생활코딩

Node.js 노드제이에스 프로그래밍

Gachon Shop – gcshop 프로젝트

2024년 2학기
8주

※ 프로젝트

* **routing**
* 인증
  - **login**
* 관리자 기능 – 사용자 관리, **DB**에 테이블 **CRUD**, 상품 입력, 코드 테이블 관리
* 게시판 생성 및 게시판에 글 **CRUD**
* 상품 조회 및 구매
* 장바구니 기능
* 경영자 화면

※ 8주차 수업의 범위

* routing

* project

  - login

01 라우팅

1. 현업에서 사용하는 라우터(URL 분류기)는 1000개 그 이상으로 늘어날 수 있음.
2. 주소 체계 변경 - 계층형으로 수정

	사용된 URL
'/'	'/'
	'/page/:pageId'
	'/login'
	'/login_process'
	'/logout_process'
	'/create'
	'/create_process'
	'/update/:pageId'
	'/update_process'
	'/delete/:pageId'
'/author'	'/author'
	'/author/create_process'
	'/author/update'
	'/author/update_process'
	'/author/delete'

3. main.js 파일 수정

- 홈디렉토리에 router 폴더를 생성

```
//사용자 정의 모듈
var db = require('./lib/db');
// var topic = require('./lib/topic');   주석처리
// var author = require('./lib/author'); 주석처리
var authorRouter = require('./router/authorRouter'); 추가
var rootRouter = require('./router/rootRouter');    추가
```

```
// app.get('/',(req,res)=>{                    모든 URL 분류기를 주석처리
//       topic.home(req,res);

// })

// app.get('/page/:pageId',(req,res)=>{
//       topic.page(req,res);
// })
```

```
app.use('/',rootRouter);                    두 줄 추가
app.use('/author',authorRouter);
```

4. router/rootRouter.js 파일 생성

```
const express = require('express');
var router = express.Router()

var topic = require('../lib/topic');

router.get('/',(req,res)=>{
    topic.home(req,res);

})

router.get('/page/:pageId',(req,res)=>{
    topic.page(req,res);
})

router.get('/login',(req,res)=>{
  topic.login(req,res);

})

router.post('/login_process',(req,res)=>{
  topic.login_process(req,res);

})

router.get('/logout_process',(req,res)=>{
  topic.logout_process(req,res);
})
```

```
router.get('/create',(req,res)=>{
    topic.create(req,res);
})

router.post('/create_process',(req,res)=>{
    topic.create_process(req,res);
})

router.get('/update/:pageId',(req,res)=>{
    topic.update(req,res);
})

router.post('/update_process',(req,res)=>{
    topic.update_process(req,res);
})

router.get('/delete/:pageId',(req, res)=>{
    topic.delete_process(req, res);
})

module.exports = router;
```

※ **router/authorRouter.js** 파일을 생성하시오

02 정적 파일

1. 정적 파일 - 이미지, 자바스크립트, **CSS** 와 같은 파일
2. **public** 폴더 생성하고 하위에 **images** 폴더 생성

   upsplash 사이트에서 이미지 다운로드
3. **main.js**에 다음 코드 추가 후 **http://localhost:3000/images/adrian.jpg** 들어가기

```
// 정적 파일
app.use(express.static('public'));
```

4. **home.ejs**에 다음 **img** 태그 추가

```
<h1><a href="/"><%=title%></a></h1>
<img src="/images/adrian.jpg" style="width:200px;height: 100px;display :block;margin-top:10px;">
<a href="/author">저자관리</a>
```

5. 파일 업로드

① multer 모듈 설치

```
npm install -s multer
```

② uploadtest.ejs 준비

```
<body>
    <%=lg%>
    <script>

        function displayFileName(){
            var fileName = $("#file").val();
            alert(fileName);
            $(".upload-name").val(fileName);
        }
    </script>
    <form action="/upload_process" method="post", enctype="multipart/form-data">
    <div class="filebox">
        <input class="upload-name" value="" placeholder="첨부파일">
        <label for="file">파일찾기</label>
        <input type="file" id="file" name="uploadFile" onchange="displayFileName()">
        <p><input type="submit" value="upload"></p>
    </div>
    </form>
</body>
```

02 정적 파일

[uploadtest.ejs 설명]

```
<body>
    <%=lg%>
    <script>

      ① function displayFileName(){
          ② var fileName = $("#file").val();
             alert(fileName);
          ③ $(".upload-name").val(fileName);
         }
    </script>
  ④ <form action="/upload_process" method="post", enctype="multipart/form-data">
  ⑤ <div class="filebox">
      ⑥ <input class="upload-name" value="" placeholder="첨부파일">
      ⑦ <label for="file">파일찾기</label>
      ⑧ <input type="file" id="file" name="uploadFile" onchange="displayFileName()">
        <p><input type="submit" value="upload"></p>
    </div>
    </form>
</body>
```

① ⑧의 파일선택 버튼에서 파일을 선택하여 파일의 이름이 바뀌면 실행되는이벤트 리스너 함수. ⑧의 onchange 이벤트 속성에 정의되어 있음.

② #file은 id가 file인 ⑧의 input 태그(파일 선택 박스)를 의미. 파일을 선택하면 파일명이 변수 fileName에 저장. $('#file') 표기는 jQuery 문법.

③ class가 upload-name인 ⑥의 textbox에 선택한 파일 이름을 출력. 이 때 경로가 "c:₩fakepath₩"로 나오는 이유는 브라우저가 보안의 이유로 경로를 막기 때문.

02 정적 파일

[uploadtest.ejs 설명]

```
<body>
    <%=lg%>
    <script>

      ① function displayFileName(){
          ② var fileName = $("#file").val();
             alert(fileName);
          ③ $(".upload-name").val(fileName);
         }
    </script>
  ④ <form action="/upload_process" method="post", enctype="multipart/form-data">
  ⑤ <div class="filebox">
      ⑥ <input class="upload-name" value="" placeholder="첨부파일">
      ⑦ <label for="file">파일찾기</label>
      ⑧ <input type="file" id="file" name="uploadFile" onchange="displayFileName()">
        <p><input type="submit" value="upload"></p>
    </div>
    </form>
</body>
```

④ form 데이터를 post 방식으로 전송할 때 전송하는 데이터를 인코딩 하기 위해 인코딩 타입에 대한 명시 필요.

- enctype 속성 : 폼 데이터를 서버로 전송할 때 데이터가 인코딩 되는 방법을 명시. 인코딩 방법은 다음 세 가지가 있음
  - application/x-www-form-urlcencoded : 디폴트 방법. enctype 속성 사용하지 않을 때 값. 일반 자료 보낼 때의 방법
  - multipart/form-data : 파일을 전송할 때 사용. 디폴트 방법에 비해 추가 정보가 함께 전송
  - text/plain : 인코딩 없이 전송. 보안에 취약하므로 비추.

02 정적 파일

[uploadtest.ejs 설명]

```
<body>
    <%=lg%>
    <script>

      ① function displayFileName(){
          ② var fileName = $("#file").val();
             alert(fileName);
          ③ $(".upload-name").val(fileName);
         }
    </script>
  ④ <form action="/upload_process" method="post", enctype="multipart/form-data">
  ⑤ <div class="filebox">
      ⑥ <input class="upload-name" value="" placeholder="첨부파일">
      ⑦ <label for="file">파일찾기</label>
      ⑧ <input type="file" id="file" name="uploadFile" onchange="displayFileName()">
        <p><input type="submit" value="upload"></p>
    </div>
    </form>
</body>
```

⑤ CSS에서 사용하기 위해 입력된 태그

⑥ 파일을 선택하면 파일명이 출력되는 박스

⑦ 파일찾기 글자 라벨. for 속성을 통해 ⑧과 연결. for의 의미 : id가 file인 태그에 대한 label이라는 의미.

⑧ 파일을 선택할 수 있는 type이 file인 input 태그. 파일을 선택하면 displayFileName()이 실행

5. 파일 업로드

③ rootRouter.js 에 다음 코드 추가

```
router.get('/upload',(req, res)=>{
    topic.upload(req, res);
})
```

④ topic.js 에 upload 메소드 추가

```
    upload: (req,res) => {
        var context = { lg: ''
                        };
        req.app.render('uploadtest',context, (err, html) => {
            res.end(html);    });
     }
```

5. 파일 업로드

⑤ rootRouter.js 에 다음 코드 추가

```
const multer = require('multer');
const upload = multer({
   storage: multer.diskStorage({
                  destination: function (req, file, cb) { cb(null, 'public/image');  },
                  filename: function (req, file, cb) {
                       var newFileName = Buffer.from(file.originalname, "latin1").toString("utf-8")
                           cb(null, newFileName); }
                   }),  });
```

```
router.post('/upload_process',upload.single('uploadFile'), (req, res)=>{
    var file = '/images/' + req.file.filename
    res.send(`
        <h1>Image Upload Successfully</h1>
        <a href="/">Back</a>
        <p><img src="${file}" alt="image 출력"/></p>`
    );
    console.log(file);

})
```

5. 파일 업로드

⑥ rootRouter.js 코드 설명

```
const multer = require('multer');
```

- multer 모듈을 import한다.
- 이미지, 동영상 등을 비롯한 여러 가지 파일들을 멀티파트 형식으로 업로드 할 때 사용하는 미들웨어.
- 멀티파트 형식이란 form 태그의 enctype이 multipart/form-data로 업로드 되는 데이터.

```
<form action="/upload_process" method="post", enctype="multipart/form-data">
```

- mulipart 폼을 통해 업로드하는 파일은 body-parser로는 처리할 수 없고 multer 모듈을 사용해야 함.
- multer 패키지 안에는 여러 종류의 미들 웨어가 들어 있음.

5. 파일 업로드

⑥ rootRouter.js 코드 설명

```
const upload = multer({
    storage: multer.diskStorage({
              destination: function (req, file, cb) { cb(null, 'public/image');  },
              filename: function (req, file, cb) {
                      var newFileName = Buffer.from(file.originalname, "latin1").toString("utf-8")
                      cb(null, newFileName); }
             }),
    });
```

- 기본 설정 – multer 함수의 인수로 설정을 넣는다.
- storage 속성 – 어디에(destination) 어떤 이름으로(filename) 저장할지를 지정한다.
- diskStorage 메소드 – 하드디스크에 업로드 파일을 저장하는 메소드
- destination과 filename함수 – req : 요청에 대한 정보
  file : 업로드 한 파일에 대한 정보
  cb : 함수, 첫번째 인수에는 에러가 있다면 에러를 넣고, 두 번째 인수에는 실제 경로나 파일 이름
  req, file의 데이터를 가공해서 cb에 넘기는 방식
- 설정이 끝나면 upload 변수 생성

5. 파일 업로드

⑥ rootRouter.js 코드 설명

```
router.post('/upload_process',upload.single('uploadFile'), (req, res)=>{
    var file = '/image/' + req.file.filename
    res.send(`
        <h1>Image Upload Successfully</h1>
        <a href="/">Back</a>
        <p><img src="${file}" alt="image 출력"/></p>`
    );
    console.log(file);

})
```

- upload 변수에 다양한 종류의 미들웨어가 들어 있음
- upload.single 미들웨어 – 하나의 파일만 업로드 할 때 사용

  single 미들웨어를 라우터 미들웨어 앞에 넣어두면 multer 설정에 따라 파일 업로드 후 req.file 객체가 생성.

  인수는 input 태그의 name
- 업로드 성공 시 결과는 req.file 객체 안에 들어 있습니다. req.body에는 파일이 아닌 데이터가 들어 있음.

Project – Gachon Shop(gcshop)

1. 프로젝트 생성 작업 순서

① person 테이블 생성

② gcshop 폴더 생성

③ 기존 작업 폴더의 package.json 파일을 복사하여 gcshop 폴더에 붙여넣기

④ vs code에서 gcshop 폴더를 열고 npm install 실행하기

⑤ gcshop 폴더에 views, lib, router, public 폴더 생성하기

Project – Gachon Shop(gcshop)

①person table 생성 – 사용자 테이블(회원, 관리자, 경영자 모두 저장)

필드명	데이터형	
loginid	varchar(10) NOT NULL	PK
password	varchar(20) NOT NULL	
name	varchar(20) NOT NULL	
address	varchar(100)	
tel	varchar(13)	000-0000-0000 형식
birth	varchar(8) NOT NULL	YYYYMMDD 형식
class	varchar(3) NOT NULL	CST(고객), MNG(관리자), CEO(경영자)
grade	varchar(1) NOT NULL	S(실버), G(골드), D(다이아몬드), S가 가장 낮고 D가 가장 높은 등급

```
insert into person values('M','M', '관리자', '서울', '010','00000000','MNG','S');
```

테이블을 생성하고 insert 문을 이용하여 관리자를 입력

Project – Gachon Shop(gcshop)

[화면 디자인을 위해 참고한 싸이트]

2022년, 2023년, 2024년 : https://getbootstrap.kr/

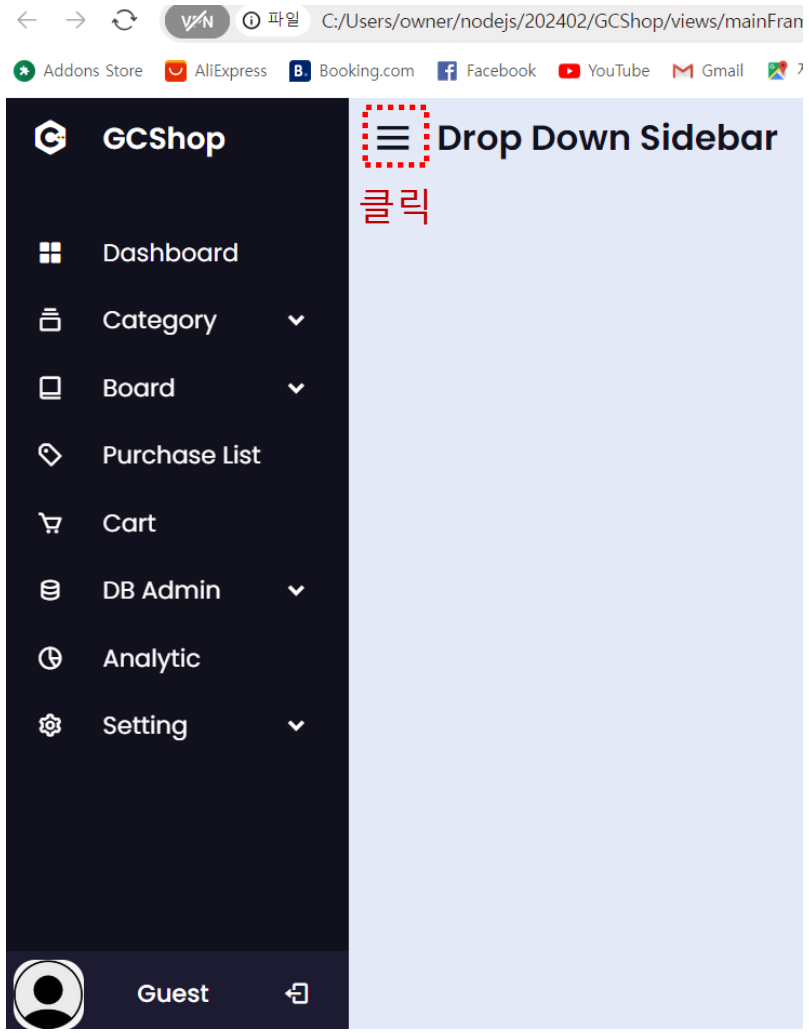
2024년 :

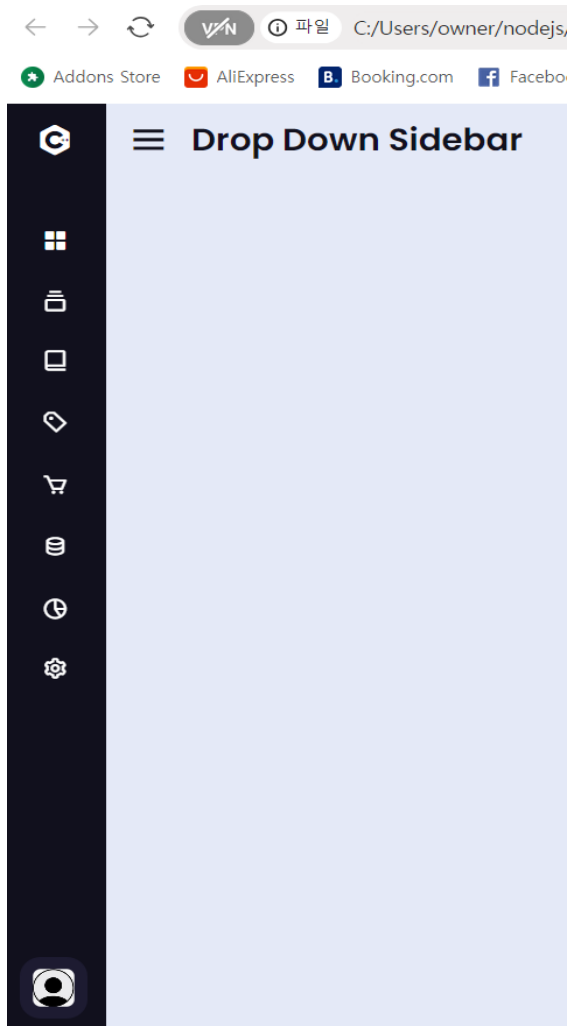
https://www.youtube.com/watch?v=ES8vJcUqE7s&list=PLImJ3umGjxdCjoBGj1eGQwcopR0P0edAK&index=3https://www.youtube.com/watch?v=ES8vJcUqE7s&list=PLImJ3umGjxdCjoBGj1eGQwcopR0P0edAK&index=3 ,

https://boxicons.com/

Project – Gachon Shop(gcshop)

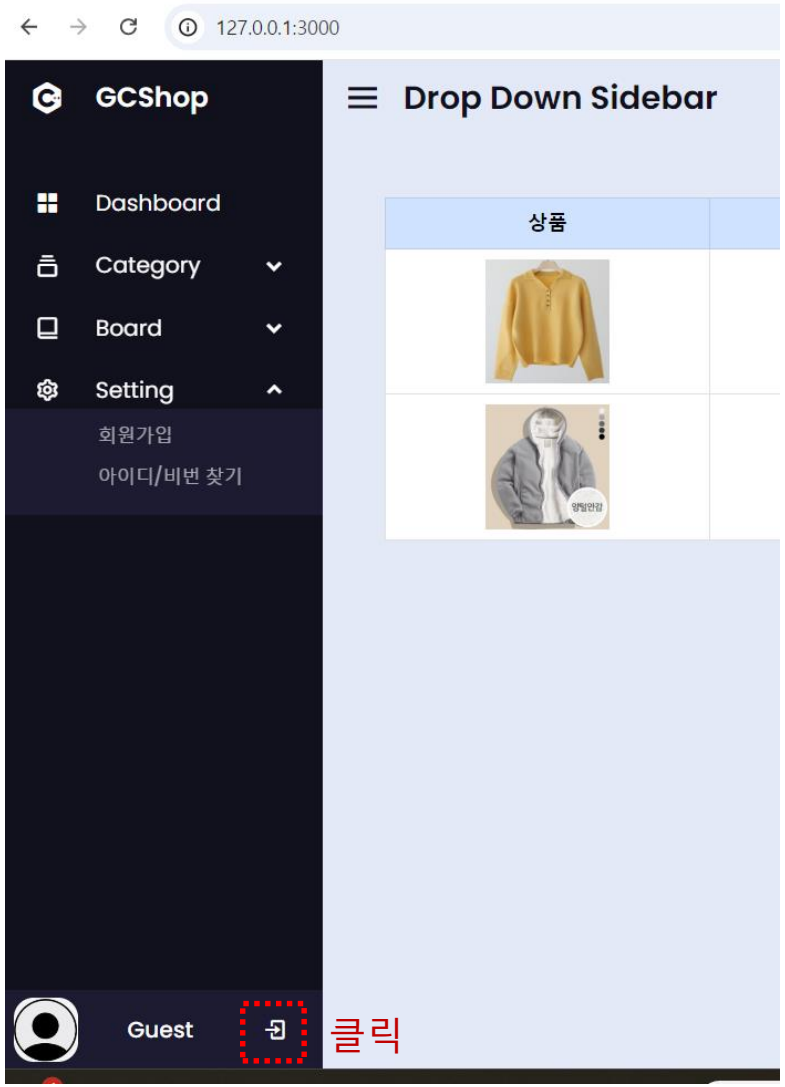
[mainFrame.html]

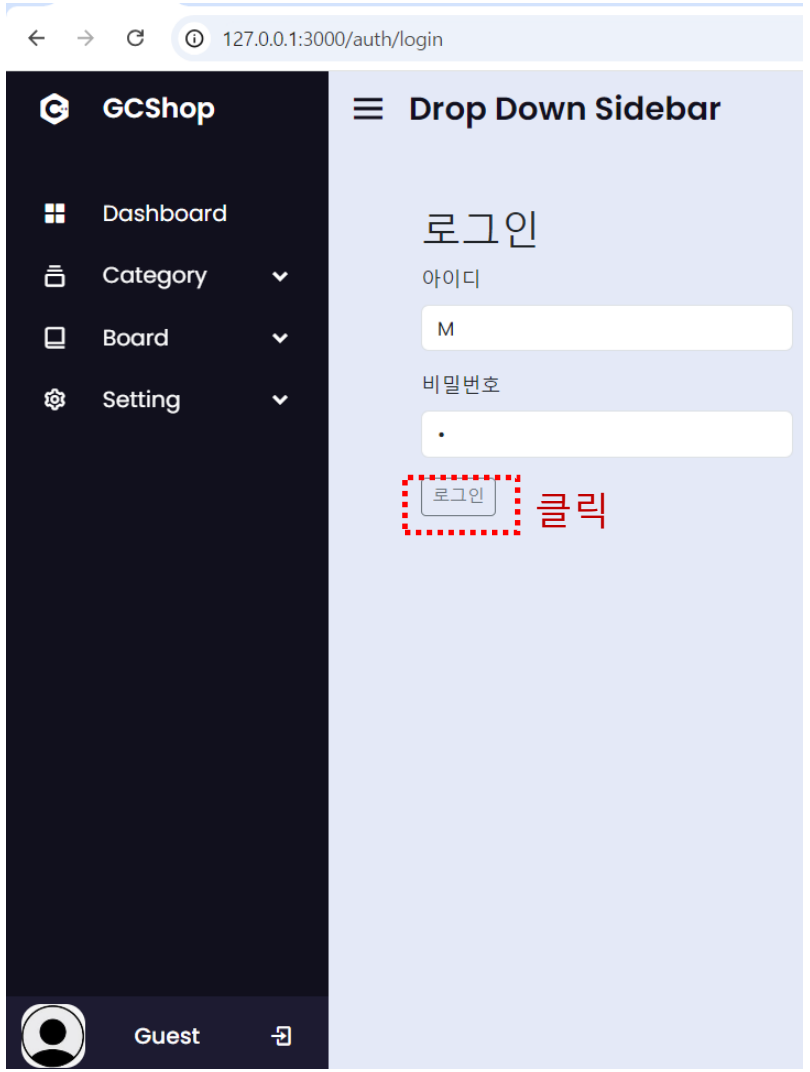


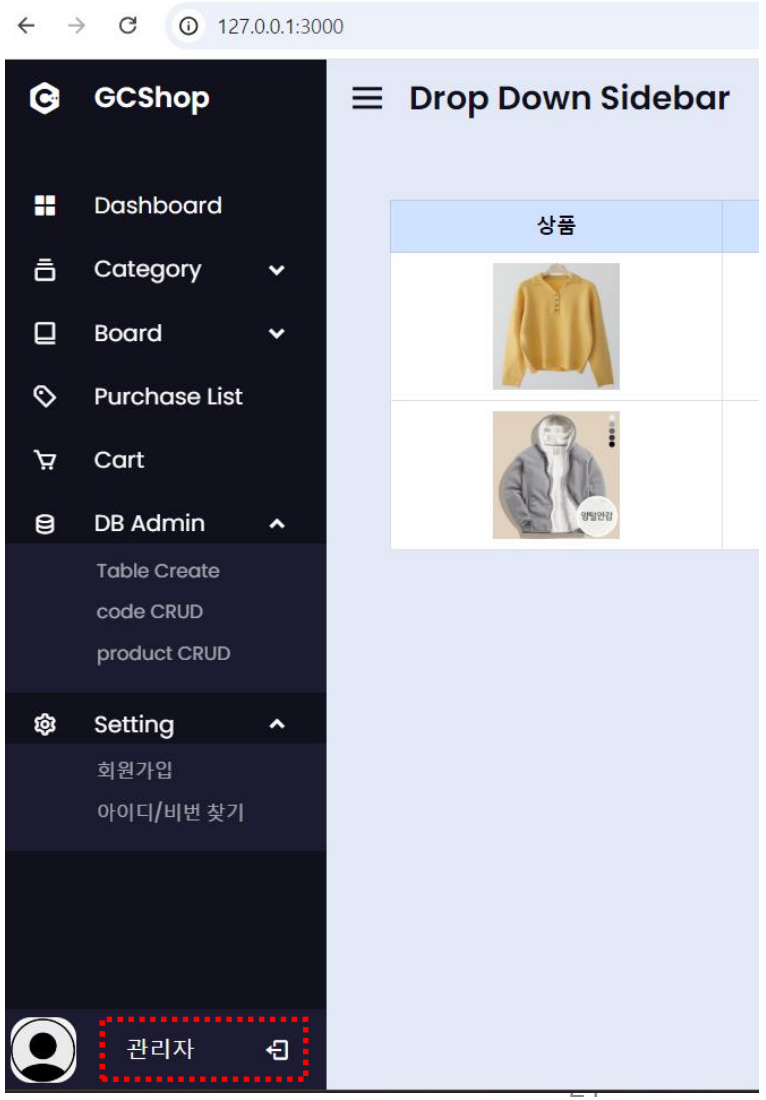


03 Project – Gachon Shop(gcshop)

로그인 전/후 화면

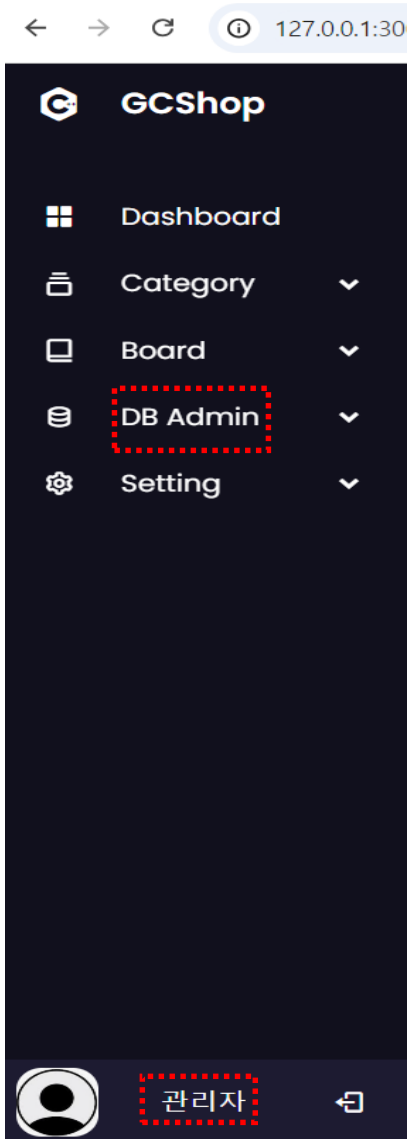


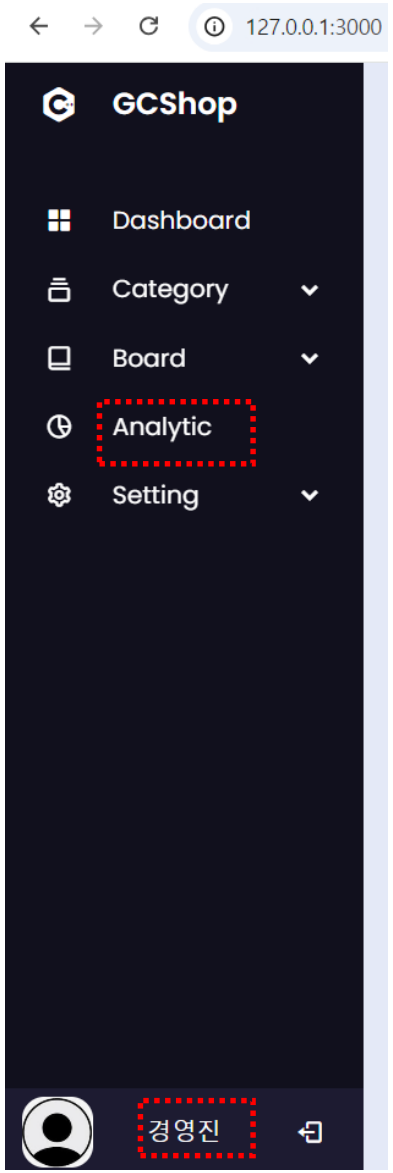


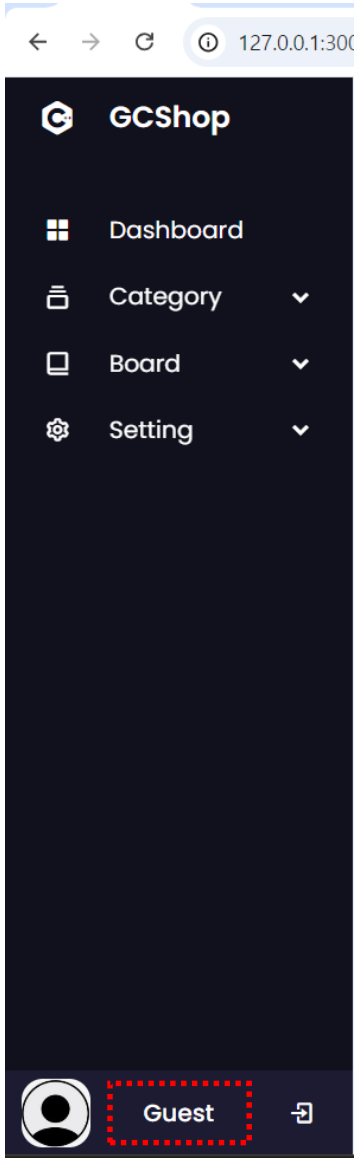


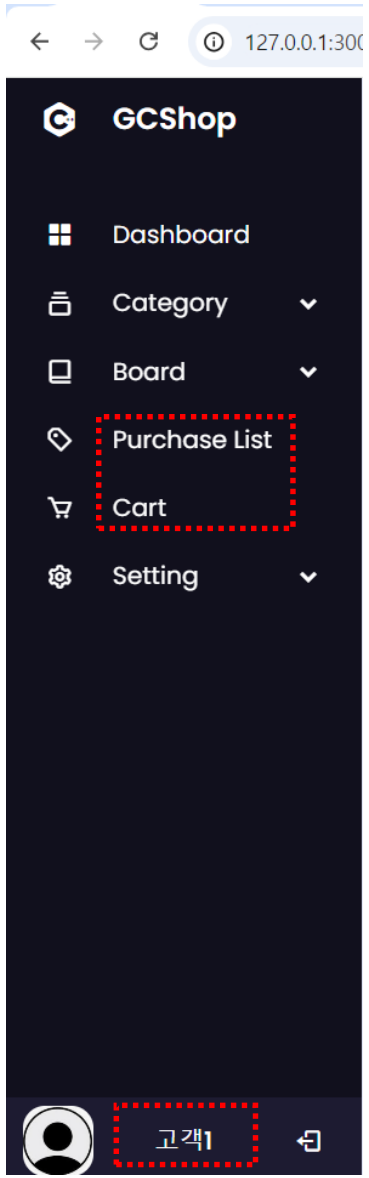
03 Project – Gachon Shop(gcshop)

관리자, 고객, 경영자, Guest 메뉴 차이









2. 로그인 기능 구현 순서

① main.js 작성

```
//express와 views 정의
const express = require('express')   ;
const app = express()    ;
app.set('views',__dirname + '/views');
app.set('view engine','ejs');

//사용자 정의 모듈
var rootRouter = require('./router/rootRouter');
var authRouter = require('./router/authRouter');
```

2. 로그인 기능 구현 ① main.js 작성

```
const express=require('express');
var session = require('express-session');
var MySqlStore = require('express-mysql-session')(session);
var bodyParser = require('body-parser');

var options = {
   host     : 'localhost',
   user     : 'nodejs',
   password : 'nodejs',
   database : 'webdb2024'
};
var sessionStore = new MySqlStore(options);
const app = express();
app.use(session({
    secret : 'keyboard cat',
    resave : false,
    saveUninitialized : true,
    store : sessionStore
 }));
app.set('views',__dirname + '/views');
app.set('view engine','ejs')
app.use(bodyParser.urlencoded({extended: false }));
```

```
const rootRouter = require('./router/rootRouter');
const authRouter = require('./router/authRouter');

app.use(express.static('public'));

app.use('/',rootRouter);
app.use('/auth',authRouter);


app.get('/favicon.ico', (req,res)=>res.writeHead(404));
app.listen(3000, ()=>console.log('Example app listening on port 3000'));
```

2. 로그인 기능 구현 순서

② /router/rootRouter.js 작성

```
const express= require('express');
const router= express.Router();

var root = require('../lib/root');

router.get('/', (req,res)=>{
    root.home(req,res)
})


module.exports = router;
```

2. 로그인 기능 구현 순서

③ /router/authRouter.js 작성

```
const express = require('express');
var router = express.Router()

var auth = require('../lib/auth');

router.get('/login',(req, res)=>{
    auth.login(req, res);
});

router.post('/login_process',(req, res)=>{
    auth.login_process(req, res);
});

router.get('/logout_process',(req,res)=>{
    auth.logout_process(req,res);
});

module.exports = router;
```

2. 로그인 기능 구현 순서

④ /lib/db.js 작성

```
var mysql = require('mysql');
var db = mysql.createConnection({
    host     : 'localhost',
    user     : 'nodejs',
    password : 'nodejs',
    database : 'webdb2024'
});
db.connect();
module.exports = db;
```

2. 로그인 기능 구현 순서

⑤ /lib/auth.js 작성

ⓐ 로그인 후 사진 옆에 로그인 한 사람의 이름이 나오도록 하는 변수

ⓑ mainfram에서 로그인 여부에 따라 변경되는 사항을 제어하기 위한 변수 로그인 전 후 아이콘 변경에 사용

ⓒ 메뉴에 따른 내용 변화

ⓓ 로그인 한 사람의 분류에 따라 변경되는 메뉴를 제어하기 위한 변수

```
var db = require('./db');
var sanitizeHtml = require('sanitize-html');

function authIsOwner(req,res){
    var name = 'Guest';
    var login = false;
    var cls = 'NON';
    if(req.session.is_logined){
        name = req.session.name;
        login = true;
        cls = req.session.cls ;
    }
    return {name,login,cls}
}

module.exports = {
    login : (req,res)=>{
            var {name, login, cls} = authIsOwner(req,res);

            var context = {
                    /*********** mainFrame.ejs에 필요한 변수 ***********/
                 ⓐ who : name,
                 ⓑ login : login,
                 ⓒ body : 'login.ejs',
                 ⓓ cls : cls
                };
                req.app.render('mainFrame',context, (err, html)=>{
                    res.end(html); })
    },
```

2. 로그인 기능 구현 순서

⑤ /lib/auth.js 작성

```
login_process : (req,res)=>{
        var post = req.body;
        var sntzedLoginid = sanitizeHtml(post.loginid);
        var sntzedPassword = sanitizeHtml(post.password);

        db.query('select count(*) as num from person where loginid = ? and password = ?',
       [sntzedLoginid,sntzedPassword],(error, results)=>{
            if (results[0].num === 1){
                db.query('select name, class,loginid, grade from person where loginid = ? and password = ?',
        [sntzedLoginid,sntzedPassword],(error, result)=>{
                    req.session.is_logined = true;
                    req.session.loginid = result[0].loginid
                    req.session.name = result[0].name
                    req.session.cls = result[0].class
                    req.session.grade = result[0].grade
                    res.redirect('/');
                })
            }
            else {  req.session.is_logined = false;
                    req.session.name = 'Guest';
                    req.session.cls = 'NON';
                    res.redirect('/');          }
        })      },
    logout_process : (req, res) => {
        req.session.destroy((err)=>{
           res.redirect('/');
        })     },
```

2. 로그인 기능 구현 순서

⑥ /lib/root.js 작성

```
const db = require('./db');
var sanitizeHtml = require('sanitize-html');

function authIsOwner(req,res){
    var name = 'Guest';
    var login = false;
    var cls = 'NON';
    if(req.session.is_logined){
        name = req.session.name;
        login = true;
        cls = req.session.cls ;
    }
    return {name,login,cls}
}
module.exports = {
    home : (req,res)=>{
         var {login, name, cls} = authIsOwner(req,res)
        var sql2 = ` select * from product;`
        db.query(sql2,(error,results)=>{
            var context = {
                        /*********** mainFrame.ejs에 필요한 변수 ***********/
                        who : name,
                        login: login,
                        body : 'test.ejs',
                        cls: cls  };
            res.render('mainFrame',context,(err,html)=>{
                res.end(html)
            });  //render end
        });  //query end
    },
```

2. 로그인 기능 구현 순서

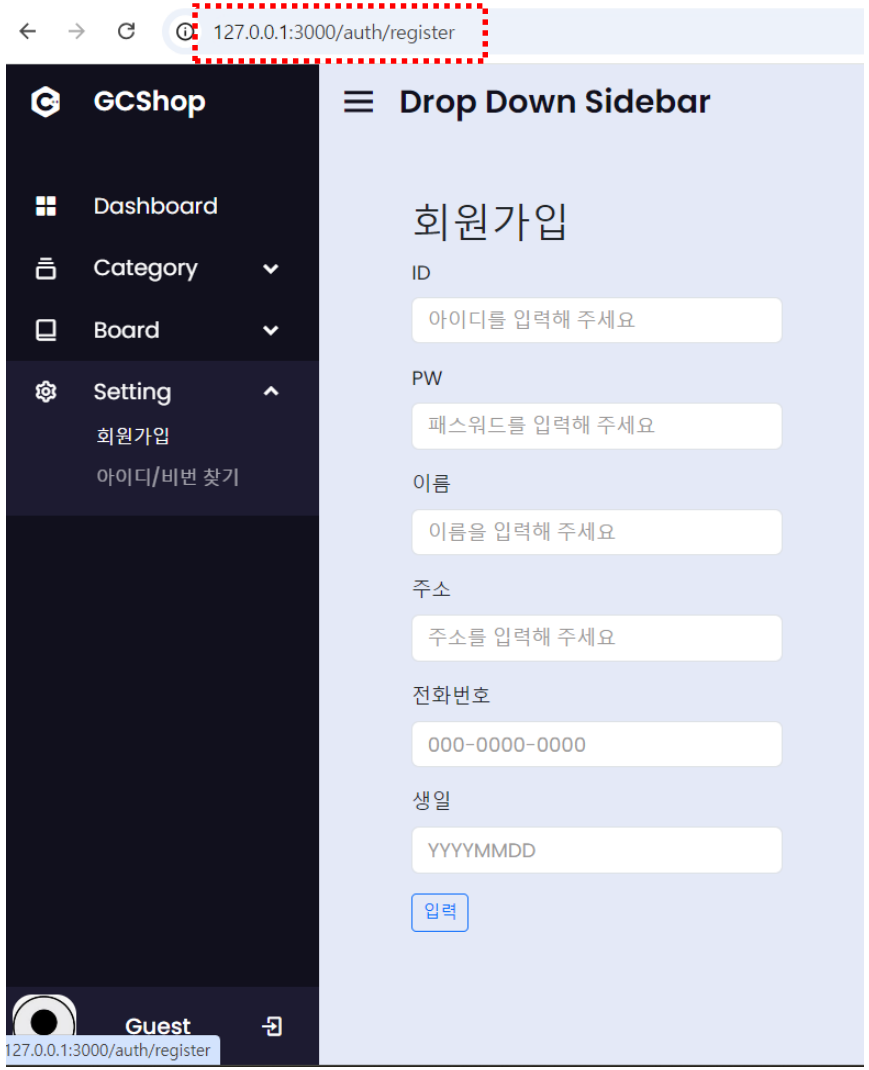
⑦ mainFrame.html을 mainFrame.ejs로 파일명 수정

⑧ 로그인 전후의 아이콘 변경을 위한 수정

⑨ 로그인 한 사람 분류에 따른 메뉴 변경을 위한 수정

3. 회원 가입

① 로그인 전에 setting메뉴의 회원 가입을 클릭하면 회원 가입 화면 나타남. 회원 가입 후 입력 버튼 클릭하면 root로 감.



입력 버튼 클릭하면 내용이 person 테이블에 저장

personCU.ejs

입력 시 : class ➔ "CST", grade ➔ "S"

3. 회원 가입

② 로그인 후 회원 가입을 클릭 하면 아무 반응도 없이 '/'로 redirect.