Name: Sayan Acharya

Roll: 002010501009

Group: A1

Stream: CSE-2024

Assignment no: 2

Subject: Internet Tech Lab

Problem Description:

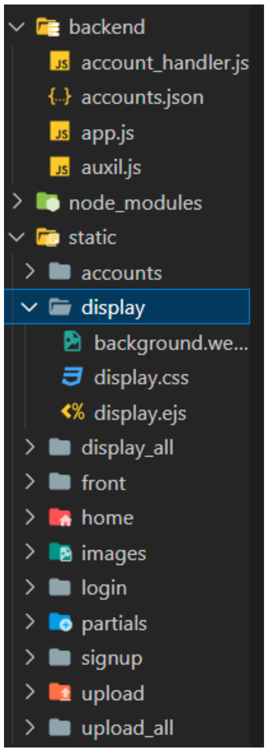
Write a multi-client data repository using Node.JS. You can use the Express framework. Clients would
upload text/image data to the server. A client may also get to view and download the collection. However,
editing of the data is not allowed by any client.
Identify where dependency injection design pattern is incorporated in the code implicitly or explicitly.

Approach and Structure:

The whole project contains 2 key folders:: backend and static.
Backend contains 3 key files:

1. app.js (the file that contains actual app logic. Various types of get/post request with params)
2. auxil.js (the file that contains helper functions for account validation, manager validation, authentication etc functions and middlewares)
3. accont_handler.js (the file that contains actual account manipulation functions for adding an account, setting up manager etc)

Static contains various folders. Each folder has an ejs file, some images and a css file. They are actually what is fetched for the frontend.


```
backend
    account_handler.js
    accounts.json
    app.js
    auxil.js
node_modules
static
    accounts
    display
        background.we...
        display.css
        display.ejs
    display_all
    front
    home
    images
    login
    partials
    signup
    upload
    upload_all
```

Code:

App.js:

```
const auxil=require('./auxil');
const { urlencoded } = require('body-parser');
const express=require('express');
const path=require('path');
const os=require('os');
const jwt=require('jsonwebtoken');
const cookie_parser=require('cookie-parser');
const file_upload=require('express-fileupload');
const fs=require('fs');
const {get_accounts,save_accounts}=require('./account_handler');

const app=express()
const accounts=get_accounts()
const static_path=path.resolve(__dirname,'../static/')
const account_path=path.resolve(__dirname,'../static/accounts/')

app.set('view engine','ejs');

app.use([
  cookie_parser(),
  urlencoded({extended:true}),
  express.json(),
  express.static(static_path),
  file_upload({
    limits: { fileSize: 50 * 1024 * 1024 },
  }),
])

app.use('/display',express.static(static_path));
app.use('/display_all',express.static(static_path));
// app.use('/upload',express.static(static_path));
// app.use('/upload_all',express.static(static_path));

app.post('/login',(req,res)=>{
  let {email,password}=req.body;
  // console.log(req.body);
    if(accounts.hasOwnProperty(email) && (password===accounts[email].password)){
      //send jwt cookie
      tok=jwt.sign({email},"whassup");
      res.cookie('jwt',tok,{maxAge:10*60*1000});
      res.json({ok:true});
    }else{
      res.json({error:'something went wrong',ok:false});
    }
})

app.post('/signup',(req,res)=>{
  let {email,password}=req.body;
    if(accounts.hasOwnProperty(email)){
      res.json({error:'user already exists',ok:false});
    }else{
      accounts[`${email}`]={password:password,is_manager:false};
      target_dir=account_path+`/${email}`
      if(!fs.existsSync(target_dir)){
        fs.mkdirSync(target_dir);
      }
      save_accounts(accounts);

      //send jwt cookie
      tok=jwt.sign({email},"whassup");
      res.cookie('jwt',tok,{maxAge:10*60*1000});
      res.json({ok:true});
    }
})

app.get('/login',(req,res)=>{
  // res.render(path.resolve(__dirname,'../static/login.ejs'))
  res.render('../static/login/login.ejs');

})

app.get('/front',(req,res)=>{
  res.render('../static/front/front.ejs');
})

app.get('/signup',(req,res)=>{
  // res.render(path.resolve(__dirname,'../static/login.ejs'))
  res.render('../static/signup/signup.ejs');

})

app.get('/logout',(req,res)=>{
  tok="";
  res.cookie('jwt','',{maxAge:1});
  res.redirect('/front');
});

app.use(auxil.auth_func);

app.get('/home',(req,res)=>{
  res.render('../static/home/home.ejs',{name:req.email,is_manager:req.is_manager});

})

app.get('/display',(req,res)=>{
  target_dir=account_path+`/${req.email}`;
  dict={}
  files=fs.readdirSync(target_dir);

  files.forEach(file=>{
    dict[file]=`accounts\\${req.email}\\${file}`;
  });
  res.render('../static/display/display.ejs',{dict,is_manager:req.is_manager});
})

app.get('/upload',(req,res)=>{
  res.render('../static/upload/upload.ejs',{file_name:null,is_manager:req.is_manager});

})

app.get('/delete_all/:email/:filename',auxil.manager_auth,(req,res)=>{
  let {email,filename}=req.params;

  target_dir=account_path+`\\${email}`;
  file_path=target_dir+`\\${filename}`;
  // console.log("deleting:",file_path);

  if(fs.existsSync(file_path)){
    fs.unlinkSync(file_path);
    res.json({ok:true});
  }else{
    res.json({ok:false});
  }
});

app.get('/download_all/:email/:filename',auxil.manager_auth,(req,res)=>{
  let {email,filename}=req.params;
```

```
            let target_dir=account_path+`\\${email}`;
            let file_path=target_dir+`\\${filename}`;
            // console.log(file_path);

            if(fs.existsSync(file_path)){
              res.download(file_path);
            }else{

            }
          });

          app.get('/display_all',auxil.manager_auth,(req,res)=>{
            let q=req.query;
            let email=q.input;
            let dict={};
            console.log(email);
            if(email){
              target_dir=account_path+`\\${email}`;
              files=fs.readdirSync(target_dir);

              files.forEach(file=>{
                dict[file]=`accounts\\${email}\\${file}`;
              });
            }
            console.log(dict)

res.render('../static/display_all/display_all.ejs',{is_manager:req.is_manager,dict,whom:email});

          });

          app.get('/upload_all',auxil.manager_auth,(req,res)=>{

res.render('../static/upload_all/upload_all.ejs',{is_manager:req.is_manager,dict:{},msg:""});

          });

          app.get('/delete/:file_name',(req,res)=>{
            target_dir=account_path+`\\${req.email}`;
            file_name=req.params['file_name'];
            file_path=target_dir+`\\${file_name}`;
            // console.log("deleting:",file_path);

            if(fs.existsSync(file_path)){
              fs.unlinkSync(file_path);
              res.json({ok:true});
            }else{
              res.json({ok:false});
            }

          })

          app.get('/download/:file_name',(req,res)=>{
            target_dir=account_path+`\\${req.email}`;
            file_name=req.params['file_name'];
            file_path=target_dir+`\\${file_name}`;
            // console.log(file_path);

            if(fs.existsSync(file_path)){
              res.download(file_path);
            }else{

            }

          });

          app.post('/upload',(req,res)=>{
            console.log(req.files);
            target_dir=account_path+`/${req.email}`
            if(!fs.existsSync(target_dir)){
              fs.mkdirSync(target_dir);
            }

            // res.json({ok:true})
            fs.writeFileSync(target_dir+`/${req.files.file.name}`,req.files.file.data);

res.render('../static/upload/upload.ejs',{file_name:req.files.file.name,is_manager:req.is_manager
});
          })

          app.post('/upload_all',auxil.manager_auth,(req,res)=>{
            // console.log(req.body)
            let email=req.body.username;
            // console.log(req.files);
            target_dir=account_path+`/${email}`
            if(!fs.existsSync(target_dir)){
              fs.mkdirSync(target_dir);
            }
            // res.json({ok:true})
            fs.writeFileSync(target_dir+`/${req.files.file.name}`,req.files.file.data);

res.render('../static/upload_all/upload_all.ejs',{file_name:req.files.file.name,is_manager:req.is_m
anager,msg:`${req.files.file.name} uploaded successfully for ${email}`});
          })

          app.listen(5000,(req,res)=>{
            console.log('listening on 5000....');
          })
```

Auxil.js:

```
//authentication function for all the requests
const jwt=require('jsonwebtoken');
const {get_accounts,save_accounts}=require('./account_handler');



function auth_func(req,res,next){
  let accounts=get_accounts();
  if(req.cookies.jwt){
    jwt.verify(req.cookies.jwt,'whassup',(err,vertoken)=>{
      if(err){
        console.log(err);
        res.redirect('/front');
      }else{
        req.email=vertoken.email;
        req.is_manager=accounts[req.email].is_manager;
        next();
      }
    })
  }else{
    res.redirect('/front');
  }


}

function manager_auth(req,res,next){
  if(!req.is_manager){
    res.status(400).send(`Forbidden access. ${req.email} is not manager!`);
  }else{
    next();
  }
}

module.exports={
  auth_func,manager_auth
}
```

account_handler.js:

```
const { Console } = require('console');
const fs=require('fs');
const path=require('path')

let path_accounts='./accounts.json';

function get_accounts(){
  let data=fs.readFileSync(path.resolve(__dirname,path_accounts),{encoding:'utf-8'});
  data=JSON.parse(data);
  return data;
}

function save_accounts(data){
  console.log(data);
  fs.writeFileSync(path.resolve(__dirname,path_accounts),JSON.stringify(data))
}

module.exports={get_accounts,save_accounts};
```

Some examples of static:

static/display/display.ejs:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <link href="./display.css" rel="stylesheet">
</head>
<body>

  <%- include ('../partials/header') %>
  <table id="table" >
    <tr id="table_heading" >
      <th> Name  </th>
      <th> Content </th>
      <th> Deletion Button </th>
      <th> Download Button </th>
    </tr>
    <% for(let k in dict) {%>
      <tr align="center">
        <td> <%=k %>  </td>
        <td> <img src="<%= dict[k]%>" alt="<%= dict[k]%>" height=300 width=300> </td>
        <% if (is_manager) { %>
          <td> <button id="id_delete_<%= k%>" class="delete_button" name="<%=
k%>">Delete</button> </td>
        <%} else { %>
          <td> <button id="id_delete_<%= k%>" class="delete_button" name="<%= k%>"
disabled="true">Delete</button> </td>
        <% } %>

        <td> <button id="id_download_<%= k%>" class="download_button" name="<%=
k%>"><a href="/download/<%= k%>" download="<%= k%>">Download</a> </button>
</td>
      </tr>
    <%}%>
  </table>

  <script>
    let delete_buttons=document.querySelectorAll(".delete_button");
    delete_buttons.forEach((b)=>{
      b.addEventListener('click',async (e)=>{
        let res= await fetch('/delete/'+b.name,{
          method:'GET',
          // headers: {
          //   'Content-Type': 'application/json'
          // },
          // body:JSON.stringify({
          //   file_name:b.name
          // }),
        });
        // alert('file deleted');
        location.reload();
      });

    });
  </script>

</body>
</html>
```

static/display/display.css:

```
body {
  background-image: url("background.webp");
  background-size: cover;
  /* background-repeat: no-repeat */
}

th, td {
  border: 1px solid black;
}

th {
  width: 500px;
  align-content: center;
}


table{
  border-width: 200px;
  align-content: center;
  font-family: cursive;
}



#table_heading{
  color:rgb(99, 42, 165)
}
```

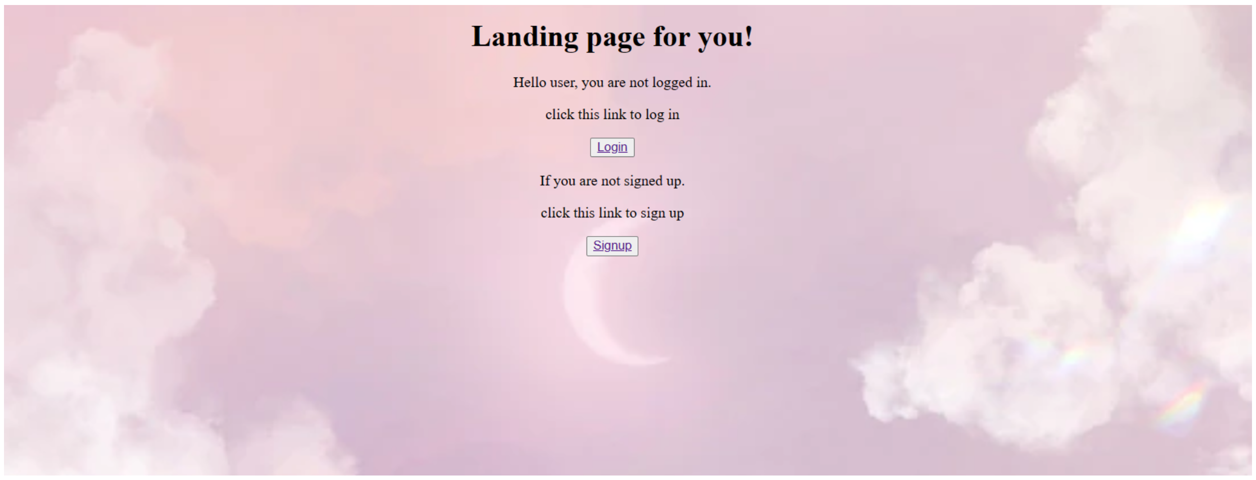
```
li, button {
  border-radius: 20px;
  margin: 20px;
}

button {
  font-weight: 900;
}

.delete_button {
  background-color: rgba(255, 0, 0, 0.425);
}

.download_button {
  background-color: rgba(0, 128, 0, 0.548);
}
```

Some outputs:



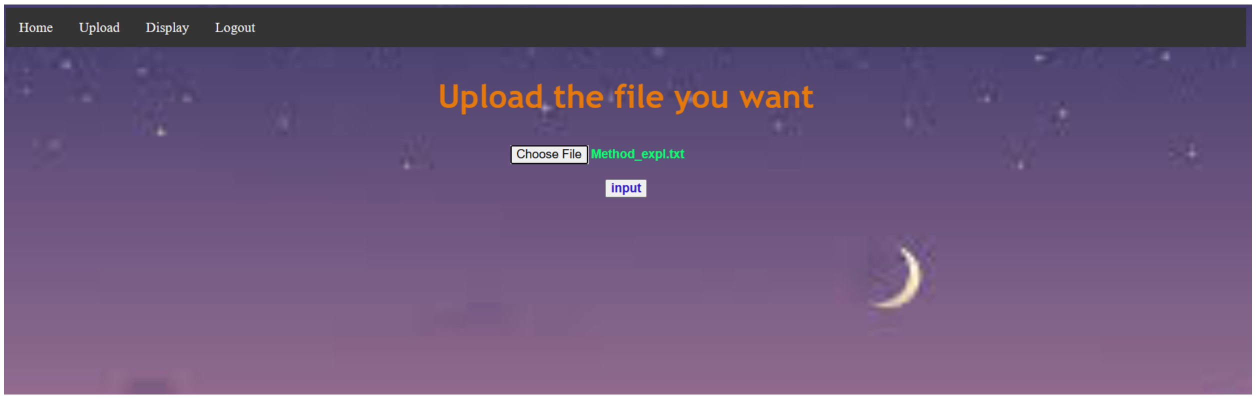
Landing page for you!

Hello user, you are not logged in.

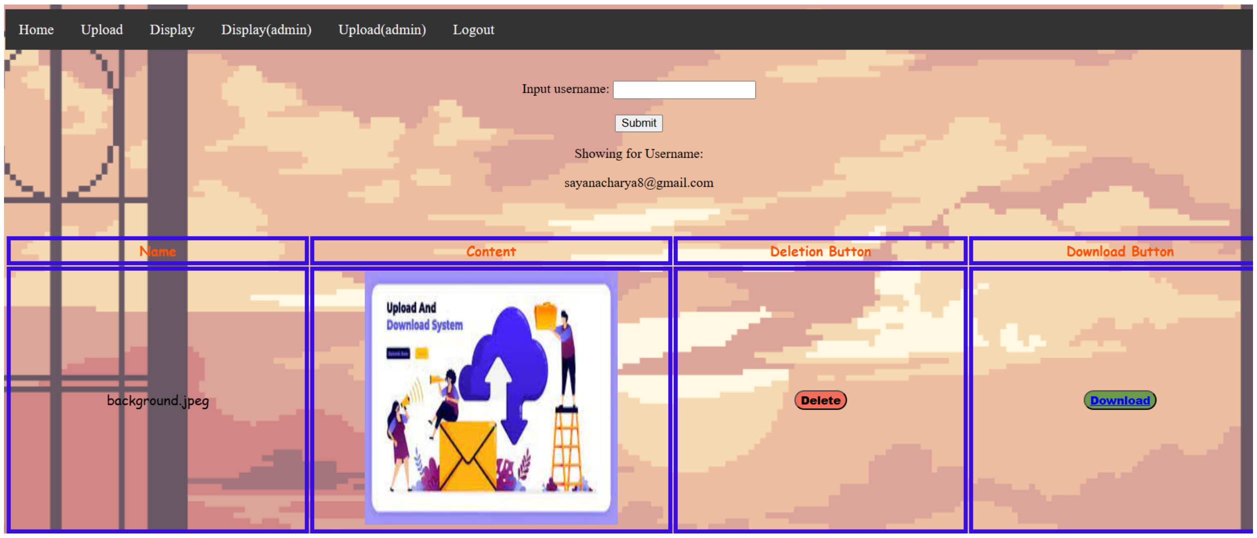
click this link to log in

If you are not signed up.

click this link to sign up



Upload the file you want

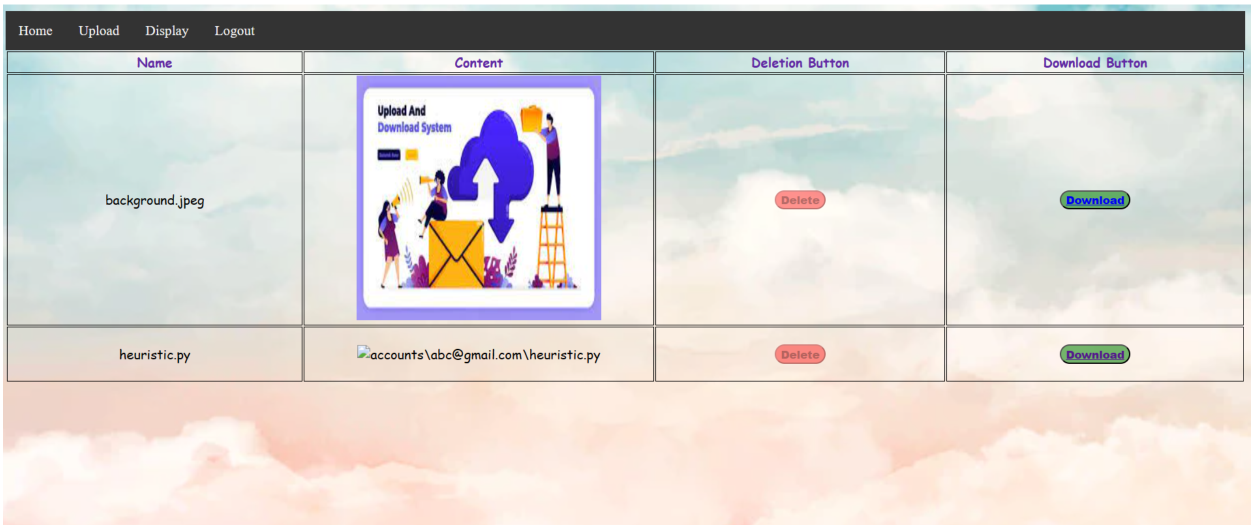


Input username:

Showing for Username:

sayanacharya8@gmail.com

Name	Content	Deletion Button	Download Button
background.jpeg	Upload And Download System	Delete	Download



Name	Content	Deletion Button	Download Button
background.jpeg	Upload And Download System	Delete	Download
heuristic.py	accounts\abc@gmail.com\heuristic.py	Delete	Download