

DB Application HW 보고서

도서관 DB

컴퓨터학과 2018100022 심은정

Schema Diagram for library DB

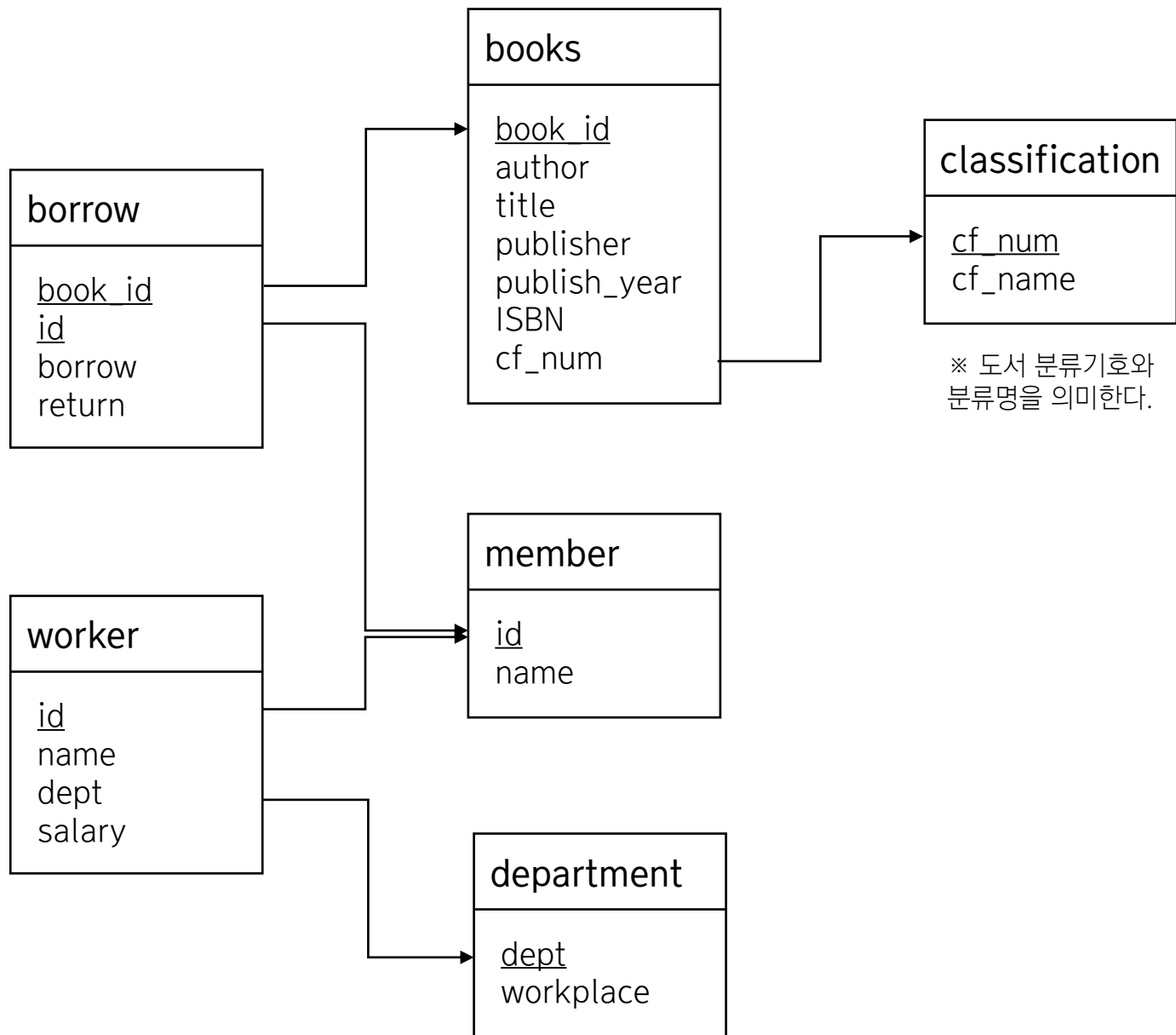


table 속성 (sql script)

```
create table classification(cf_num numeric(3,0) primary key, cf_name char(5));
```

```
create table books(book_id numeric(8,0), author varchar(15), title varchar(15),  
publisher varchar(10), publish_year numeric(4,0), ISBN numeric(13,0), cf_num  
numeric(3,0) references classification, primary key(book_id));
```

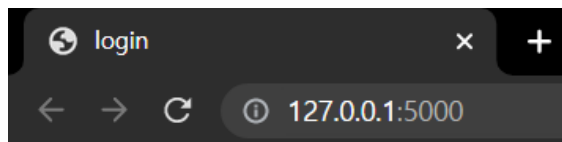
```
create table member(id varchar(8) primary key, name varchar(20));
```

```
create table department(dept varchar(20) primary key, workplace varchar(20));
```

```
create table worker(id varchar(8) references member, name varchar(20), dept varchar(20)  
references department, salary numeric(8,2), primary key(id));
```

```
create table borrow(book_id numeric(8,0) references books, id varchar(8) references  
member, borrow date, return date, primary key(book_id, id));
```

1 로그인



id:

name:

login

worker menu

sign up

```
@app.route('/')  
def main():  
    return render_template("login.html")
```

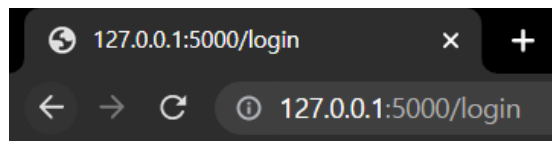
첫 화면으로, id와 회원 이름을 가지고 로그인한다.
관리자 전용 메뉴가 있는데 관리자 로그인을 따로 설정하지는 않았고 기능만 구현했다.

2 로그인 -> 3 홈 화면

```
@app.route('/login', methods=['POST'])
def login():
    id = request.form['id']
    name = request.form['name']
    cur.execute("select id, name from member where id = '{}' and name = '{}'"
                .format(id, name))
    result = cur.fetchall()

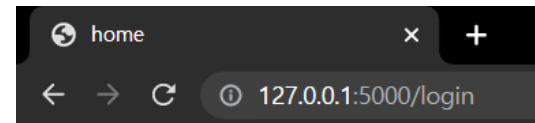
    try:
        if id == result[0][0] and name == result[0][1]:
            return render_template("home.html")
    except:
        return "ID나 회원명 다시 확인하세요"
```

입력된 id와 name으로 select해서 DB에서 받아온 값과 일치하면 home으로 넘긴다.



ID나 회원명 다시 확인하세요

ID나 회원명이 다를 때



도서 검색
책 제목 입력:

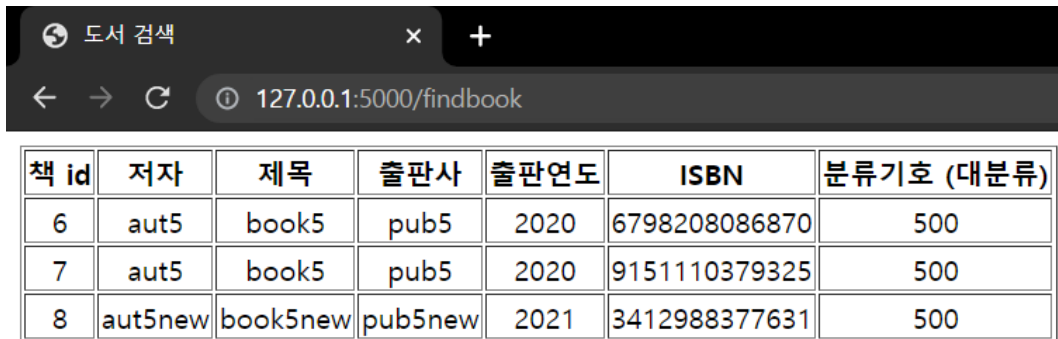
재고 검색
책 제목 입력:

대출기간 연장 신청
회원 id 입력:

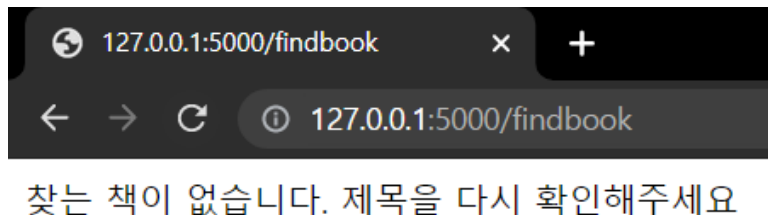
연장할 도서 id 입력:

정상적 로그인 후 홈 화면

3 (1) 도서 검색



책 id	저자	제목	출판사	출판연도	ISBN	분류기호 (대분류)
6	aut5	book5	pub5	2020	6798208086870	500
7	aut5	book5	pub5	2020	9151110379325	500
8	aut5new	book5new	pub5new	2021	3412988377631	500



찾는 책이 없습니다. 제목을 다시 확인해주세요

```
# 제목 검색
@app.route('/findbook', methods=['POST'])
def findbook():
    findtitle = request.form['title']
    cur.execute("select * from books where title like '%{}%';".format(findtitle))
    titleresult = cur.fetchall()

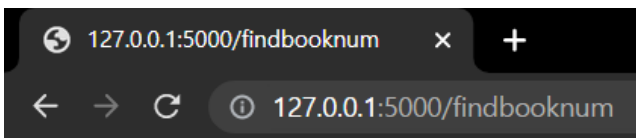
    try:
        if findtitle == titleresult[0][2]:
            return render_template("findbook.html", findtitle=titleresult)
    except IndexError:
        return "찾는 책이 없습니다. 제목을 다시 확인해주세요"
```

찾을 때 %title%을 써서 검색한 내용을 포함만 하고 있으면 다 출력하는 식으로 한다.
해당되면 findbook.html로 보내서 표를 띄운다.

3 (2) 재고 검색

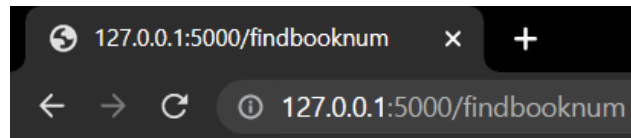
```
# 재고 현황
@app.route('/findbooknum', methods=['POST'])
def findbooknum():
    title = request.form['title2']
    cur.execute("select count(title) from books where title = '{}';".format(title))
    result = cur.fetchall()

    if result[0][0] != 0:
        return str(result[0][0])
    else: return "현재 도서관에 구비되어 있지 않습니다."
```



2

book5를 검색했을 때.
재고 검색의 경우 완전히 동일하게 검색해야 count에 포함시키도록 한다.



현재 도서관에 구비되어 있지 않습니다.

book12를 검색했을 때.
table에 존재하지 않으면 현재 도서관에 없는 책이라고 뜬다.

3 (3) 대출기간 연장 신청

대출 연장 확인

127.0.0.1:5000/borrowmore

반납일이 14일 후로 연장되었습니다.

도서 id	회원 id	대출일자	연장된 반납예정일
10	WK000001	2021-05-10	2021-06-07

```
flask=# select * from borrow;
```

book_id	id	borrow	return
4	MEM00004	2021-05-03	2021-05-17
5	MEM00001	2021-05-07	2021-05-21
6	MEM00001	2021-05-07	2021-05-21
9	MEM00003	2021-05-06	2021-05-20
10	WK000001	2021-05-10	2021-05-24

(5개 행)

연장 전 psql에서의 borrow table

```
flask=# select * from borrow;
```

book_id	id	borrow	return
4	MEM00004	2021-05-03	2021-05-17
5	MEM00001	2021-05-07	2021-05-21
6	MEM00001	2021-05-07	2021-05-21
9	MEM00003	2021-05-06	2021-05-20
10	WK000001	2021-05-10	2021-06-07

(5개 행)

연장 후 psql에서의 borrow table

```
# 대출연장
@app.route('/borrowmore', methods=['POST'])
def borrowmore():
    id = request.form['id']
    bookid = request.form['bookid']
    cur.execute("select id, book_id from borrow where id = '{}' and book_id = '{}";".format(id, bookid))
    result = cur.fetchall()

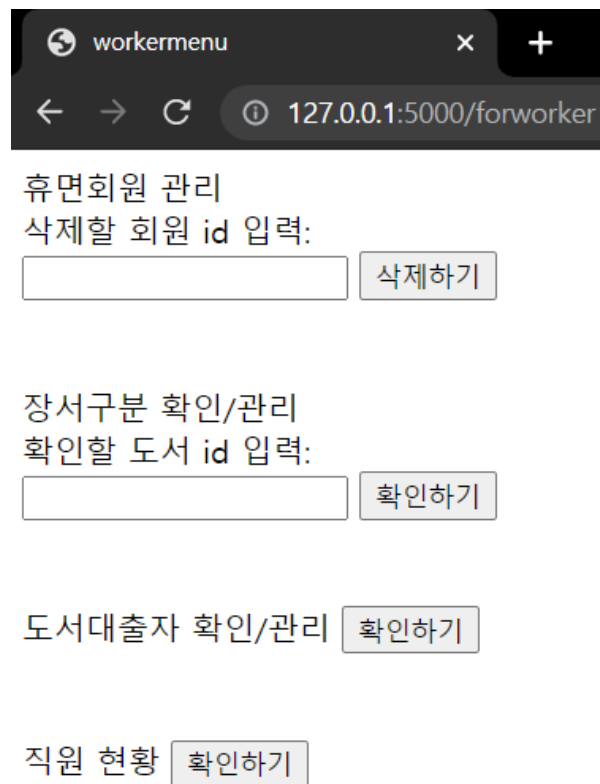
    try:
        if id == result[0][0] and bookid == str(result[0][1]):
            cur.execute("update borrow set return = return + interval '14 day' where id = '{}' and book_id = '{}";".format(id, bookid))
            connect.commit()
            cur.execute("select * from borrow where id = '{}' and book_id = '{}";".format(id, bookid))
            tableresult = cur.fetchall()
            return render_template("borrowmore.html", more = tableresult)

    except IndexError: return "찾는 책이 없습니다. 회원 id나 도서 id를 다시 확인해주세요"
```

‘date’ type로 저장했기 때문에 update로 interval ‘14 days’를 더해주면
날짜 기준으로 덧셈이 이루어져서 5월 38일이 아닌 6월 7일로, 정상적으로 계산된다.

날짜 update 후 borrow table의 해당 부분을 표로 만든
borrowmore.html로 연결

4 관리자 메뉴



workermenu

127.0.0.1:5000/forworker

휴면회원 관리
삭제할 회원 id 입력:

삭제하기

장서구분 확인/관리
확인할 도서 id 입력:

확인하기

도서대출자 확인/관리

확인하기

직원 현황

확인하기

관리자 메뉴에는 회원 삭제, 도서 분류명 확인, 대출자 관리, 직원 현황 보기가 있다.

4 (1) 회원 삭제

test용으로 (aaa, aaa)라는 member를 생성하고 id = 'aaa' 인 tuple을 입력했다.

```
flask=# select * from member;
 id | name
-----+-----
WK000001 | worker1
WK000002 | worker2
MEM00001 | member1
MEM00002 | member2
MEM00003 | member3
MEM00004 | member4
WK000003 | worker3
WK000004 | worker4
WK000005 | worker5
MEM00005 | member5
MEM00006 | member6
aaa   | aaa
(12개 행)
```

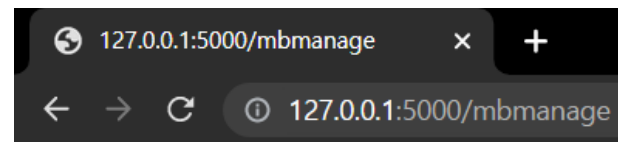


```
flask=# select * from member;
 id | name
-----+-----
WK000001 | worker1
WK000002 | worker2
MEM00001 | member1
MEM00002 | member2
MEM00003 | member3
MEM00004 | member4
WK000003 | worker3
WK000004 | worker4
WK000005 | worker5
MEM00005 | member5
MEM00006 | member6
(11개 행)
```

입력된 id를 받아와서 해당 id가 member에 존재하면 tuple을 삭제한다.

```
@app.route('/mbmanage', methods=['POST'])
def mbmanage():
    id = request.form['id']
    cur.execute("select id from member where id = '{}';".format(id))
    delid = cur.fetchall()

    try:
        if id == delid[0][0]:
            cur.execute("delete from member where id = '{}';".format(id))
            connect.commit()
            return "해당 휴면회원이 삭제되었습니다."
    except IndexError:
        return "입력된 회원 id를 다시 확인해주세요."
```



해당 휴면회원이 삭제되었습니다.

4 (2) 장서 구분/확인

book manage		
127.0.0.1:		
도서 id	분류기호	분류명
1	100	철학

book manage		
127.0.0.1:50		
도서 id	분류기호	분류명
7	500	기술과학

book manage		
127.0.0.1		
도서 id	분류기호	분류명
9	800	문학

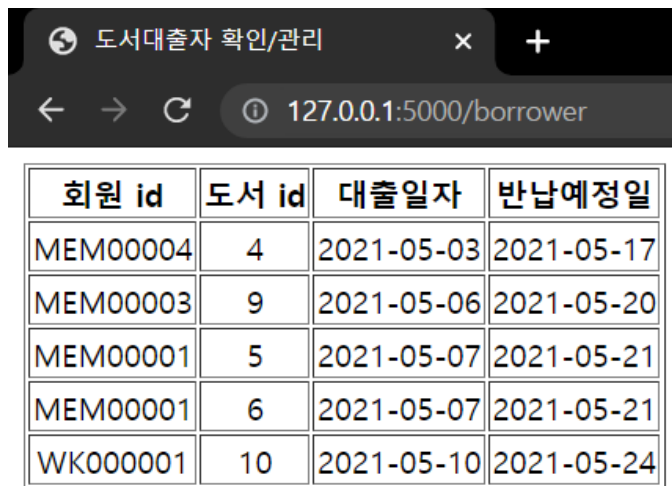
입력된 도서 id와 함께 그 도서의 분류기호, 분류명을 표로 출력함

```
@app.route('/bookmanage', methods=['POST'])
def bookmanage():
    bookid = request.form['bookid']
    cur.execute("select book_id, books.cf_num, cf_name from books, classification where books.cf_num = classification.cf_num and book_id = {};".format(bookid))
    bookmg = cur.fetchall()

    try:
        if bookid == str(bookmg[0][0]):
            return render_template("bookmanage.html", bookmanage = bookmg)
    except IndexError:
        return "입력된 도서 id를 다시 확인해주세요."
```

bookid는 문자이고 bookmg[0][0]은 정수형이라 자료형을 같게 해서 비교해주었고
books, classification 두 table의 cartesian product로부터 값을 구했다.

4 (3) 도서대출자 확인



The screenshot shows a web browser window with the title '도서대출자 확인/관리' (Book Borrower Confirmation/Management). The address bar displays '127.0.0.1:5000/borrower'. Below the browser window is a table with the following data:

회원 id	도서 id	대출일자	반납예정일
MEM00004	4	2021-05-03	2021-05-17
MEM00003	9	2021-05-06	2021-05-20
MEM00001	5	2021-05-07	2021-05-21
MEM00001	6	2021-05-07	2021-05-21
WK000001	10	2021-05-10	2021-05-24

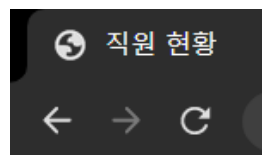
```
@app.route('/borrower', methods=['POST'])
def borrowermanage():

    cur.execute("select id, book_id, borrow, return from member natural join borrow order by borrow;")
    borrowermg = cur.fetchall()

    return render_template("borrowerinfo.html", borrower = borrowermg)
```

member와 borrow를 natural join하여 book_id가 공통인 것들, 즉 현재 도서 대출 중인 회원들의 대출 현황과 대출일자, 반납예정일을 대출일자가 가장 빠른 것부터 표로 출력하였다.

4 (4) 직원 현황 확인



직원 현황 보기

부서	직원 수
행정	2
사서	3

```
flask=# select * from worker;
      id      | name   | dept | salary
-----+-----+-----+-----
WK000001 | worker1 | 행정 | 3210.00
WK000002 | worker2 | 사서 | 2870.00
WK000003 | worker3 | 사서 | 2890.00
WK000004 | worker4 | 사서 | 3000.00
WK000005 | worker5 | 행정 | 2900.00
(5개 행)
```

```
@app.route('/workers', methods=['POST'])
def workermanage():

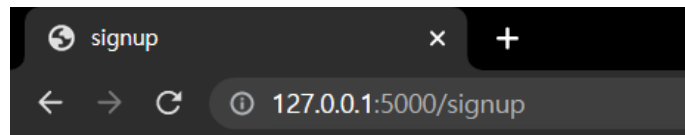
    cur.execute("select dept, (select count(*) from worker where worker.dept = department.dept) from department;")
    workermg = cur.fetchall()

    return render_template("workerinfo.html", worker = workermg)
```

직원들이 속한 부서 / 한 부서에 속한 직원들을 count한 것을 함께 표로 출력한다.

=> 부서별 직원들을 출력

5 회원가입



ID 중복 검사 버튼 클릭 시 반응 없으면 가입 가능

중복 검사 확인용 id 입력:

ID 중복 검사

id:

name:

회원가입

참고사항

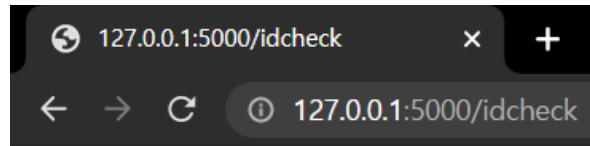
아이디는 문자, 숫자 혼용 가능 8자 이내

사용자명은 문자, 숫자 혼용 가능 20자 이내

로그인하러 가기

회원은 고유의 id로 구분되기 때문에 중복일 수 없으므로 확인하는 검사가 필요하다.
그리고 나서 id와 name으로 회원가입을 하여 DB에 등록한다.

5 (1) ID 중복 검사



아이디 중복 / 다시 확인

id를 이미 존재하는 값인 WK000001로 입력했을 때

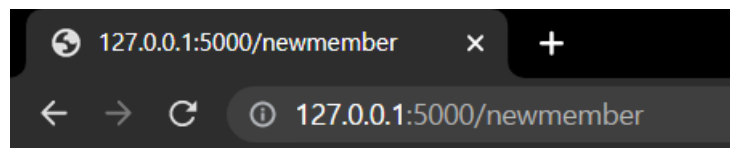
```
#sign up page. html
@app.route('/idcheck', methods=['POST'])
def idcheck():
    id = request.form["id"]
    cur.execute("select id from member where id = '{}';".format(id))
    findid = cur.fetchall()
    print(findid)
    try:
        if id == findid[0][0]:
            return "아이디 중복 / 다시 확인"
    except IndexError:
        return render_template("sign_up_page.html")
```

id가 중복되지 않는 경우라면 해당 페이지를 그대로 띄우도록 하였다.

5 (2) 회원가입

id:

name:

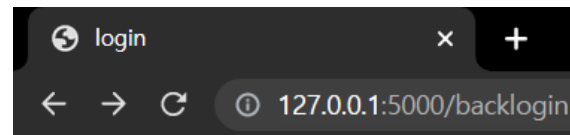


가입 완료 / 이전 화면으로 돌아가 로그인하세요

(aaa, hello) 계정을 만들었다.

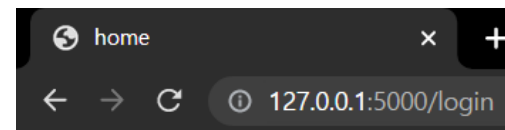
```
flask=# select * from member;
 id | name
----+-----
WK000001 | worker1
WK000002 | worker2
MEM000001 | member1
MEM000002 | member2
MEM000003 | member3
MEM000004 | member4
WK000003 | worker3
WK000004 | worker4
WK000005 | worker5
MEM000005 | member5
MEM000006 | member6
aaa | hello
(12개 행)
```

psql에서 확인하면 새로운 계정이 table에 올라와 있다.



id:

name:



도서 검색
책 제목 입력:

재고 검색
책 제목 입력:

대출기간 연장 신청
회원 id 입력:

연장할 도서 id 입력:

정상적으로 로그인할 수 있다.