

# oingo

---

Yiang Zeng, yz4420, N15860799

Yue Zhao, yz5447, N10247509

## introduction

---

The 'oingo' is an app that allows users to share useful information via devices based on social, geographic, temporal, and keyword constraints. Users can publish information in the form of short notes, receive notes based on their own status, set filters getting the notes they want, and add friends with their account.

As an app, 'oingo' has the following functions: creating and editing account, writing notes, setting filters, making comments, request and receive friendship.

First, the app should have sign in and sign up page. New users can create a new account and old users can login to an exist account. And users can edit their profile so other people can know some basic information about them.

Second, users should be able to publish notes. The notes need to consist of a few words, and should associate with a location, radius of interest, a schedule, and some tags that specify the notes. The users can also decide whether others can make comments. Location and radius, schedule, tags are constraints which decide whether a filter can receive a certain note.

Third, users can set their filters to get the information they need. Users need to set the location and the radius where the filter is activated. And each filter has a schedule when the filter is activated. Then a state is needed so that when user is not in the certain status the filter will not activate. The function of the filter is to decrease the information users will get so users will not be bothered by mass of notes.

Finally, users can search other user and request friendship or receive friendship request. And users can make comment under the notes if notes' poster allows.

## Our main function of this project

---

1. Provide the login and sign up for new users.
2. On the main page, user can see the notes that other user post , and also have some button linked to other functions.
3. User can post the notes with the comment right
4. User can add some filters depend on their own needs.
5. When post notes, our project connect to GOOGLE MAP API to get the longitude, latitude, address and name of this location.
6. When add filters to filter the node with source(from friend only or from all) and also can connect to GOOGLE MAP API to get the longitude, latitude, address and name of this location.
7. User also can set the repeat time of each filter, such as 'remind use 3 days once' and 'each 2 weeks' .
8. User can send friendship request to other users; And also can receive and accept the friendship request to add a friend.
9. The "recent post" section will show the 8 most recent post ordered by post time.
10. The "search" section and "tag" section can help the user search for notes that they are interested in.

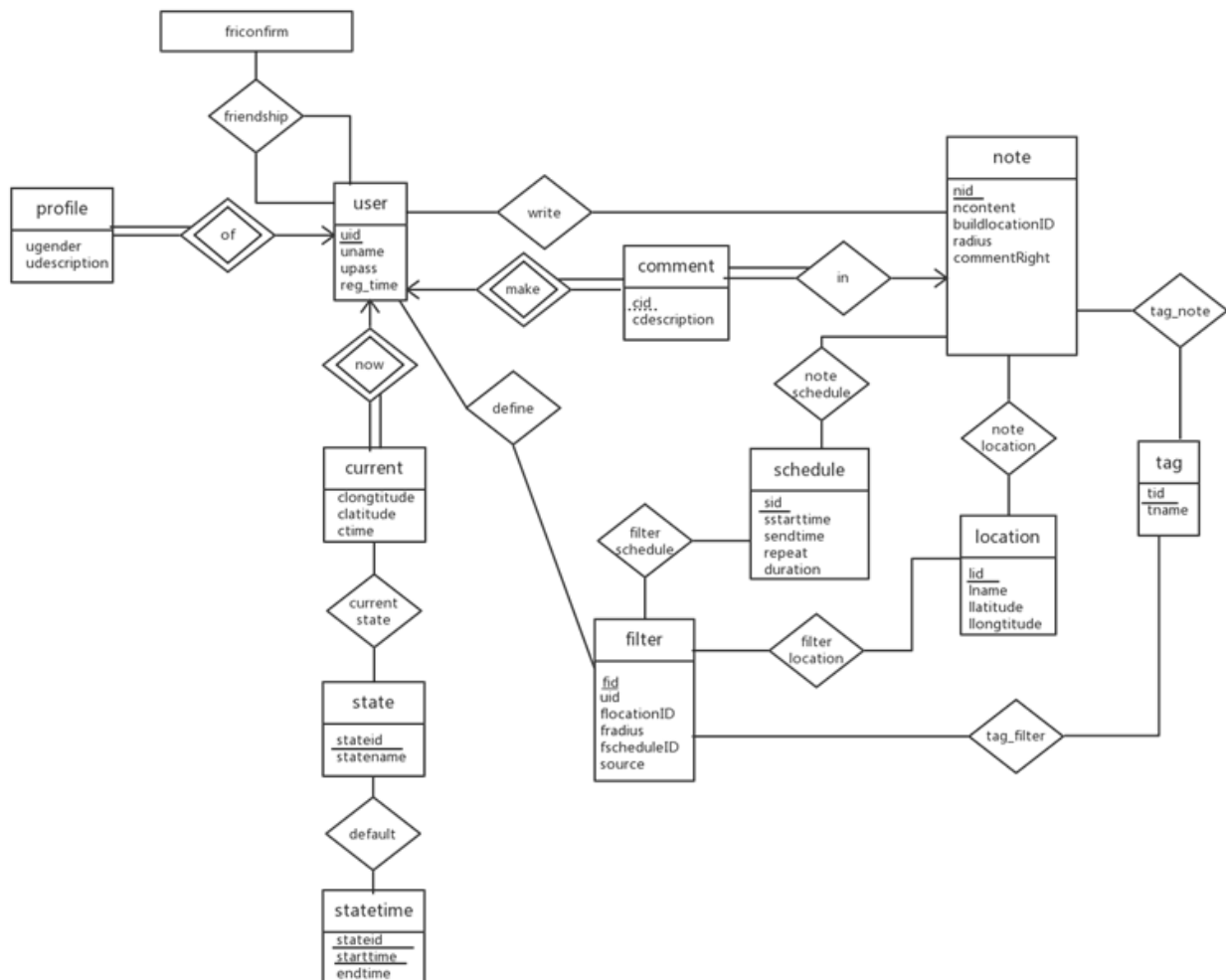
11. The “current” button can lead the user to a page that they can set their current location and time.
12. On the home page, we show the project detail of name, post time, tag, max fund and all the other details so that a user can see what are the other user doing.
13. The user can add comment on every single note !
14. We’ve implemented Security Functions to prevent SQL injections, Cross-Site Scripting and so on.

## Database design

### Explanations

To design the database of ‘oingo’, we considered the relationship between app and database and the frequency of changes of each table. Because every time users operate the app, like edit the filter. The tuple and attributes of some tables in the database will change. And we know that changing a large table may cost much time, so it is better to split these easily changing attributes to a individual table.

### ER-diagrams



### Relation Schema

**user** (uid, uemail, uname, upass, reg\_time)

**profile** (uid, ugender, udescription)

`schedules` (sid,sstarttime,sendtime,repeat,duration)

`location` (lid,lname,llatitude,llongitude)

`note` (nid,uid,ncontent,buildlocationid,radius,sid,commentRight)

`tag` (tid,tname)

`state` (stateid,statename)

`statetime` (stateid,starttime,endtime)

`current` (uid,clongitude,clatitude,ctime,ctime,ctime,ctime)

`filter` (fid,uid,flocationid,fradius,fscheduleID,stateid,sourse)

`tag-note` (tid,nid)

`tag-filter` (tid,fid)

`friendship` (uid1,uid2,friconfirm1,friconfirm2)

`comment` (cid,uid,nid,cdescription)

## Foreign key

Profile.uid is the foreign key reference to user.uid.

Note.uid is the foreign key reference to user.uid.

Note.buildlocationid is the foreign key reference to location.lid.

Note.sid is the foreign key reference to schedules.sid.

Statetime.stateid is the foreign key reference to state.stateid.

Current.uid is the foreign key reference to user.uid.

Current.stateid is the foreign key reference to state.stateid.

Filter.uid is the foreign key reference to user.uid.

Filter.stateid is the foreign key reference to state.stateid.

Filter.flocationID is the foreign key reference to location.lid.

Filter.fscheduleID is the foreign key reference to schedules.sid.

Tag-note.tid is the foreign key reference to tag.tid.

Tag-note.nid is the foreign key reference to note.nid.

Tag-filter.tid is the foreign key reference to tag.tid.

Tag-filter.fid is the foreign key reference to filter.fid.

Friendship.uid1 is the foreign key reference to user.uid.

Friendship.uid2 is the foreign key reference to user.uid.

Comment.uid is the foreign key reference to user.uid.

Comment.nid is the foreign key reference to note.nid.

## Description of tables

### User & profile & friendship & current & statetime

The above five tables all related to users themselves. User insert a new tuple when he sign up, the table `user` will record the user name and password user enter in the website, the `reg_time` from the system, and generate the user id by system.

The `profile` table record basic information user want to provide, user can edit information every time.

Table `friendship` links two user id and use `friconfirm` attribute to show whether it is a request or receiver confirm the request.

Table `current` record the users' current time, location, and state. When app activate, the tuple of the user will update every 5 minutes.

The state of a user can change automatically or manually. Table `statetime` is the schedule how state changes automatically.

### filter & note & comment

The above three tables record filter user design, the notes user post, and the comments user make. Table `note` has an attribute to activate comment.

### state & tag

Table `state` and `tag` are two individual table which store the key words using for matching. When user current state is different from the state set in a filter, the filter will not activate. And the function of `tag` is to match filter and notes. Only at least one tag in filter and note is the same filter will let the note pass. Both 'state' and `tag` can be defined by users. While user is trying to create a new tag or state, it will first scan the table to see whether it already exist. If not, we insert a new one.

### schedule & location

Table `schedule` and `location` is used to match different filter and notes. And is a parameter to decide whether a filter should activate. In location we use latitude and longitude to calculate distance. With the same latitude, every 0.00001 degree equals 1 meter. With the same longitude, every 0.00001 degree equals 1.1 meter. We may use more accurate method later. For `schedule`, if the note schedule and filter schedule have overlap, it means the note can pass the filter.

## We advance our database in following aspects

### 1. friendship table

Last version of our database, we use one attribute to clarify if two users are in a friendship: if it is 0, means user A has send a friendship request to user B; if it is 1, means the user who is request is consent to make friend with user A.

This version of database, we add one more attribute –that is 2 attribute to clarify which one is the user who request a relationship.

## 2. User table

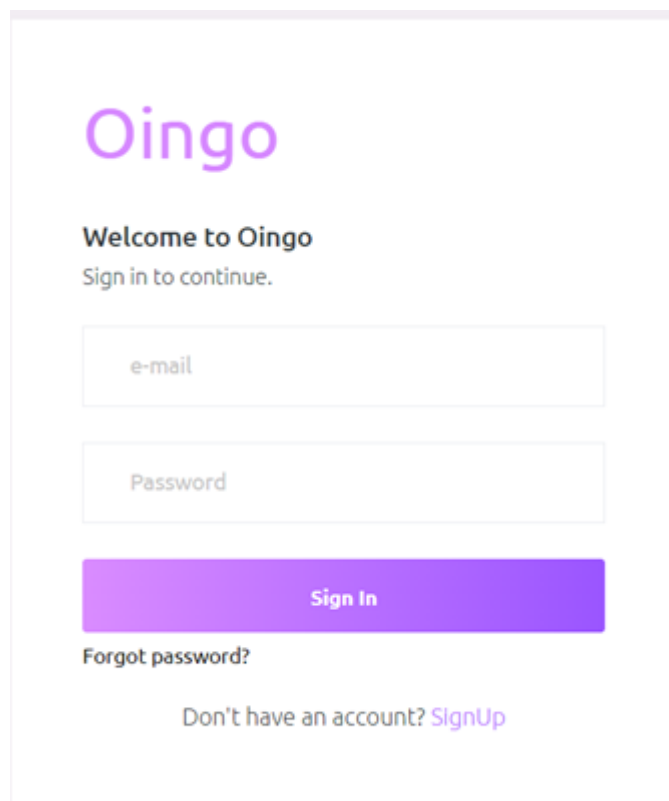
Last version, we only use a auto-increased uid as a candidate key. This version, we all a new attribute email —people can use email to login, unlike last version, they are required to remember their own uid, because it is the only candidate key.

###

## Function design and implement

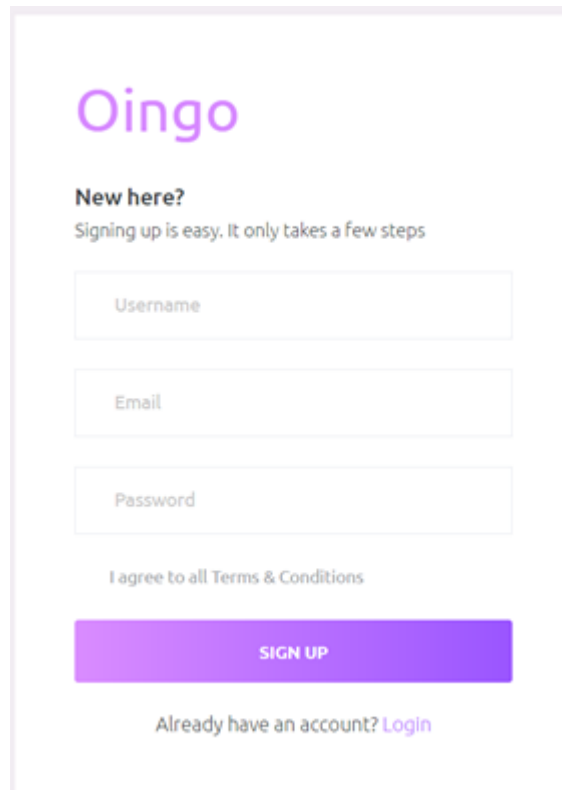
---

### 1. User login and sign up



The image shows a login and sign-up form for a service named "Oingo". The form is contained within a light gray border. At the top, the word "Oingo" is displayed in a large, purple, sans-serif font. Below the logo, the text "Welcome to Oingo" is written in a bold, dark gray font, followed by "Sign in to continue." in a smaller, regular dark gray font. There are two input fields: the first is labeled "e-mail" and the second is labeled "Password", both in a light gray font. Below these fields is a prominent purple button with the text "Sign In" in white. Under the button, the text "Forgot password?" is displayed in a dark gray font. At the bottom, the text "Don't have an account? SignUp" is shown, with "SignUp" in a purple font matching the logo.

The login page provides a window for users to login to the index page, they need to register before logging into the index.

A sign-up form for 'Oingo'. The form is centered on a light gray background. At the top, the 'Oingo' logo is in a purple font. Below it, the text 'New here?' is in bold, followed by 'Signing up is easy. It only takes a few steps'. There are three input fields: 'Username', 'Email', and 'Password', each with a light gray border and placeholder text. Below the 'Password' field is a checkbox labeled 'I agree to all Terms & Conditions'. At the bottom is a purple 'SIGN UP' button. Below the button is a link that says 'Already have an account? Login'.

In the design of sign-in and sign-up page, we use cookie to save the user's login status, the user logs in. After successful login, the server generates a specific cookie and returns it to the client. The client next accesses any page under the domain name, and sends the cookie information to the server. The server is verified. To determine if the user is logged in and Record the user's behavior.

when user try to type the url like <http://localhost/phpmyadmin/www.demo.com/pages/Home.php>, our website will judge whether they have already signed in or not, if not, it will jump to the login page directly.

We also prevent malicious injection by strictly limiting input characters.

```
<form class="pt-3" action="signincheck.php" method="post">
  <div class="form-group">
    <input type="email" class="form-control form-control-lg" name="email"
placeholder="e-mail">
  </div>
  <div class="form-group">
    <input type="password" maxLength="6"
onKeyUp="this.value=this.value.replace(/\\D/g, '')" class="form-control form-control-lg"
name="password" placeholder="Password">
  </div>
```

## 2.home page

When user login successfully, the website will jump to the home page directly, the home.php looks like the picture below:

Poster	nid	Location	Content	Start Time	End Time	Repeat
The Mermaid Inn	1	96 2nd Avenue, New York	Nice weather!	2018-11-25 12:00:00	2018-11-25 13:00:00	everyday
The Mermaid Inn	3	96 2nd Avenue, New York	So many prople in Central Park!	2018-11-25 12:00:00	2018-11-25 13:00:00	everyday
The Mermaid Inn	5	96 2nd Avenue, New York	Colorful Mandarin Duck Excites New Yorkers!	2018-11-25 12:00:00	2018-11-25 13:00:00	everyday
The Mermaid Inn	2	96 2nd Avenue, New York	A big tree here!	2018-11-25 12:00:00	2018-11-25 13:00:00	everyday
Poster	nid	Location	Content	Start Time	End Time	Repeat

In this picture, user can see that their are many buttons to jump the link to different php pages. In this page, user can change their states from sleep, breakfast, at work, lunch, dinner, play and also show them the notes other users post. And user also can click the button of nid to jump to the link of comment.php to add comment on this note.

To show all the details of the node, which effect many tables. Show the filters filtered by the notes:

```
SELECT nid,ncontent,fullnote.sstarttime,fullnote.sendtime,lname,laddress,commentRight
FROM (SELECT fid,sstarttime,sendtime,llatitude,llongitude
FROM current NATURAL JOIN ((filter JOIN schedules ON filter.fscheduleID=schedules.sid)
JOIN location ON location.lid=filter.flocationID)
WHERE POW((POW((clatitude-llatitude)*110000,2)+POW((clongitude-
llongitude)*100000,2)),-2) < `fradius` AND (`ctime` BETWEEN `sstarttime` AND
`sendtime`) AND current.uid = '.$user['uid'].' AND filter.uid= '.$user['uid'].' AS
activefilter,(SELECT *
FROM (note NATURAL JOIN schedules) JOIN location ON note.buildlocationid=location.lid
WHERE nid IN (SELECT nid FROM `tag_note` NATURAL JOIN (SELECT tid
FROM (SELECT fid,sstarttime,sendtime,llatitude,llongitude
FROM current NATURAL JOIN ((filter JOIN schedules ON filter.fscheduleID=schedules.sid)
JOIN location ON location.lid=filter.flocationID)
WHERE POW((POW((clatitude-llatitude)*110000,2)+POW((clongitude-
llongitude)*100000,2)),-2) < `fradius` AND (`ctime` BETWEEN `sstarttime` AND
`sendtime`))AS act_fil NATURAL JOIN `tag_filter`) AS hastag)) AS fullnote
WHERE POW((POW((activefilter.llatitude-
fullnote.llatitude)*100000,2)+POW((activefilter.llongitude-
fullnote.llongitude)*100000,2)),-2) < fullnote.radius AND ((activefilter.sstarttime
BETWEEN fullnote.sstarttime AND fullnote.sendtime) OR (activefilter.sendtime BETWEEN
fullnote.sstarttime AND fullnote.sendtime))
```

First, select the active filter under current conditions. And then Filter all the notes that contain the filter tag. Finally, filter the notes that meet the conditions based on the time and place.

### 3. Filter php:

#### 1. Insert schedules.

Check if tag is exist, if it is exist, fetch the tid of tag; if not, insert tag tuples into tag table. Tid of tag is auto-increased, therefore, the tid of this filter is the largest tid of the tag table.

Location of filter is chosen by flocationID, from the exist locationall table.

```
if ($_POST) {
    $sql1 = "INSERT INTO `schedules`(`starttime`, `sendtime`, `repeat`, `duration`)
VALUES ('".$_POST['starttime']."' , '".$_POST['endtime']."' , '".$_POST['Repeat']."' , '0')";
    $sql2 = "INSERT INTO `location`(`lname`, `laddress`, `llatitude`, `llongitude`)
VALUES
('".$_POST['Name']."' , '".$_POST['Address']."' , '".$_POST['latitude']."' , '".$_POST['longitude']."'
)";

    $DB->query($sql1);
    $sid = $DB->lastInsertId();
    $DB->query($sql2);
    $lid = $DB->lastInsertId();

    $sql3 = "INSERT INTO `filter`(`uid`, `flocationID`, `fradius`, `fscheduleID`,
`stateid`, `source`) VALUES
('".$_user['uid']."' , '".$_lid."' , '".$_POST['range']."' , '".$_sid."' , '".$_POST['State']."' , '".$_POST['S
ource']."' )";

    $DB->query($sql1);
    $DB->query($sql3);
    $fid = $DB->lastInsertId();

    $tagarr = explode(';', $_POST['Tag']);
    foreach ($tagarr as $v) {
        $row = $DB->query("select * from `tag` where `tname` = '{{$v}}' limit 1")-
>fetch(PDO::FETCH_ASSOC);
        if ($row) {
            $tid = $row['tid'];
        } else {
            $DB->query("INSERT INTO `tag`(`tname`) VALUES ('{{$v}}')");
            $tid = $DB->lastInsertId();
        }
        $DB->query("INSERT INTO `tag_filter`(`tid`, `fid`) VALUES ('$tid', '$fid')");
    }
    exit('<script>alert("添加成功! ");location.replace(document.referrer);</script>');
}
```

#### 2. After insert the filter to the filter table, our interface show all of the filter of this user.



To pop out the relevant notes and remind user: we need made a join among filter table, note table, current table, tag table, tag\_node table, and tag\_filter table. It looks like huge workload, but we can simplify it as far as we can:

a).Base on our current time and current location: we make a join between filter and current table, using the location and time.

b).To calculate the distance between the current location and location of filter, we define a function to do it.

The screenshot shows the Oingo application interface. The top navigation bar includes the Oingo logo, a search bar, and user profile information. The left sidebar contains navigation links: Home, Notes & Comments, Filter (active), Profile, and Friend.

The main content area displays the 'New Filter' form. It includes a map of New York City, a location input field with latitude and longitude coordinates, and a start time input field. Below these are fields for Name, Address, Tag, State, and Source. The form has 'SUBMIT' and 'CANCEL' buttons.

Below the form, there is a table showing the list of filters. The table has columns for Action, Filter, Location, Address, Start Time, End Time, and Repeat. The table shows two entries, each with a 'DELETE' button.

Action	Filter	Location	Address	Start Time	End Time	Repeat
<a href="#">DELETE</a>	1	Poco - NYC	33 Avenue B, New York	2018-11-25 12:00:00	2018-11-25 13:00:00	every day
<a href="#">DELETE</a>	9	emmm	Brooklyn Bridge New York, NY 10038	2018-12-16 12:00:00	2018-12-16 13:00:00	not repeat

Showing 1 to 2 of 2 entries [Previous](#) [Next](#)

Base on the filter survive, we can join filter survive with the note table, using location , time, tag.

```
select *,distance(c.longitude,c.latitude,v2.longitude,v2.latitude)
```

```
from current c,v2
```

```
where distance(c.longitude,c.latitude,v2.longitude,v2.latitude) < v2.fradius and  
c.ctime between v2.sstarttime and v2.sendtime;
```

## 4. Personal information

user can update their profile in the profile.php, which is shown as the picture below:

The screenshot shows the Oingo web application interface. On the left is a sidebar with navigation links: Home, Notes & Comments, Filter, Profile (highlighted), and Friend. The main content area displays the 'Account Information' form for user JXP (ID 10014). The form includes a 'Basic form elements' section with a 'Gender' dropdown (set to 'Female') and a 'Description' text area. At the bottom of the form are 'SUBMIT' and 'CANCEL' buttons. The footer contains the copyright notice: 'Copyright © 2018. Company name All rights reserved. 网页模板'.


You can choose the information that you want to update and the others will remain unchanged.


## 5. Post note




The basic goal of post.php is to insert a note tuple into the note table. When people post a note, it effect several tables : note table, locational table, schedule table, tag table, and tag\_note table.

To post note, user can click the note button to jump to the Note.php, which is shown below:

The screenshot shows the 'New Filter' form in the Oingo application. The sidebar is identical to the previous page. The main content area features a map of New York City at the top. Below the map are several input fields: 'Location' (with latitude and longitude sub-fields), 'Start Time' (2018-11-11 11:11), 'End Time' (2018-11-11 11:11), 'Repeat' (set to 'None'), 'Name', 'Address', 'Tag' (with a 'use"/>




breakfast

Tiger Nixon	21	ABC Kitchen	35 East 18th Street, New York	I love the store!	2018-11-30 12:00:00	2018-12-01 12:00:00
Tiger Nixon	24	The Mermaid Inn	96 2nd Avenue, New York	So many people in Central Park!	2018-11-25 12:00:00	2018-11-25 13:00:00
Tiger Nixon	42	ABC Kitchen	35 East 18th Street, New York	I love the store!	2018-11-30 12:00:00	2018-12-01 12:00:00
Tiger Nixon	43	Poco - NYC	33 Avenue B, New York	Celebrating the holiday season at the Oculus.	2018-11-25 12:00:00	2018-11-25 14:00:00
Tiger Nixon	52	Kumo Sushi	214 1st Avenue, New York	So far so good....	2018-11-25 12:00:00	2018-11-25 13:00:00
Tiger Nixon	56	The Mermaid Inn	96 2nd Avenue, New York	So many people in Central Park!	2018-11-25 12:00:00	2018-11-25 13:00:00
Tiger Nixon	74	ABC Kitchen	35 East 18th Street, New York	I love the store!	2018-11-30 12:00:00	2018-12-01 12:00:00
Tiger Nixon	75	Poco - NYC	33 Avenue B, New York	Celebrating the holiday season at the Oculus.	2018-11-25 12:00:00	2018-11-25 14:00:00
Tiger Nixon	84	Kumo Sushi	214 1st Avenue, New York	So far so good....	2018-11-25 12:00:00	2018-11-25 13:00:00

Showing 1 to 10 of 11 entries

Previous
Next

we can see that there are three main function in this page. First, we can post note with the information of location we point on the Google map. Second, user can see all of the notes he or she have post before, and to manager their notes, user can click the button with the uid of the note.

First, we insert a schedule tuple to the schedules table, every time people insert a note, we create a new schedule tuple and insert it into the schedules table, because the possibility that different user can make a same schedule is too small and it is also a waste of time to scan into the schedule check if there is a exactly same schedule in the schedule table. However, we can give a option to each user to choose the schedule they used before.

Because the sid is auto-increased so we many need to fetch the largest sid of the schedule, because the largest sid is the latest sid if user do not use their past schedule.

Second, we insert a new note into the note table. In the interface, user need to type in the time they want to be reminded, location , radius, comment right, also their content of the note.

## 6. Friendship request and accept

Our interface show three tables about the friendship. For example, when a user login the website, the interface shows his friend list, people who are not his/hers friend yet and user who send him a friendship request.

we can achieve 3 functions in this interface: send friend request to user who are not the user's friend yet, accept other user's friendship request and show the friendship list.



\$uname

\$uid

me

tes &amp; Comments

:er

ofile

end

## Friend Search

Friend Search

Userid

uid

SUBMIT

## Received Request

Horizontal form layout

Userid	Name	Confirm
--------	------	---------

Userid

Name

Microsoft Word

```

<?php
include './header.php';
if(isset($_GET['v'])){
    if($_GET['v'] == 'yes'){
        $DB->query("UPDATE `friendship` SET `friConfirm1`='1',`friconfirm2`='1' WHERE
uid1='{$_GET['uid1']}' and uid2 = '{$_GET['uid2']}'");
    }elseif($_GET['v'] == 'no'){
        $DB->query("delete from `friendship` WHERE uid1='{$_GET['uid1']}' and uid2 =
'{$_GET['uid2']}'");
    }
    exit('<script>alert("success! ~~");location.replace(document.referrer);</script>');
}
if($_POST['act']=='FriendSearch'){
    $sql="select * from user where uid = '{$_POST['uid']}' and uid !=
'{$_user['uid']}'";
    $row=$DB->query($sql)->fetch();
    if($row){
        $DB->query("INSERT INTO `friendship`(`uid1`, `uid2`, `friConfirm1`,
`friconfirm2`) VALUES ('{$_user['uid']}', '{$_POST['uid']}', '1', '0')");
        exit('<script>alert("send request successfully!
~~");location.replace(document.referrer);</script>');
    }else{
        exit('<script>alert("no such user! ~~");location.replace(document.referrer);
</script>');
    }
}
?>

```

In this page, it will show three parts of information: friend search, received request and the friend list of this user. To add a friend, just type in the uid of the user people want send friend request to, and the friendship request will be sent immediately.

Show people send request to the user:

```
if(isset($_GET['v'])){
    if($_GET['v'] == 'yes'){
        $DB->query("UPDATE `friendship` SET `friConfirm1`='1',`friconfirm2`='1' WHERE
uid1='{$_GET['uid1']}' and uid2 = '{$_GET['uid2']}'");
    }elseif($_GET['v'] == 'no'){
        $DB->query("delete from `friendship` WHERE uid1='{$_GET['uid1']}' and uid2 =
'{$_GET['uid2']}'");
    }
    exit('<script>alert("success! ~~");location.replace(document.referrer);</script>');
}
```

Accept or reject:

```
if($_POST['act']=='FriendSearch'){
    $sql="select * from user where uid = '{$_POST['uid']}' and uid !=
'{$_user['uid']}'";
    $row=$DB->query($sql)->fetch();
    if($row){
        $DB->query("INSERT INTO `friendship`(`uid1`, `uid2`, `friConfirm1`,
`friconfirm2`) VALUES ('{$_user['uid']}', '{$_POST['uid']}', '1', '0')");
        exit('<script>alert("request send! ~~");location.replace(document.referrer);
</script>');
    }else{
        exit('<script>alert("No such user! ~~");location.replace(document.referrer);
</script>');
    }
}
```

To send friend request:

```
$sql="SELECT `uname`,`uid` FROM `user` WHERE uid in (SELECT uid2 FROM `friendship`
WHERE uid1='{$_user['uid']}' and FriConFirm1='1' and FriConFirm2='1')";
$rows=$DB->query($sql);
while ($row=$rows->fetch()){
    echo <<<HTML
    <tr>
    <td><button type="button" id="Confirm">{$row['uid']}</button></td>
    <td>{$row['uname']}</td>
    <tr>
```

## 7. Current

When user click the "current " button, he will get into the page to set his current time or current location. To set the current location, user can select point from the GOOGLE MAP we connect with the google map API to get the latitude, longitude , address and name of the location. And type in the current time.

Oingo

Search projects

breakfast

ctest

10003

breakfast

Home

Notes & Comments

Note

Current

Comment

Filter

Profile

Friend

Set Your Location

地图 卫星图像

Location

latitude

longitude

Start Time

2018-11-11 11:11:11

SUBMIT

CANCEL

## 8. Comment

Oingo

Search projects

breakfast

ctest

10003

breakfast

Home

Notes & Comments

Note

Current

Comment

Filter

Profile

Friend

Poster	nid	Location	Content	Start Time	End Time
The Mermaid Inn	3	96 2nd Avenue, New York	So many prople in Central Park!	2018-11-25 12:00:00	2018-11-25 13:00:00

Comment

SUBMIT

userid	username	cotent
10001	atest	cool
10003	ctest	pretty
10004	dtest	haha
10003	ctest	12_16test

## FUNCTION

### Filter.php

Addfilter(); insert filter (filtered condition)

Uidfilter();

all filters for this user

Noteoffilter();

Create and display view v2: a note that satisfies all current filters for the current user

```
function noteoffilter() {
    global $DB;
```

```

    $sql = "create view v2 as select * from filter natural join schedules natural join
locationall where uid = '{$_SESSION['']}' and flocationID =lid and fscheduleID =
schedules.sid;";
    $result = $DB->query($sql);
    $sql2 = "select * from v2";
    $result2 = $DB->query($sql2);
    if ($result === false) {
        echo "error";
    } else {
        $fieldCount = mysql_num_fields($result);

        echo "<table border='1'>";

        echo "<tr>";
        for ($i = 0; $i < $fieldCount; ++$i) {
            $fieldName = mysql_field_name($result, $i);
            echo "<th>" . $fieldName . "</th>";
        }

        //echo "<th>action</th>";
        echo "</tr>";

        while ($rec = mysql_fetch_assoc($result)) {
            echo "<tr>";
            for ($i = 0; $i < $fieldCount; $i++) {
                $fieldName = mysql_field_name($result, $i);
                echo "<td>" . $rec[$fieldName] . "</td>";
            }
            echo "</tr>";
        }
    }
}

```

### Info\_note();

Create and display the view: V3: Display all the information of all the current notes; set the note reminder time schedule; tag tag; address location;

```

function info_note() {
    echo "_____";
    echo $buildlocationID;
    $sql = "drop view v1;";
    $result1 = mysql_query($sql);
    $sql2 = "create view v1 as select * from note natural join locationall natural join
schedules where buildlocationid =lid and note.sid = schedules.sid;";
    $result = mysql_query($sql2);
    $sql = "select * from v1";
    $result = mysql_query($sql);
    if ($result === false) {
        echo "error";
    } else {

```

```

        $fieldCount = mysql_num_fields($result);

        while ($rec = mysql_fetch_assoc($result)) {
            echo "<tr>";
            for ($i = 0; $i < $fieldCount; $i++) {
                $fieldName = mysql_field_name($result, $i);
                echo "<td>" . $rec[$fieldName] . "</td>";
            }
            echo "</tr>";
        }
    }
}

```

That is, the foreign key sid(schedule), lid(location) & referenced in the note is connected to the note through the tag\_note table;

### **Note\_cur();**

first info\_note() to get v2; get the current time (current table) and the matching note in v2 (

```

function note_cur() {
    info_note();
    echo "_____";
    echo $buildlocationID;
    $sq2 = "select *,distance(c.clongitude,c.clatitude,v2.llongitude,v2.llatitude)
    from current c,v2
    where distance(c.clongitude,c.clatitude,v2.llongitude,v2.llatitude) < v2.fradius
    and c.ctime between v2.sstarttime and v2.sendtime;";
    $result = mysql_query($sq2);
    if ($result === false) {
        echo "error";
    }
}

```

That is, the distance between the current location and the location in the note less than fradius in the filter)

### **AutoState();**

AutoState() is the function we update the current state automatedly.



```

function AutoState() {
    global $DB;
    $nowtime = date('H:i:s');
    $sql = "select `stateid` from statetime where '$nowtime' > starttime and
'$nowtime' < endtime limit 1";
    $row = $DB->query($sql)->fetch(PDO::FETCH_ASSOC);
    if ($row) {
        $stateid = $row['stateid'];
    } else
        $stateid = 1;
    $sql = "select * from state where stateid = '{$stateid}' limit 1";
    $state = $DB->query($sql)->fetch(PDO::FETCH_ASSOC);
    return $state['statename'];
}

```

## Post.php

Postnote(); insert a new note; and another table referenced by the note (schedule;tag; tag\_note)

```

if ($_POST) {
    $sql1 = "INSERT INTO `schedules`(`starttime`, `sendtime`, `repeat`, `duration`)
VALUES ('{$_POST['starttime']}','{$_POST['endtime']}','{$_POST['Repeat']}','0')";
    $sql2 = "INSERT INTO `location`(`lname`, `laddress`, `llatitude`, `llongitude`)
VALUES ('{$_POST['Name']}','{$_POST['Address']}','{$_POST['latitude']}','{$_POST['longitude']}')";

    $DB->query($sql1);
    $sid = $DB->lastInsertId();
    $DB->query($sql2);
    $lid = $DB->lastInsertId();

    $sql3 = "INSERT INTO `note`(`uid`, `ncontent`, `buildlocationid`, `radius`, `sid`,
`commentRight`) VALUES ('{$user['uid']}','{$_POST['Content']}','{$lid}','{$_POST['range']}','{$sid}','{$_POST['range']}')";

    $DB->query($sql1);
    $DB->query($sql3);
    $nid = $DB->lastInsertId();

    $tagarr = explode(';', $_POST['Tag']);
    foreach ($tagarr as $v) {
        $row = $DB->query("select * from `tag` where `tname` = '{$v}' limit 1")->fetch(PDO::FETCH_ASSOC);
        if ($row) {
            $tid = $row['tid'];
        } else {

```

```

        $DB->query("INSERT INTO `tag`(`tname`) VALUES ('{$v}')");
        $tid = $DB->lastInsertId();
    }
    $DB->query("INSERT INTO `tag_note`(`tid`,`nid`) VALUES ('$tid','$nid')");
}
exit('<script>alert("success! ");location.replace(document.referrer);</script>');
}

```

Note, need to be changed: the address latitude and longitude here, should not be manually entered, passed from the Google MAP API

### Current:

Simulated position change

### header:

The content of the top part of the page where the header is stored, the state is manually switched there.

### Function

Function is basically a commonly used function, such as multiple pages will be used.

### Config.php

Config stored database connection parameters

### Common.php

To connection database

## Summery

---

Through this project, we learn how to combine PHP, MySQL and JavaScript together to implement a website and achieve some interesting function on the website. In our website, user not only can post note to share interesting things with other users but also can found some fantastic location to spend their time efficiently. Using the filter they add, the website can remind them never miss the chance to have a good time. They also can make friends with each other base on the note match one's filter.

To post a note, people can use the Google map to get the exactly location that is the latitude and longitude.

After they post the filter, the website will response with all of the note that match with the user's all filter and the all of the filter of this user. He also can manage the filter by click the delete button between the filter to remove the filter he does not need. In current.php, user can change their current time and current location. And each time the user refresh the website, the database will get the new value of the current table. Also, it is common that when some one planned few weeks ago to go some where, but for now, they are too tired to go there, they can change their current state to sleep and the website will leave the user alone instead of disturbing him with the reminder.

If we have chance to advance our project, we may add a 'guide' function that can guide user to the location when the note pop out to remind them. It is very inconvenient to remember the address of the destination and switch to another app to figure out how to get there. So we want to add this guide function. For example, when a user around SOHO 1km, and the website figure out to remind him to shopping there. After

click a 'guide' button, the website can connect to the map app and send the location of the destination to the app to guide the user to the place they want to go there.