if unset) */
  "come to have fun",
   /* URL (defaults if unset) */
  'http://home.ustc.edu.cn/~tongsw/eat.png',
  /* Icon */
  'https://onesignal.com/images/notification_logo.png',
  {
    /* Additional data hash */
    notificationType: 'news-feature'
  },
  [{ /* Buttons */
    /* Choose any unique identifier for your button. The ID of the clicked button is passed to you so you can identify which button is clicked */
    id: 'like-button',
    /* The text the button should display. Supports emojis. */
    text: 'Like',
```

```
    /* A valid publicly reachable URL to an icon. Keep this small because it's downloaded on each notification display. */
    icon: 'http://home.ustc.edu.cn/~tongsw/eat.png',
    /* The URL to open when this action button is clicked. See the sections below for special URLs that prevent opening any window. */
    url: 'https://example.com/?_osp=do_not_open'
  },
  {
    id: 'read-more-button',
    text: 'Read more',
    icon: 'http://i.imgur.com/MIxJp1L.png',
    url: 'https://example.com/?_osp=do_not_open'
  }]
);
});
}
```



Post function

```
dbRef.on('value',function(snapshot){
  $('#messagesDiv').html(" ");
  snapshot.forEach(function(childSnapshot){
    var childKey=childSnapshot.key;
```

```
    var childData=childSnapshot.val();

displayMessage(childData.topic,childData.date,childData.time,childData.max_num,childData.location);
    function displayMessage(topic,date,time,max,loc){
    $('#messagesDiv').prepend('<div class="showbox"><li><label class="showhighlight">topic</label>   '+topic+'</li>'+
  '<li><label class="showhighlight">date </label>   '+date+'</li>'+
  '<li><label class="showhighlight">max </label>   '+max+'</li>'+
  '<li><label class="showhighlight">loc   </label>   '+loc+'</li></ul>'+
  '<button type="button" onclick= "writeJoin(this.id);notice()" id='+childKey+' >E-Talk</button> </div> <p> </p><p> </p>');
    $('#messagesDiv')[0].scrollTop = $('#messagesDiv')[0].scrollHeight;
  }
  })
    });
    }
```

View_history Page

View-history page can be accessed by clicking navigation bar in home page. This function is design to view every meal user have participanted in before , which includes both meal initialed by user himself/herself or join in other meal. Additionally, in view history page, each hyperlink of meal history is also an entrance for user to leave review.

The mechanism of view-history page's implementation is consisting of two parts. Query the post information from the firebase and dynamically create div tags to load the post information of post in html code.

To begin with, the following chart present the structure of user_post table in firebase, which record post and its participants.

user_post:
|— post_id:
|— user_id:user_name
|— user_id:user_name

Firstly,as for obtain data from firebase, we are following the logic of the following pesuedo code:

```
for each (user_id,username) in user_post table"
  if user_id == current_user_id
    push post_id into array
    //then use the post_id to get more detail information of post in post table
```

And correspoding javascript can be refered below, we use firebase query to access database and check with deisred condition:

```
var ref = firebase.database().ref("user_post/");
ref.orderByChild('user_id').once('value').then(function (snapshot) {
        snapshot.forEach(function (childSnapshot) {
            if(curr_user_id in childSnapshot.val()){
              console.log(childSnapshot.val());
                post_info = {};
                post_info.post_id = childSnapshot.key;
                post_array.push(post_info);
            }
            })
```

Additionally, to load all the post information in the front-end, html code cannot follow the traditionaly way,. Rather, it need to be generated based on the length of the arry which store the query result. The function mentioned is implemented as folloing code snippet suggests:

```
    for (i = 0; i < post_array.length; i++) {
            var div = document.createElement("div");
            var h2 = document.createElement("h2");
            h2.id = "h2" + i
            var a = document.createElement("a");
            div.class = "history_meal";
            div.id = "div" + i;
            a.innerHTML = post_array[i].post_id+ ":"+  post_array[i].date;
            document.getElementById("main-content").appendChild(div);
            document.getElementById("div" + i).appendChild(h2);
            document.getElementById("h2" + i).appendChild(a);
        }
              })
```

The format of html generated:

```
<div id="div0">
    <h2 id = "h20"> post date1</h2>
</div>
<div id="div1">
    <h2 id = "h21"> post date2</h2>
</div>
```

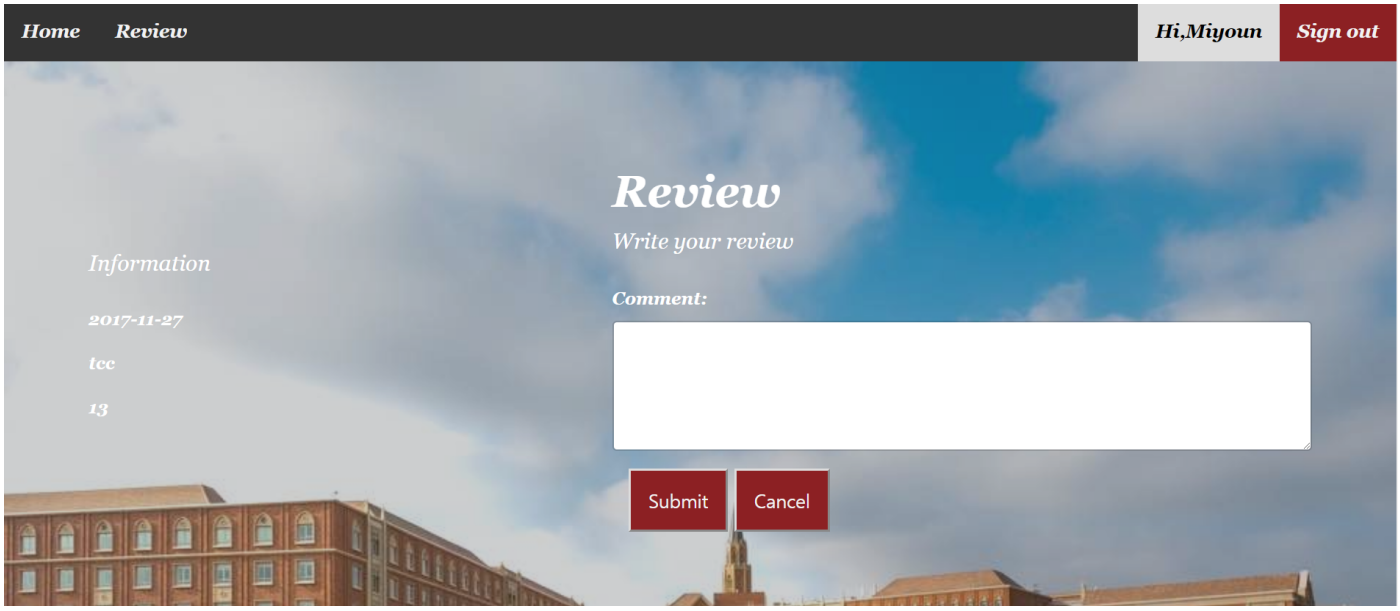
Besides, each div generated is a hyperlink, which will direct user to review page via the link. User is able to write review for this post. The paramter passing is different from other function. Since the tag in gtml is dynamically generated. We add the onclick function to get the innerhtml once clicking. The following code snippet shows how the onclick works:

```
// add this when generate html
a.onclick = createSession;
function createSession(){
    var tempid=this.innerHTML.split(":");
    myUserEntity.post_id = tempid[0];
```

```
    myUserEntity.curr_user_id = curr_user_id;
    myUserEntity.Name=userEntity.Name;
    //Store the entity object in sessionStorage where it will be accessible from all pages
of your site.
    sessionStorage.setItem('myUserEntity',JSON.stringify(myUserEntity));
    key = sessionStorage.getItem('myUserEntity');

}
```

Review

This page is for writing review after meeting. When user click their each history of post, page moves to review page.



When window moves from view_history.html to review.html, pass parameters of user information.

```
var userEntity = JSON.parse(sessionStorage.getItem('myUserEntity'));
    if (!userEntity){
    alert("not login");
     window.location.href="index.html"}
     else{
       console.log("Hi," + userEntity.Name);
       console.log(userEntity);
       document.getElementById("username").innerHTML = "Hi," + userEntity.Name;}
```

In left part of the page, there is an information about posting. A list of information is date, location and time. View_history.html pass parameter about user_id and post, so function find match information about post and print on the window.

```
function read_information(){
    //read firebase and show left side
```

```
    var partcpt_id_array = [];
    var partcpt_name_array = [];
    var ref = firebase.database().ref("post/");
    ref.orderByChild('user_id').once('value').then(function (snapshot) {
        snapshot.forEach(function (childSnapshot) {
            console.log(childSnapshot.key)
            //change 1 to the post_id pass from previous page.
                if (childSnapshot.key == userEntity.post_id) {
                    var date = childSnapshot.val().date;
                    var location = childSnapshot.val().location;
                    var time = childSnapshot.val().time;
                    document.getElementById("date").innerHTML = date;
                    document.getElementById("location").innerHTML = location;
                    document.getElementById("time").innerHTML = time;
                }//if
        }) // for each childsnapshot
    });
```

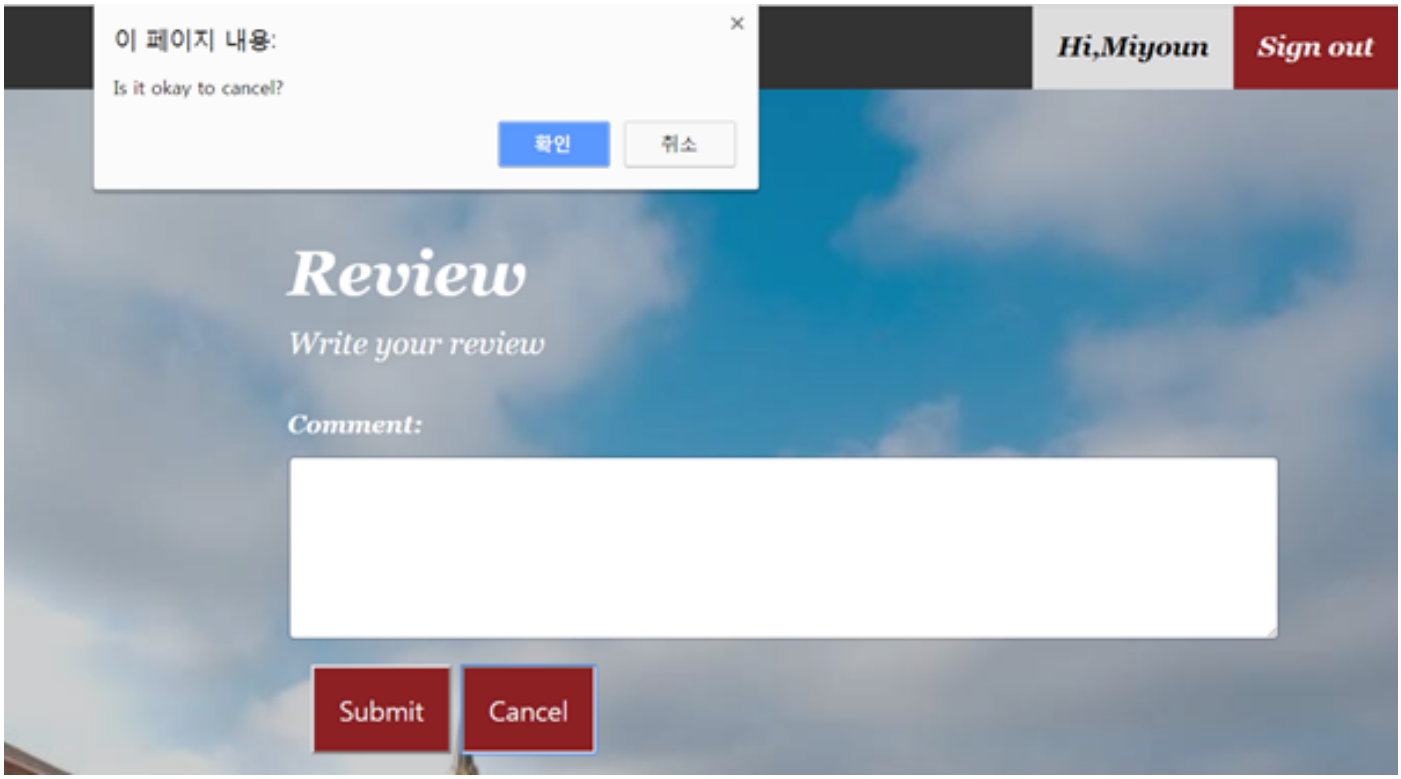
In right part of window, user can write the review. There is a text box for writing and are two kinds of button for submit and cancel. Submit button operate store data which contain post_id, writer_id, text to Firebase database in /review. Before storing, check length of text. If there is no data in text box, alert to “No text in review” and ask confirm to submit.

```
function writeNewPost(post_id, writer_id,review_text) {
  var text_id = document.getElementById("review_text");
  var text = text_id.value;
  //alert(text.length)
  if(text_id.value.length == 0){
    alert("No text in review");
    window.location.reload();
  }
  // A post entry.
  var postData = {
    post_id: post_id,
    writer_id: writer_id,
    text: text
  };
  //alert("okay");
  if(confirm("Is it okay to submit?"))
  {
    // Get a key for a new Post.
    var newPostKey = firebase.database().ref().child('review').push().key;

    // Write the new review.
    var updates = {};
    updates['/review/' + newPostKey] = postData;
    return firebase.database().ref().update(updates);
  }
}
```

-L-831FVTrgZ3pjeiYeX

- post_id: "151180644534
- text: "I was goo
- writer_id: "miyouns

cancel button is refresh the page. For decreasing fault, ask one more time to cancel




```
function cancel_refresh(){
  //when click the cancel button, check one more
  if(confirm("Is it okay to cancel?"))
  {
    window.location.href="review.html";
  }
}
```

Review_history

This page is user can see other participants' reviews about same post.

This page prints only reviews that have same post_id, therefore every childSnapshot check post_id. If childSnapshot have same post_id, they put the array for print.

```
var post_id_array = [];
var text_array = [];
var writer_id_array = [];
var ref = firebase.database().ref("review/");
ref.orderByChild('review_id').once('value').then(function (snapshot) {
    snapshot.forEach(function (childSnapshot) {
      //alert("here"+childSnapshot.val());
      console.log(childSnapshot.val());
      if( userEntity.post_id == childSnapshot.val().post_id){
        console.log("in");
        //alert("here");
        //alert(userEntity.post_id);
        //post_id_array.push(childSnapshot.key);
        post_id_tem = childSnapshot.val().post_id;
        text_tem = childSnapshot.val().text;
        writer_id_tem = childSnapshot.val().writer_id;
        post_id_array.push(post_id_tem);
        text_array.push(text_tem);
        writer_id_array.push(writer_id_tem);
      }
    }
```

Because it doesn't know how many reviews in Firebase, we have to make dynamic list for printing. While program run post_id_array times, every element put innerHTML and print it.

```
//++connect above code
for (i = 0; i < post_id_array.length; i++) {
  var div = document.createElement("div");
  var h2 = document.createElement("h2");
```

```
  var a = document.createElement("a");
  h2.id = "h2" + i;
  div.style.width = "100%";
  div.style.height = "100px";
  div.style.color = "black";
  div.class = "review";
  div.id = "div" + i;
  div.style.position = "relative";
  div.style.left =  "500px";
  div.style.top =  "120px";
  a.style.color = "black";
  a.innerHTML = "post: "+post_id_array[i]+"writer: "+writer_id_array[i]+" review: "+text_array[i];
  document.getElementById("main-content").appendChild(div);
  document.getElementById("div" + i).appendChild(h2);
  document.getElementById("h2" + i).appendChild(a);
  }
});
```

Advantage and disadvantage of cloud database

- Advantage
  - Accessibility: Stored files can be accessed from anywhere via Internet connection
  - Flexibility: NoSQL
  - Fast time to development: Do not need to spend time for build structure of Database
  - Less spend Money: in case of small company like start up, they can service without data center(infra structure)
- Disadvantage
  - Security concern
  - Not available once without internet connection
  - Expensive cost when they exceed their basic usage

Contribution of Project

- Design Project - ALL
- Design Data base - ALL
- Make pages
  - Log in(index.html) - Anran Guo
    - Login Function using Google API
  - Post(post.html) - Lu Zhang
    - Write post Function
    - Join other's post
  - View_history(view_history.html) - Anran Guo
    - Print a list of individual history of join posts

    - Writing_review(review.html) - Miyoun Song
        - Write review to each post
    - Review_history (review_history.html) - Miyoun Song
        - Print a list of reviews that have a same post_id
- Test and Prepare Demonstration - ALL
- Writing Report - ALL

Reference

[1] **Firebase documents** https://firebase.google.com/docs/database/web/read-and-write

[2] **Javascript** https://www.javascript.com/

[3] **htmlcss** https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input

[4] **json** https://www.w3schools.com/js/js_json_intro.asp