Final Report

Group 7: XU HuaYang && WU Zhuoming && WU Zihao

Problems:

Most current online storage system can only use as a backup file system for users, users can not search the system to see whether there are some resources they needed have already been uploaded to the system by others. Although this protect the privacy of users, but also reduce the efficiency of resource utilization.

Goals:

Build an online storage system on which we can not only use it as a backup file system to backup our files, but also act like an anonymous shared resource platform where users can search files uploaded as public resources without knowing who upload the files.

Description:

APP Name: Anonymous Files

Basic functions:

1)User authentication: Login, Register

-Registered user can login and manage their own files just like the normal
online storage file system.

-Anonymous user will not be assigned the storage space to storage their own
file.

2)Upload:

-Registered user can use the upload function to upload files. They will be given two mode of upload, private or public:

- Private upload: files will be uploaded to user’s storage space, and this file
is not accessible by others.

-Public upload: file will be uploaded to the public space with erasing uploader's name. This file can be accessed by anyone. The public files will appear on the File Pool and every one can see it.
-Anonymous user can also upload file, there will be a file size limit for the
anonymous user and each upload will generate a unique Extract Code. Once
the file is downloaded by others, the timer will start. The file record will be erased one day after the beginning of the timer.
-All uploaded files can be renamed by uploader and added three tages to recognize it.

3)Download:

For registered user, they can

-Download files upload by themselves.
-Download files upload by others in public upload mode.
-Download files upload by anonymous user with Extract Code.

For anonymous user, they can

-Download files upload by others in public upload mode.
-Download files upload by anonymous user with Extract Code.

4)Share:

For registered user

-To shared public files, they can just tell others the files name.
-Upload file as anonymous user and share Extract Code.
For anonymous user
- Upload file as anonymous user and share Extract Code.

5)Search:

There are three modes of searching:

-User can input file name or part of the file name and search. All files match the file name or part of the file name will be shown. This part is implemented in the way of fuzzy search.
-User can input the tags or related keywords to search the results. The The files that have the corresponding tags will be shown.
-User can input the extract code, which is used for anonymous uploading file. The correct extract code will lead the client to the only matching file.

The three modes are merged together. The clients will input his searching content into one text box and the searching algorithm will choose compatible and related functions for him. The results will be combined together and then shown

Technical Details && Implementation

For connection security, force https connection:

First generate a private server.key and server.cert used in https connection.

```
var server = https.createServer({
  key:fs.readFileSync('server.key'),
  cert:fs.readFileSync('server.cert')
},app)
.listen(3000,function(){

})
```

Then use the module 'express-force-https' to force user to browse the web using https. The request from a http connection is forbidden.

```
var https = require('https');
var secure = require('express-force-https');
```

1)Client Side:

-Generate UI through the web page.

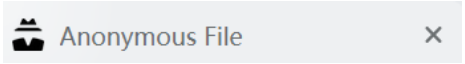
The web page is render using ejs, we can easily render the page by p roviding some additional data.

Implementation of letting web page show user name after logged in: First render the page using loginState,username. Those data are initialized in the first connection.

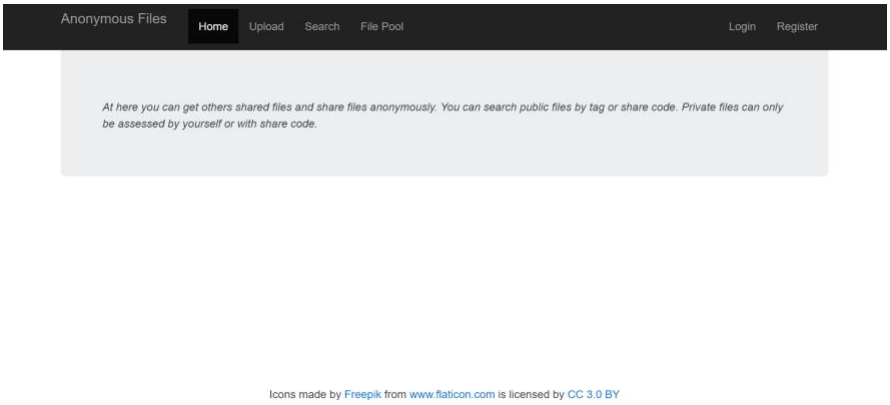
```
res.render('index', { title: 'Anonymous Files',data:tempData,page:0,loginState:req.session.loginState,username:req.session.username});
```

Then we can use those data in front end. EJS allow use to write java script code directly in web page.

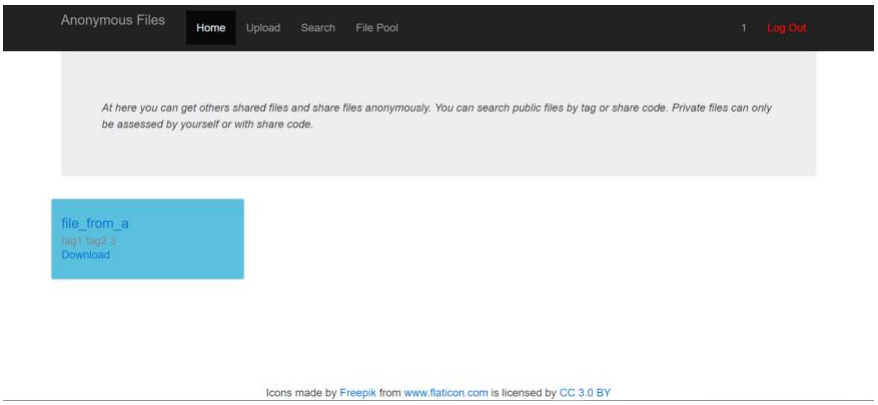
```
<ul class="nav navbar-nav pull-right">
  <% if(loginState != 2){%>
  <li ><a href="/login">Login</a></li>
  <li><a href="/register">Register</a></li>
  <%}else{%>
  <li><a><%=username%></a></li>
  <li><a id="logout" style="color: #FF0000" >Log Out</a></li>
  <%}%>
</ul>
```



Web page before log in:



Web page after log in:



Implementation of displaying files in web page:

The data of files are stored in mysql database. So we fetch the corr esponding data then render page with them.

```
mysql.Display(client,function(result){
tempAnonyousData = JSON.parse(JSON.stringify(result));
console.log(tempAnonyousData);
res.render('index', { title: 'Anonymous Files',data:tempAnonyousData,
```

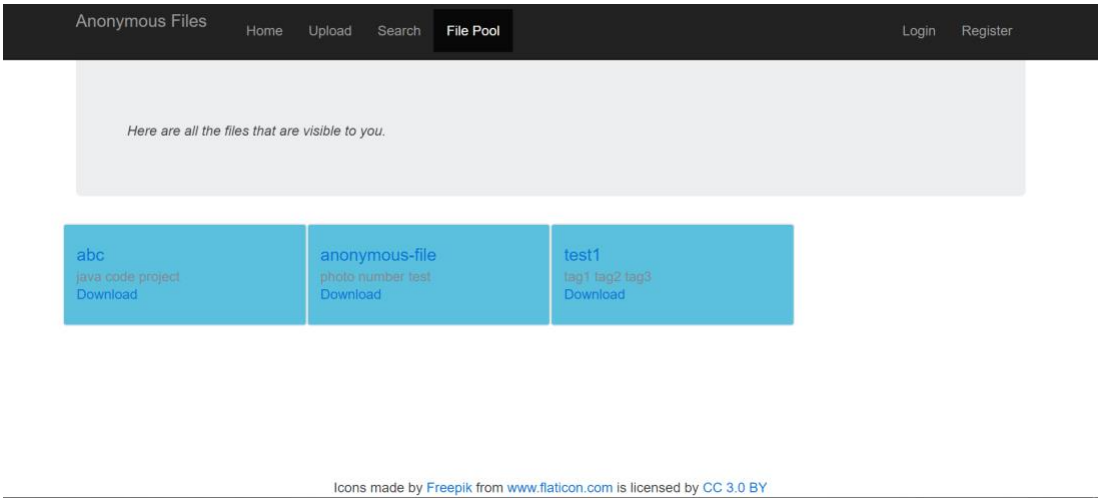
The write a loop using javascript to generate cards to display data.

First we include a <a href=" <%=temp%> target=" _blank" > to implement the function that when the user click this card, he will be redirect to a new page to see the content of the file.

Then we display the file name, the 3 tags of file and the download link.

In download link, use down=" <%=data[i]['file_name']%> to implement the function that when the user click this link, the download will start.

```
<% for(var i=0;i < data.length;i++){%>
<% var temp="."+data[i]["path"];%>
<a href="<%=temp%>"  target="_blank"><div class="card col-sm-3 text-white bg-info mb-3">
<div class="card-body" id="<%=i%>">
  <br>
  <h5 class="card-title" id="title"><%=data[i]["file_name"]%></h5>
  <h6 class="card-subtitle mb-2 text-muted" id="tag"><%=data[i]["tag1"]%> <%=data[i]["tag2"]%> <%=data[i]["tag3"]%></h6>
  <a href="<%=temp%>" class="card-link" id="download<%=i%>" download="<%=data[i]['file_name']%>" >Download</a>
</div>
<br>
</div></a>
<%}%>
```



2)Server Side:

Using ngrok to expose your website run on localhost.

To expose a https connection, you have to login first.

```
#./ngrok authtoken Your Tunnel Authtoken
```

Then expose the website using following command.

```
./ngrok http https://localhost:3000
```

Then other computers can connect to this website.

```
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://0b0d9f54.ngrok.io -> https://localhost:3000
Forwarding                    https://0b0d9f54.ngrok.io -> https://localhost:3000
```

-Use MySQL to manage the database system for user authentication and file record management.

MYSQL data base management:

Separating the using and implementation of database.Hide all the details in dao/dbConnect.js.

```
var mysql = require('dao/dbConnect.js');
var client = mysql.connect();
```

In dao/dbConnect.js file:

Using exports to let those function visible in index.js.

```
//export the defined function to the project.

exports.connect = connectServer;
exports.Login = Login;
exports.Register = Register;

exports.Display = Display;
exports.DisplayUser = DisplayUser;
exports.DisplayFilePool = DisplayFilePool

exports.Search = Search;
exports.Upload = Insert;
exports.UploadForAnonymous = InsertForAnonymous;
```

3)Upload Side

Muter is used for efficient storage. The uploaded file will be stored in a temporary folder and then moved to the corresponding folder.

```
var storage = multer.diskStorage({
    destination: function (req, file, cb) {
        cb(null, outtestFolder);
    },
    filename: function (req, file, cb) {
        //var external = path.extname(file.originalname);
        //cb(null, file.fieldname + '-' + Date.now() + external);
        file_name = file.originalname;
        file_extend = path.extname(file.originalname);

        console.log(file_name);
        cb(null, file.originalname);
    }
});
```

A piece of data recording the information of the uploaded file. An unique id will be created as a key of the data.

```
var id = crypto.createHash('sha256').update(Date()+file_name).digest('hex');
```

The information of the data is stored in a MySQL database:

```
function Insert(client,id,file_name,uploader,type,path,tag1,tag2,tag3,callback){
    //var id = crypto.createHash('sha256').update(Date()+file_name).digest('hex');
    var parm = [id, file_name, uploader, type, path, tag1, tag2, tag3];

    var insert_stmt  = 'INSERT INTO file (id,file_name,uploader,type,path,tag1,tag2,tag3) VALUES(?,?,?,?,?,?,?,?)';
    client.query(insert_stmt, parm, function (err,result){
            if(err){
                console.log('[INSERT ERROR]', err.message);
            }
            var str = JSON.stringify(result);
            console.log(str);
            callback(str);
        });
}
```

If the file is uploaded anonymously, a extract code will be generated using md5 hashing. The extract code will be returned to the uploader in order to locate the file.

```
var extract_code = md5(Date()+file_name);
```

The information of the anonymous file will be stored in another table in database, A timer is set and the file will be deleted after one day.

```
function InsertForAnonymous(client,extract_code,fileName,full_path,type,tag1,tag2,tag3,callback){
    var anonymous_insert_stmt = 'INSERT INTO anonymous (id, file_name, path, type,expire_time,tag1,tag2,tag3) VALUES(?, ?, ?, ?,?,?,?,?)'
    //var extract_code = md5(Date()+fileName);
    var date = new Date();
    var year = date.getFullYear();
    var month = date.getMonth();
    var day = date.getDate()+1;
    var hour = date.getHours();
    var minute = date.getMinutes();
    var second = date.getSeconds();
    var tomorrow = year+'-'+month+'-'+day+' '+hour+'-'+minute+'-'+second;
    var formated_tomorrow = moment(new Date()).format(tomorrow);
    var newFileName = tomorrow + fileName;

    //----------------------just use fileName instead of newFileName
    var parm = [extract_code, fileName, full_path, type,formated_tomorrow,tag1,tag2,tag3];

    client.query(anonymous_insert_stmt, parm, function(err, result){
        if(err){
            console.log('[ANONYMOUS INSERT ERRER]', err.message);
        }
        var str = JSON.stringify(result);
        console.log(str);
        console.log(extract_code);
        callback(extract_code);
    });
}
```

4) Search Side

Only one input box is given to users. Once the back-end gets the input, it will choose the searching mode according to the length of the input.

A id of a file uploaded by registered user has a length of 64 bits.

```
if(search_content.length == 64){
    //search by id
    var select_stmt = '';
    if(loginState == 2){
        select_stmt = select_stmt_part + "WHERE id ='"+search_content+"' AND (uploader='"+ username +"' OR type = 'public')";
    }else{
        select_stmt = select_stmt_part + "WHERE id ='"+search_content+"' AND type = 'public'";
    }
    client.query(select_stmt, function (err, result){
        if(err){
            console.log('[SELECT ERROR]', err.message);
        }
        results = result;
        //console.log(result);
        callback(result);
    });
}
```

An extract code has a length of 32 bits.

```
else if(search_content.length == 32){
    var result1 = [];


    var select_anonymous_stmt = "SELECT * FROM anonymous WHERE id = '"+search_content+"'";
    client.query(select_anonymous_stmt, function(err, result){
        if(err){
            console.log('[ANONYMOUS SLECT ERROR]', err.message);
        }
        result1 = result;
        //console.log(result1);
        callback(result);
    });
}
```

Other inputs will be searched using fuzzy search. It will not only search the name of the file, but also will search the tags of the files.

```
else{
    var select_stmt1 = '';
    var select_stmt2 = '';
    var select_stmt3 = '';
    var select_stmt4 = '';
    var select_stmt5 = "(" + select_stmt_part2 + "WHERE file_name ='" + search_content + "' LIMIT 2)";
    var select_stmt6 = "(" + select_stmt_part2 + "WHERE file_name LIKE '%"+ search_content+"%' LIMIT 5)";
    var select_stmt7 = "(" + select_stmt_part2 + "WHERE find_in_set('"+ search_content+"', file_name) LIMIT 5)";
    var select_stmt8 = "(" + select_stmt_part2 + "WHERE find_in_set('"+search_content+"', tag1) OR find_in_set('"+search_content+"',t


    if(loginState == 2){
        select_stmt1 = "(" + select_stmt_part + "WHERE file_name ='"+search_content+"' AND (uploader='"+ username +"' OR type = 'publ
        select_stmt2 = "(" + select_stmt_part + "WHERE file_name LIKE '%"+ search_content+"%' AND (uploader='"+ username +"' OR type
        select_stmt3 = "(" + select_stmt_part + "WHERE find_in_set('"+ search_content+"', file_name) AND (uploader= '" + username + "
        select_stmt4 = "(" + select_stmt_part + "WHERE (find_in_set('"+search_content+"', tag1) OR find_in_set('"+search_content+"',t
    }else{
        select_stmt1 = "(" + select_stmt_part + "WHERE file_name ='"+search_content+"' AND  type = 'public' LIMIT 2)";
        select_stmt2 = "(" + select_stmt_part + "WHERE file_name LIKE '%"+ search_content+"%' AND type = 'public' LIMIT 5)";
        select_stmt3 = "(" + select_stmt_part + "WHERE find_in_set('"+ search_content+"', file_name) AND type = 'public' LIMIT 5)";
        select_stmt4 = "(" + select_stmt_part + "WHERE (find_in_set('"+search_content+"', tag1) OR find_in_set('"+search_content+"',t
    }
    var unionString = " union ";
    var final = [];
    var finalStatement = select_stmt1 + unionString + select_stmt2 + unionString + select_stmt3 + unionString + select_stmt4 + unionS
    console.log(finalStatement);
    client.query(finalStatement,function(err,result){
        if(err){
            console.log('[SELECT1 ERROR]', err.message);
        }
        final = result;
        callback(final);
    });
}
```

Whenever anyone use searching functions, it will first detect whether there are any expired file. If expired files exists, expired files will be deleted first.

```
var date = new Date();
var year = date.getFullYear();
var month = date.getMonth();
var day = date.getDate();
var hour = date.getHours();
var minute = date.getMinutes();
var second = date.getSeconds();
var right_now = year+'-'+month+'-'+day+' '+hour+'-'+minute+'-'+second;
var formated_now = moment(new Date()).format(right_now);

var delete_stmt = "Delete from anonymous WHERE expire_time > '" + formated_now "'";

client.query(delete_stmt, function(err, result){
    if(err){
        console.log('[DELETE ERROR]', err.message);
    }
    console.log("DELETE successfully");
});
```

5)Login Side

The information of the user is stored in MySQL. The username is stor-ed in plain text and the password is hashed and stored in the MySQL. When the user tries to login, its username and password will be sent to the back-end. Then the password will be hashed using MD5. The back-end will compare the computed results with the retrieved results from the database.

```
.post('/login',function(req,res)
{
    var client = mysql.connect();
    mysql.Login(client,req.body.username,
        function(result){
            console.log(result.length);
            if(result.length == 1){
                console.log(result[0].pass);
                var md5 = crypto.createHash('md5');
                md5.update(req.body.password);
                var pass = md5.digest('hex');
                if(pass == result[0].pass){//login sucess
                    username = req.body.username;
                    loginState = 2;
                    have_logined = true;
                    res.redirect('/');
                }
                else{//login fail
                    loginState = 1;
                    res.redirect('login');
                }
            }
            else{ //login fail
                loginState = 1;
                res.redirect('login');
            }
        });
});
```

6)Register Side

Repeated username is prevented during registering. The password is hashed using MD5 and then sent to the back-end.

```
.post('/register',function(req,res)
{
    var md5 = crypto.createHash('md5');
    md5.update(req.body.password);
    var pass = md5.digest('hex');
    var client = mysql.connect();
    mysql.Register(client,req.body.username,pass,
        function(err){
            if(err){
                RegisterState = 1;
                res.redirect('/register');
            }
            else{
                RegisterState = 2;
                res.redirect('/login');
            }
        });
});
```

Features and impact:

The Anonymous File is more like a resource pool, or like a private cloud st orage. Anyone can upload the resource with various kind of methods and then others can get it from our application easily.

-Novelty

Since we have various kinds of method to access the file, but most of them can not get the file uploader’s information, which can have a good protect ion on user’s privacy. What’s more, the function ‘upload as anonymous’ is a quite safe and convenient way to shared files to others because it does not need to login and after download, no file record will exist on the server, which means that no one can get the copy this file. This function i s just like the ‘burn after reading’, we believe that this function will bring

-Technical impact

Basically, we have merged the advantages of online storage system and resource pool, we make sure that user can do the backup and search for resource within one application. Adding the anonymous characteristic, users' privacy has a better protection.

-Social impact

As shown in our application name, this is an anonymous file resource platform, although we just want to protect the user privacy, we cannot predict that if there will be someone to shared or transmit some banned resource through our application. We are considering some methods to balance the privacy and justice.

Methodology:

We plan to build our cloud storage system based on two parts. One is about the client side. We need to build the interface. Also, the deduplication and encryption are done locally. Another is the server side, the file information (including file index, file name, owner and permission) and user account information will be stored in a MYSQL data base with necessary encryption. Then using the file information to rebuild the file and transmit it to client side.

We provide a website for client to communicate with server. There are mainly three features. First is the file upload part. User can upload a file even he has not logged in, while he can only set the file to private after the logged in condition. Then our cloud storage system will perform deduplication and encryption locally, it will enhance security in data transmission and reduce bandwidth. Secondly, after user has logged in, he can set the share code for private file owned by him. Others can get access to the file with the share code only once, then the code will get expired. Finally, we provide a simple GUI for user to view the file stored on his cloud storage. He can easily search public file on the server.

We need to implement those classes and functions listed. The user account management may have some template to follow. Also, deduplication is common in cloud storage implementation and we have known the algorithm and procedure to implement it. Our main problem is to keep the file on server with security and achieve endurable performance.

Evaluation Plan:

-User authentication:

Users register their accounts and use the account number and password to log in their account. Only when account number and password are matched, the user finishes authentication.

-Upload after authentication:

After authentication, users can upload files in public mode or private mode.

The files, which are uploaded in public mode, will appear and be stored in public field. The uploaders' name will be erased and thus not seen by anyone. The public file is accessible to and can be downloaded by anyone.

The files, which are uploaded in private mode, will appear and be stored in the uploader's private field. The private field is only accessible to the uploader himself unless
the uploader shares a sharing link to others.

-Upload before authentication:(as anonymous users)

Without login, users are only allowed to upload files under a specific limit for the size.

If the file size is above the limit, the system will reject the uploading and no files will
be uploaded to the storage system.

If the file size is within the limit, the system will accept the uploading and the file will
be uploaded to storage system. The file will be stored in the anonymous field. An extract
code will be generated and sent back to the uploader.

The extract code will be used to download the corresponding file.

-Download:

For authenticated users, they can download the files in their own private field. They cannot download the files in other users' private fields.

Authenticated users can generate and share a sharing link of a certain file in their private field. Then the user can share this sharing link in order to let others download the corresponding file in the private field.

The files in public filed are accessible to anyone and thus anyone can download.

The files in anonymous field can only be accessed and downloaded through the extract code by anyone. The files can only be downloaded once. Once the file is downloaded by others through the extract cod

e, the file and its record will be erased. The extract will become i
nvalid too.

-Search:

For an authenticated user, he can search a certain file through its name and access the relative results from public field and his private field. The files in other users' private field will not be s hown. For an unauthenticated user, he can search a certain file thro ugh its name and access the relative results only from public field. The clients can also use tags to search files or he can just input p art of the file name to get the compatible results.

-File Management:

For an authenticated user, he can see his private and public files in his home page. He can manage them, which means that he can uploa d, delete or rename them.For an unauthenticated user, he can only se e
the files in public field and download.

Conclusion:

Anonymous File has been implemented successfully. Multiple users are in con sideration. For safety reasons, https are forced to use when using Anonymous File. We choose to build Anonymous File on website, because it is more portable and can be accessible to anyone as long as he can use Internet. As a anonymous sharing pla tform, Anonymous File only store the files, which means that it will not run, dete ct or analyze the file. This may cause some malicious files to be uploaded and exp osed to the public. However, user still can share his files to anyone anonymously using our platform. Anonymous File protect the freedom of uploader and the uploade d files will not be scanned or analyzed in order to protect the privacy of the upl oaders.

Acknowledgement:

The existing framework we use: Express, Bootstrap and MySQL.

Explanation of the significant discrepancy in the commit lines:

We used the express framwork to start our project, then we established the testing enviromnet on XuHuayang's Computer, including the npm packages. After distributing the workload, we initially works seperatly. While during the process of merging code, we together use XuHuayang's computer to merge and test each function by copying and pasting the file. So almost all the commits are done by XuHuayang. We distributed the workload fair enough and each one had done 1/3 part of whole project.