데이터베이스시스템 및 활용

개요

* Python 3의 flask, pymysql을 사용하였다 웹페이지들은 templates 디렉토리 안에 있는 html 파일들로 구성되어 있다. MariaDB를 사용하였다.
* 공공데이터포털에 있는 병원, 약국 OpenAPI를 사용하였고, 지도는 Kakao API를 사용하였다.
* 테이블의 스키마와 데이터는 dump file로 dr\_hyuDB에 저장되었다.
* 현재 데이터에는 clinic 테이블에 병원(5km)의 데이터, pharmacy 테이블에 약국(5km)의 데이터, user 테이블의 customers.csv파일(중복된 아이디는 제외함)의 데이터가 저장되어 있다.
* 모든 유저의 타입은 저장 설정되지 않은 상태이다.
* 테이블은 clinic(병원 정보 저장), clinic\_complete(병원의 처방기록 저장), clinic\_reserve(병원의 환자 예약 기록 저장), clinic\_user(유저가 어떤 병원과 연결이 되어있는 지 저장), patient\_favorite(환자의 자주 가능 병원 리스트 저장), pharmacy (약국 정보 저장), pharmacy\_user(유저와 유저의 약국 정보 저장), pharmacy\_reserve(약국의 예약정보 저장), prescription(처방전 저장), type(유저의 타입을 저장), user(유저 정보 저장)가 있다.
* 약국 혹은 병원으로 로그인하게 될 경우, 약국/병원은 상위 데이터부터 순차적으로 유저와 매칭된다.

각 함수 설명

* + DBConnection 함수는 customers.csv 파일에서 필요한 column만 골라서 user table에 insert하는 함수이다. “customers.csv” 파일이 존재하고 user 테이블이 비었을 경우에만 삽입한다.

def DBconnection():

    sql = "SELECT COUNT(\*) FROM USER"

    cur = conn.cursor()

    cur.execute(sql)

    res = cur.fetchall()[0][0]

    if res == 0:

        try:

            f = open("customers.csv", 'r', encoding='utf-8')

            if f != None:

                rdr = csv.reader(f)

                header = next(rdr)

                for line in rdr:

                    cur.execute("INSERT IGNORE INTO user (name, phone, local, domain, passwd, lat, lng) VALUES ('{}','{}', '{}', '{}', '{}', '{}', '{}')".format(line[0], line[1], line[2], line[3], line[4], float(line[6]), float(line[7])))

                    conn.commit()

        except Exception as ex:

            print(ex)

        print("initial DB set FINISHED")

* + getData 함수는 Open API를 불러오는 함수(OpenAPI\_Clinic, OpenAPI\_Pharmacy)를 호출한다. 초기에 두 테이블에 아무 데이터도 들어가있지않을 경우에만 실행된다.

def getData():

    cur = conn.cursor()

    cur.execute("SELECT COUNT(\*) FROM CLINIC")

    cnt = cur.fetchall()[0][0]

    if cnt == 0:

        OpenAPI\_Clinic(0, 5)

    cur.execute("SELECT \* FROM pharmacy")

    res = cur.fetchall()

    print(len(res))

    if len(res) == 0:

        OpenAPI\_Pharmacy(0, 5)

* + OpenAPI\_Clinic은 한양대를 기준으로 거리 범위(단위 km) [first, second) 안에 있는 데이터를 불러오는 함수이다. haversine이라는 모듈을 사용해서 거리를 계산하였다. 프로그램에 필요한 attribute만 뽑아서 clinic 테이블에 삽입하였다. 또한 중복되는 데이터가 없도록 “INSERT IGNORE” sql문을 써주었다.

def OpenAPI\_Clinic(first, second):

    hanyang = (37.5585146, 127.0331892)

    cur = conn.cursor()

    for i in range(1, 6):

        URL = "http://apis.data.go.kr/B552657/HsptlAsembySearchService/getHsptlBassInfoInqire?serviceKey=pmxTNo7r3bisQw121DLdNQRyKuuird7xfX3MUhAqJWwH9H2UZ6D%2FqPZsyEgsegkEJStT4YTfcLVkWipJKnVElA%3D%3D&numOfRows=100&pageNo={}".format(i)

        request = ul.Request(URL)

        response = ul.urlopen(request)

        rescode = response.getcode()

        if rescode == 200:

            result\_data = response.read()

            rD = xmltodict.parse(result\_data)

            rDJ = json.dumps(rD)

            rDD = json.loads(rDJ)

            tdata = rDD["response"]['body']['items']['item']

            for j in range(100):

                data = tdata[j]

                if 'wgs84Lat' not in data.keys() or 'wgs84Lon' not in data.keys():

                    continue

                else:

                    place = (float(data['wgs84Lat']), float(data['wgs84Lon']))

                    if haversine(hanyang, place) <= second and haversine(hanyang,place) > first:

                        if 'dutyTime1s' not in data.keys():

                            data['dutyTime1s'] = data['dutyTime1c'] = '0000'

                        if 'dutyTime2s' not in data.keys():

                            data['dutyTime2s'] = data['dutyTime2c'] = '0000'

                        if 'dutyTime3s' not in data.keys():

                            data['dutyTime3s'] = data['dutyTime3c'] = '0000'

                        if 'dutyTime4s' not in data.keys():

                            data['dutyTime4s'] = data['dutyTime4c'] = '0000'

                        if 'dutyTime5s' not in data.keys():

                            data['dutyTime5s'] = data['dutyTime5c'] = '0000'

                        if 'dutyTime6s' not in data.keys():

                            data['dutyTime6s'] = data['dutyTime6c'] = '0000'

                        if 'dutyTime7s' not in data.keys():

                            data['dutyTime7s'] = data['dutyTime7c'] = '0000'

                        sql = '''INSERT IGNORE INTO CLINIC (name, hpid, lat, lng, treatment, dutyTime1s,

                                    dutyTime1c, dutyTime2s, dutyTime2c, dutyTime3s, dutyTime3c, dutyTime4s,

                                    dutyTime4c, dutyTime5s, dutyTime5c, dutyTime6s, dutyTime6c, dutyTime7s, dutyTime7c, address)

                                    VALUES ("{}","{}",{},{},"{}",{},{},{},{},{},{},{},{},{},{},{},{},{},{},"{}")'''.format(

                                    data['dutyName'], data['hpid'], data['wgs84Lat'], data['wgs84Lon'], data['dgidIdName'],

                                    data['dutyTime1s'], data['dutyTime1c'], data['dutyTime2s'], data['dutyTime2c'],

                                    data['dutyTime3s'], data['dutyTime3c'], data['dutyTime4s'], data['dutyTime4c'],

                                    data['dutyTime5s'], data['dutyTime5c'], data['dutyTime6s'], data['dutyTime6c'],

                                    data['dutyTime7s'], data['dutyTime7c'], data['dutyAddr'])

                        cur.execute(sql)

                        conn.commit()

    cur.close()

* + OpenAPI\_Pharmacy는 한양대를 기준으로 거리 범위가 [first,second) 안에 있는 약국들의 데이터를 찾아 pharmacy 테이블에 삽입하는 함수이다. OpenAPI\_Clinic과 같은 방식으로 구현하였다.

def OpenAPI\_Pharmacy(first, second):

    hanyang = (37.5585146, 127.0331892)

    cur = conn.cursor()

    for i in range(1,6):

        URL = "http://apis.data.go.kr/B552657/ErmctInsttInfoInqireService/getParmacyListInfoInqire?serviceKey=pmxTNo7r3bisQw121DLdNQRyKuuird7xfX3MUhAqJWwH9H2UZ6D%2FqPZsyEgsegkEJStT4YTfcLVkWipJKnVElA%3D%3D&numOfRows=100&pageNo={}".format(i)

        request = ul.Request(URL)

        response = ul.urlopen(request)

        rescode = response.getcode()

        if rescode == 200:

            result\_data = response.read()

            rD = xmltodict.parse(result\_data)

            rDJ = json.dumps(rD)

            rDD = json.loads(rDJ)

            tdata = rDD["response"]['body']['items']['item']

            for j in range(100):

                data = tdata[j]

                if 'wgs84Lat' not in data.keys() or 'wgs84Lon' not in data.keys():

                    continue

                else:

                    place = (float(data['wgs84Lat']), float(data['wgs84Lon']))

                    if haversine(hanyang, place) <= second and haversine(hanyang,place) > first:

                        sql = "INSERT IGNORE into pharmacy (name, hpid, lat, lng, address) VALUES ('{}','{}','{}','{}','{}')".format(data['dutyName'],data['hpid'], data['wgs84Lat'], data['wgs84Lon'], data['dutyAddr'])

                        cur.execute(sql)

                        conn.commit()

    cur.close()

* + signIn은 로그인 하는 함수이다. Local, domain, password, type을 입력을 받는다. 로그인 정보가 맞지 않는 경우에는 ‘sign in failed’라는 문자열을 출력해주어 다시 입력받도록 하였다. 로그인을 하였으나, 타입(라디오 버튼을 모두 선택하지 않고 sumbit하거나, not determined 버튼을 클릭한 후 submit하는 경우)이 정해져 있지 않은 경우에는 타입을 정하는 화면(user\_type.html)을 출력해주었다. 타입이 정해져 있을 경우에는 로그인할 때, 같이 타입을 입력 받아 일치하는 경우 각 유저타입에 맞는 화면을 띄워주었다.

@app.route('/sign\_in', methods=['GET', 'POST'])

def signIn():

    global user, clinic, pharmacy

    user = {}

    clinic = {}

    pharmacy = {}

    result = ""

    if request.method == 'POST':

        cur = conn.cursor(pymysql.cursors.DictCursor)

        sql = "SELECT \* FROM USER WHERE local='{}' AND domain='{}' AND passwd='{}'".format(request.form['local'], request.form['domain'], request.form['passwd'])

        cur.execute(sql)

        user = cur.fetchall()

        if len(user) == 0:

            return render\_template("sign\_in.html", result="Sign in Failed. Try Again")

        user = user[0]

        sql = "SELECT \* FROM TYPE WHERE ulocal='{}' AND udomain='{}'".format(user['local'], user['domain'])

        cur.execute(sql)

        userType = cur.fetchall()

        if len(userType) == 0:

            sql = "INSERT INTO TYPE (ulocal, udomain) VALUES ('{}', '{}')".format(user['local'], user['domain'])

            cur.execute(sql)

            conn.commit()

            return render\_template("user\_type.html")

        userType = userType[0]

        if request.form['type'] == 'none':

            return render\_template("user\_type.html")

        elif request.form['type'] == 'patient' and userType['patient'] == 1:

            return render\_template("patient.html", name=user['name'])

        elif request.form['type'] == 'clinic' and userType['clinic'] == 1:

            cur = conn.cursor(pymysql.cursors.DictCursor)

            sql = "SELECT \* FROM clinic c WHERE c.hpid=(SELECT cu.cid FROM clinic\_user cu WHERE cu.ulocal='{}' AND cu.udomain='{}') and c.name=(SELECT cu.cname FROM clinic\_user cu WHERE cu.ulocal='{}' and cu.udomain='{}')".format(request.form['local'], request.form['domain'], request.form['local'], request.form['domain'])

            cur.execute(sql)

            clinic = cur.fetchall()[0]

            OpenTime = ""

            if clinic['dutyTime1s'] != '0':

                OpenTime += " Mon: " + clinic['dutyTime1s'][:-2] + ':' + clinic['dutyTime1s'][-2:] + " ~ " + clinic['dutyTime1c'][:-2] + ':' + clinic['dutyTime1c'][-2:]

            if clinic['dutyTime2s'] != '0':

                OpenTime += "\n Tue: " + clinic['dutyTime2s'][:-2] + ':' + clinic['dutyTime2s'][-2:] + " ~ " + clinic['dutyTime2c'][:-2] + ':' + clinic['dutyTime2c'][-2:]

            if clinic['dutyTime3s'] != '0':

                OpenTime += "\n Wed: " + clinic['dutyTime3s'][:-2] + ':' + clinic['dutyTime3s'][-2:] + " ~ " + clinic['dutyTime3c'][:-2] + ':' + clinic['dutyTime3c'][-2:]

            if clinic['dutyTime4s'] != '0':

                OpenTime += "\n Thu: " + clinic['dutyTime4s'][:-2] + ':' + clinic['dutyTime4s'][-2:] + " ~ " + clinic['dutyTime4c'][:-2] + ':' + clinic['dutyTime4c'][-2:]

            if clinic['dutyTime5s'] != '0':

                OpenTime += "\n Fri: " + clinic['dutyTime5s'][:-2] + ':' + clinic['dutyTime5s'][-2:] + " ~ " + clinic['dutyTime5c'][:-2] + ':' + clinic['dutyTime5c'][-2:]

            if clinic['dutyTime6s'] != '0':

                OpenTime += "\n Sat: " + clinic['dutyTime6s'][:-2] + ':' + clinic['dutyTime6s'][-2:] + " ~ " + clinic['dutyTime6c'][:-2] + ':' + clinic['dutyTime6c'][-2:]

            if clinic['dutyTime7s'] != '0':

                OpenTime += "\n Sun: " + clinic['dutyTime7s'][:-2] + ':' + clinic['dutyTime7s'][-2:] + " ~ " + clinic['dutyTime7c'][:-2] + ':' + clinic['dutyTime7c'][-2:]

            clinic['time'] = OpenTime

            clinic['dis'] = haversine((user['lat'], user['lng']), (clinic['lat'], clinic['lng']))

            return render\_template("clinic.html", distance=clinic['dis'],name = clinic['name'], hpid = clinic['hpid'], number=clinic['numDoctor'], treatment=clinic['treatment'], lat=clinic['lat'], lng=clinic['lng'], location=clinic['address'], time=clinic['time'])

        elif request.form['type'] == 'store' and userType['pharmacy'] == 1:

            cur = conn.cursor(pymysql.cursors.DictCursor)

            sql = "SELECT \* FROM pharmacy p WHERE p.hpid=(SELECT pu.phid FROM pharmacy\_user pu WHERE pu.ulocal='{}' AND pu.udomain='{}') and p.name=(SELECT pu.phname FROM pharmacy\_user pu WHERE pu.ulocal='{}' and pu.udomain='{}')".format(user['local'], user['domain'], user['local'], user['domain'])

            cur.execute(sql)

            pharmacy = cur.fetchall()[0]

            return render\_template("store.html", name=pharmacy['name'], hpid=pharmacy['hpid'], lat=pharmacy['lat'], lng=pharmacy['lng'], location=pharmacy['address'])

        else:

            result = "Choose Type again"

    return render\_template("sign\_in.html", result=result)

* + 로그인을 했을 때, 타입이 정해지지 않았을 때 실행되는 함수이다. 타입은 복수선택이 가능하다. 타입을 정했을 경우, type 테이블에 유저타입과 type의 정보를 넣는다. 이후, 어떤 타입으로 로그인할지 정할 수 있는 화면(page\_type.html)으로 넘어간다. 타입은 환자, 병원, 약국, 기타가 있다.

@app.route('/user\_type', methods=['GET', 'POST'])

def userType():

    global cnt\_user\_clinic, user, cnt\_user\_phar

    if request.method == 'POST':

        cur = conn.cursor()

        type\_list = request.form.getlist('type')

        print(type\_list)

        if 'clinic' in type\_list:

            sql = "INSERT into clinic\_user (ulocal, udomain, cid, cname) SELECT '{}', '{}', c.hpid, c.name FROM clinic c LIMIT {}, 1".format(user['local'], user['domain'], cnt\_user\_clinic)

            cur.execute(sql)

            conn.commit()

            cnt\_user\_clinic += 1

        if 'pharmacy' in type\_list:

            sql = "INSERT into pharmacy\_user (ulocal, udomain, phid, phname) SELECT '{}', '{}', ph.hpid, ph.name FROM pharmacy ph LIMIT {}, 1".format(user['local'], user['domain'], cnt\_user\_phar)

            cur.execute(sql)

            conn.commit()

            cnt\_user\_phar += 1

        if len(type\_list) == 1:

            sql = "UPDATE type SET {}=TRUE WHERE ulocal='{}' AND udomain='{}'".format(type\_list[0], user['local'], user['domain'])

        elif len(type\_list) == 2:

            sql = "UPDATE type SET {}=TRUE, {}=TRUE WHERE ulocal='{}' AND udomain='{}'".format(type\_list[0], type\_list[1], user['local'], user['domain'])

        elif len(type\_list) == 3:

            sql = "UPDATE type SET {}=TRUE, {}=TRUE, {}=TRUE WHERE ulocal='{}' AND udomain='{}'".format(type\_list[0], type\_list[1], type\_list[2], user['local'], user['domain'])

        elif len(type\_list) == 4:

            sql = "UPDATE type SET {}=TRUE, {}=TRUE, {}=TRUE, {}=TRUE WHERE ulocal='{}' AND udomain='{}'".format(type\_list[0], type\_list[1], type\_list[2],type\_list[3], user['local'], user['domain'])

        else:

            return render\_template("user\_type.html", result="CHOOSE AT LEAST ONE")

        cur.execute(sql)

        conn.commit()

        return render\_template("page\_type.html")

    return render\_template("user\_type.html", result="")

* + 유저의 타입을 정한 후, 어떤 타입으로 로그인할 지 정하는 함수이다. 선택지는 환자, 병원, 상점(약국)이 있고, 그 중 하나를 선택하면 그 화면으로 넘어간다.

@app.route('/page\_type', methods=['GET', 'POST'])

def pageType():

    global clinic, user, pharmacy

    if request.method == "GET":

        page\_type = request.args.get('type')

        if page\_type == 'patient':

            return render\_template("patient.html", name=user['name'])

        elif page\_type == 'clinic':

            cur = conn.cursor(pymysql.cursors.DictCursor)

            sql = "SELECT \* FROM clinic c WHERE c.hpid=(SELECT cu.cid FROM clinic\_user cu WHERE cu.ulocal='{}' AND cu.udomain='{}') and c.name=(SELECT cu.cname FROM clinic\_user cu WHERE cu.ulocal='{}' and cu.udomain='{}')".format(user['local'], user['domain'], user['local'], user['domain'])

            cur.execute(sql)

            clinic = cur.fetchall()[0]

            OpenTime = ""

            if clinic['dutyTime1s'] != '0':

                OpenTime += " Mon: " + clinic['dutyTime1s'][:-2] + ':' + clinic['dutyTime1s'][-2:] + " ~ " + clinic['dutyTime1c'][:-2] + ':' + clinic['dutyTime1c'][-2:]

            if clinic['dutyTime2s'] != '0':

                OpenTime += "\n Tue: " + clinic['dutyTime2s'][:-2] + ':' + clinic['dutyTime2s'][-2:] + " ~ " + clinic['dutyTime2c'][:-2] + ':' + clinic['dutyTime2c'][-2:]

            if clinic['dutyTime3s'] != '0':

                OpenTime += "\n Wed: " + clinic['dutyTime3s'][:-2] + ':' + clinic['dutyTime3s'][-2:] + " ~ " + clinic['dutyTime3c'][:-2] + ':' + clinic['dutyTime3c'][-2:]

            if clinic['dutyTime4s'] != '0':

                OpenTime += "\n Thu: " + clinic['dutyTime4s'][:-2] + ':' + clinic['dutyTime4s'][-2:] + " ~ " + clinic['dutyTime4c'][:-2] + ':' + clinic['dutyTime4c'][-2:]

            if clinic['dutyTime5s'] != '0':

                OpenTime += "\n Fri: " + clinic['dutyTime5s'][:-2] + ':' + clinic['dutyTime5s'][-2:] + " ~ " + clinic['dutyTime5c'][:-2] + ':' + clinic['dutyTime5c'][-2:]

            if clinic['dutyTime6s'] != '0':

                OpenTime += "\n Sat: " + clinic['dutyTime6s'][:-2] + ':' + clinic['dutyTime6s'][-2:] + " ~ " + clinic['dutyTime6c'][:-2] + ':' + clinic['dutyTime6c'][-2:]

            if clinic['dutyTime7s'] != '0':

                OpenTime += "\n Sun: " + clinic['dutyTime7s'][:-2] + ':' + clinic['dutyTime7s'][-2:] + " ~ " + clinic['dutyTime7c'][:-2] + ':' + clinic['dutyTime7c'][-2:]

            clinic['time'] = OpenTime

            clinic['dis'] = haversine((user['lat'], user['lng']), (clinic['lat'], clinic['lng']))

            return render\_template("clinic.html", distance=clinic['dis'],name=clinic['name'], hpid=clinic['hpid'], number=clinic['numDoctor'], treatment=clinic['treatment'], lat=clinic['lat'], lng=clinic['lng'], location=clinic['address'], time=clinic['time'])

        elif page\_type == 'store':

            cur = conn.cursor(pymysql.cursors.DictCursor)

            sql = "SELECT \* FROM pharmacy p WHERE p.hpid=(SELECT pu.phid FROM pharmacy\_user pu WHERE pu.ulocal='{}' AND pu.udomain='{}') and p.name=(SELECT pu.phname FROM pharmacy\_user pu WHERE pu.ulocal='{}' and pu.udomain='{}')".format(user['local'], user['domain'], user['local'], user['domain'])

            cur.execute(sql)

            pharmacy = cur.fetchall()[0]

            return render\_template("store.html", name=pharmacy['name'], hpid=pharmacy['hpid'], lat=pharmacy['lat'], lng=pharmacy['lng'], location=pharmacy['address'])

    return render\_template("page\_type.html")

* + createAccount()는 회원가입을 하는 함수이다. 로컬, 도메인, 전화번호, 비밀번호, 위도, 경도, 타입을 입력 받고 type 테이블과 user 테이블에 그 정보를 저장해준다. 성공적으로 계정을 만들었을 경우에는 create\_success.html을 출력한다.

@app.route('/create\_account', methods=['POST', 'GET'])

def createAccount():

    global cnt\_user\_clinic, cnt\_user\_phar

    result = ""

    if request.method == "POST":

        cur = conn.cursor()

        sql = "INSERT INTO USER (name, phone, local, domain, passwd, lat, lng) VALUES ('{}','{}','{}','{}', '{}', {}, {})".format(request.form['name'], request.form['phone'], request.form['local'], request.form['domain'], request.form['passwd'], float(request.form['lat']), float(request.form['lng']))

        cur.execute(sql)

        conn.commit()

        type\_list = request.form.getlist('type')

        if 'clinic' in type\_list:

            sql = "INSERT into clinic\_user (ulocal, udomain, cid, cname) SELECT '{}', '{}', c.hpid, c.name FROM clinic c LIMIT {}, 1".format(request.form['local'], request.form['domain'], cnt\_user\_clinic)

            cur.execute(sql)

            conn.commit()

            cnt\_user\_clinic += 1

        if 'pharmacy' in type\_list:

            sql = "INSERT into pharmacy\_user (ulocal, udomain, phid, phname) SELECT '{}', '{}', ph.hpid, ph.name FROM pharmacy ph LIMIT {}, 1".format(request.form['local'], request.form['domain'], cnt\_user\_phar)

            cur.execute(sql)

            conn.commit()

            cnt\_user\_phar += 1

        if len(type\_list) == 0:

            return render\_template("create\_account.html", result = "Click your user type")

        elif len(type\_list) == 1:

            sql = "INSERT INTO TYPE (ulocal, udomain, {}) VALUES ('{}', '{}', TRUE)".format(type\_list[0] , request.form['local'], request.form['domain'])

        elif len(type\_list) == 2:

            sql = "INSERT INTO TYPE (ulocal, udomain, {}, {}) VALUES ('{}', '{}', TRUE, TRUE)".format(type\_list[0], type\_list[1], request.form['local'], request.form['domain'])

        elif len(type\_list) == 3:

            sql = "INSERT INTO TYPE (ulocal, udomain, {}, {}, {}) VALUES ('{}', '{}', TRUE, TRUE, TRUE)".format(type\_list[0], type\_list[1], type\_list[2], request.form['local'], request.form['domain'])

        elif len(type\_list) == 3:

            sql = "INSERT INTO TYPE (ulocal, udomain, {}, {}, {}, {}) VALUES ('{}', '{}', TRUE, TRUE, TRUE, TRUE)".format(type\_list[0], type\_list[1], type\_list[2],type\_list[3], request.form['local'], request.form['domain'])

        cur.execute(sql)

        conn.commit()

        return render\_template("create\_success.html")

    return render\_template("create\_account.html")

* + 기본적인 병원의 정보를 가져오고, clinic.html을 출력하는 함수이다. SELECT문을 사용하여 데이터를 가지고 오고, 병원의 오픈 시간을 OpenTime 변수에 하나의 문자열로 저장하여 같이 출력해준다.

@app.route("/clinic")

def clinic():

    global clinic, user

    if len(clinic) == 0:

        return redirect(url\_for(signIn))

    cur = conn.cursor(pymysql.cursors.DictCursor)

    sql = "SELECT \* FROM clinic c WHERE c.hpid=(SELECT cu.cid FROM clinic\_user cu WHERE cu.ulocal='{}' AND cu.udomain='{}') and c.name=(SELECT cu.cname FROM clinic\_user cu WHERE cu.ulocal='{}' and cu.udomain='{}')".format(user['local'], user['domain'], user['local'], user['domain'])

    cur.execute(sql)

    clinic = cur.fetchall()[0]

    OpenTime = ""

    if clinic['dutyTime1s'] != '0':

        OpenTime += " Mon: " + clinic['dutyTime1s'][:-2] + ':' + clinic['dutyTime1s'][-2:] + " ~ " + clinic['dutyTime1c'][:-2] + ':' + clinic['dutyTime1c'][-2:]

    if clinic['dutyTime2s'] != '0':

        OpenTime += "\n Tue: " + clinic['dutyTime2s'][:-2] + ':' + clinic['dutyTime2s'][-2:] + " ~ " + clinic['dutyTime2c'][:-2] + ':' + clinic['dutyTime2c'][-2:]

    if clinic['dutyTime3s'] != '0':

        OpenTime += "\n Wed: " + clinic['dutyTime3s'][:-2] + ':' + clinic['dutyTime3s'][-2:] + " ~ " + clinic['dutyTime3c'][:-2] + ':' + clinic['dutyTime3c'][-2:]

    if clinic['dutyTime4s'] != '0':

        OpenTime += "\n Thu: " + clinic['dutyTime4s'][:-2] + ':' + clinic['dutyTime4s'][-2:] + " ~ " + clinic['dutyTime4c'][:-2] + ':' + clinic['dutyTime4c'][-2:]

    if clinic['dutyTime5s'] != '0':

        OpenTime += "\n Fri: " + clinic['dutyTime5s'][:-2] + ':' + clinic['dutyTime5s'][-2:] + " ~ " + clinic['dutyTime5c'][:-2] + ':' + clinic['dutyTime5c'][-2:]

    if clinic['dutyTime6s'] != '0':

        OpenTime += "\n Sat: " + clinic['dutyTime6s'][:-2] + ':' + clinic['dutyTime6s'][-2:] + " ~ " + clinic['dutyTime6c'][:-2] + ':' + clinic['dutyTime6c'][-2:]

    if clinic['dutyTime7s'] != '0':

        OpenTime += "\n Sun: " + clinic['dutyTime7s'][:-2] + ':' + clinic['dutyTime7s'][-2:] + " ~ " + clinic['dutyTime7c'][:-2] + ':' + clinic['dutyTime7c'][-2:]

    clinic['time'] = OpenTime

    clinic['dis'] = haversine((user['lat'], user['lng']), (clinic['lat'], clinic['lng']))

    return render\_template("clinic.html", distance=clinic['dis'], name = clinic['name'], hpid = clinic['hpid'], number=clinic['numDoctor'], treatment=clinic['treatment'], lat=clinic['lat'], lng=clinic['lng'], location=clinic['address'], time=clinic['time'])

* + 병원의 예약 환자 리스트를 보여주는 함수이다. 병원과 환자예약정보가 담겨있는 clinic\_reserve 테이블에서 환자이름, 환자 전화번호, 예약날짜, 예약시간 데이터를 가지고 온 후 출력해준다.

@app.route("/clinic\_reserve\_list")

def clinic\_reserve\_list():

    global clinic

    cur = conn.cursor(pymysql.cursors.DictCursor)

    sql = "SELECT pname, pnumber, date, time from clinic\_reserve where cname='{}' and cid='{}'".format(clinic['name'], clinic['hpid'])

    cur.execute(sql)

    result = cur.fetchall()

    List = ""

    for res in result:

        List += "Name: " + res['pname'] + " Phone: " + res['pnumber'] + " Date: " + res['date'].strftime('%Y-%m-%d') + " Time: " + str(res['time'])

        List += '\n'

    return render\_template("clinic\_reserve\_list.html", name=clinic['name'], hpid=clinic['hpid'], list=List)

* + 환자를 처방하는 함수이다. 먼저 /treat\_patient에 들어갈 경우, 병원의 예약 리스트를 출력한다. 빈 칸의 환자들의 예약정보를 입력하여 처방할 수 있다.

@app.route("/treat\_patient", methods=['POST','GET'])

def clinic\_treat():

    global clinic

    cur = conn.cursor(pymysql.cursors.DictCursor)

    sql = "SELECT pname, pnumber, date, time from clinic\_reserve where cname='{}' and cid='{}'".format(clinic['name'], clinic['hpid'])

    cur.execute(sql)

    result = cur.fetchall()

    List = ""

    for res in result:

        List += "Name: " + res['pname'] + " Phone: " + res['pnumber'] + " Date: " + res['date'].strftime('%Y-%m-%d') + " Time: " + str(res['time'])

        List += '\n'

    return render\_template("treat\_patient.html", name=clinic['name'], hpid=clinic['hpid'], list=List)

* + Clinic\_treat 함수에서 처방전을 입력한 후, submit을 눌리면 clinic\_prescribe함수가 실행된다. Clinic\_prescribe에서는 prescription 테이블에 환자의 처방전을 저장한다. 병원이름은 자동으로 처방전에 저장된다. 처방이 완료된 데이터를 저장하는 clinic\_complete 테이블에 병원의 정보와 환자의 정보를 저장한다. 마지막으로 처방이 완료된 건은 clinic\_reserve 테이블에서 삭제한다.

@app.route("/clinic\_prescribe", methods=['POST','GET'])

def clinic\_prescribe():

    global clinic, user

    cur = conn.cursor(pymysql.cursors.DictCursor)

    if request.method == "POST":

        pname = request.form['Name']

        pnumber = request.form['Phone']

        date = datetime.datetime(int(request.form['year']), int(request.form['mon']), int(request.form['date']), int(request.form['hour']), int(request.form['min']))

        medicine = request.form['medicine']

        timeDosage = request.form['timeDosage']

        dayDosage = request.form['dayDosage']

        total = request.form['total']

        sql = "INSERT into prescription (date, clinic, pname, pnumber, medicine, timeDosage, dayDosage, totalDosage) VALUES ('{}', '{}', '{}', '{}', '{}', {}, {}, {})".format(date, clinic['name'], pname, pnumber, medicine, timeDosage, dayDosage, total)

        cur.execute(sql)

        conn.commit()

        sql = "SELECT date from clinic\_reserve where pname='{}' and pnumber='{}' and cname='{}' and cid='{}'".format(pname, pnumber, clinic['name'], clinic['hpid'])

        cur.execute(sql)

        tmpdate = cur.fetchall()[0] #check the day seperately

        sql = "INSERT into clinic\_complete (pname, pnumber, date, cname, cid, ulocal, udomain) VALUES ('{}', '{}', '{}', '{}', '{}', '{}', '{}')".format(pname, pnumber, tmpdate['date'], clinic['name'], clinic['hpid'], user['local'], user['domain'])

        cur.execute(sql)

        conn.commit()

        sql = "DELETE FROM clinic\_reserve where pname='{}' and pnumber='{}' and cname='{}' and cid='{}'".format(pname, pnumber, clinic['name'], clinic['hpid'])

        cur.execute(sql)

        conn.commit()

        return render\_template("clinic\_prescribe.html", total=total, dosage=timeDosage, daydosage=dayDosage, clinic=clinic['name'], medicine=medicine, date=date.strftime("%Y %m %d %H %m"), pname=pname, pnumber=pnumber, name=clinic['name'], hpid=clinic['hpid'])

* + 처방이 완료된 모든 환자의 리스트를 볼 수 있고, 검색할 수 있는 함수이다. 이름, 전화번호, 날짜를 입력하여 환자의 정보를 clinic\_complete 테이블에서 가지오고, 화면에 출력해준다.

@app.route("/clinic\_list\_all", methods=["POST", "GET"])

def clinic\_list\_all():

    global clinic, user

    cur = conn.cursor(pymysql.cursors.DictCursor)

    if request.method == "POST":

        pname = request.form['Name']

        pnumber = request.form['Phone']

        date = datetime.datetime(int(request.form['year']), int(request.form['mon']), int(request.form['date']))

        sql = "SELECT \* from clinic\_complete where ulocal='{}' and udomain='{}' and pname='{}' and pnumber='{}' and date='{}'".format(user['local'], user['domain'], pname, pnumber, date)

        cur.execute(sql)

        result = cur.fetchall()

        line = ""

        if len(result) != 0:

            result = result[0]

            line += "Name: " + result['pname'] + "\nPhone: " + result['pnumber'] + "\nDate: " + result['date'].strftime('%Y %m %d')

        line += '\n'

        return render\_template("clinic\_list\_all.html", result=line, name=clinic['name'], hpid=clinic['hpid'])

    sql = "SELECT pname, pnumber, date from clinic\_complete where ulocal='{}' and udomain='{}'".format(user['local'], user['domain'])

    cur.execute(sql)

    result = cur.fetchall()

    List = ""

    for res in result:

        List += "Name: " + res['pname'] + " Phone: " + res['pnumber'] + " Date: " + res['date'].strftime('%Y %m %d')

        List += '\n'

    return render\_template("clinic\_list\_all.html", name=clinic['name'], hpid=clinic['hpid'], list=List)

* + 가장 많이 방문한 병원 목록을 등록하는 함수이다. SELECT문을 이용하여 patient\_favorite 테이블에서 자주 방문하는 병원 목록병원 리스트를 출력하고, clinic에서 모든 병원의 리스트가 출력된다. 빈칸에 병원이름과 병원ID를 입력하게 되면 그 병원의 정보가 patient\_favorite 테이블에 저장된다.

@app.route("/register\_most\_visited", methods=['POST', 'GET'])

def register\_favorite():

    global user

    cur = conn.cursor(pymysql.cursors.DictCursor)

    sql = "SELECT name, hpid, address from clinic"

    cur.execute(sql)

    result = cur.fetchall()

    List = ""

    for res in result:

        List += "Name: " + res['name'] + " ID: " + res['hpid'] + " Location: " + res['address']

        List += '\n'

    sql = "SELECT cname, cid from patient\_favorite where pname='{}' and pnumber='{}'".format(user['name'], user['phone'])

    cur.execute(sql)

    result = cur.fetchall()

    favor = ""

    for res in result:

        favor += "Name: " + res['cname'] + " ID: " + res['cid']

        favor += '\n'

    if request.method == "POST":

        cname = request.form['cname']

        cid = request.form['cid']

        sql = "INSERT into patient\_favorite (pname, pnumber, cid, cname) VALUES ('{}','{}','{}','{}')".format(user['name'], user['phone'], cid, cname)

        cur.execute(sql)

        conn.commit()

        line = "Register {} {} as your favorite clinic".format(cname, cid)

        cur.close()

        return render\_template("register\_most\_visited.html", favor=favor, list=List, result=line, name=user['name'])

    cur.close()

    return render\_template("register\_most\_visited.html", favor=favor, list=List, name=user['name'])

* + 최근 방문한 병원을 출력하는 함수이다. clinic\_complete과 clinic\_user 테이블에서 데이터를 가지고 온다. 가장 최근 방문한 병원이 제일 상위에 출력한다.

def recent\_clinic():

    global user

    cur = conn.cursor(pymysql.cursors.DictCursor)

    sql = "SELECT c.cid, c.cname, p.date from clinic\_user c, clinic\_complete p WHERE c.ulocal=p.ulocal and p.udomain=c.udomain and p.pnumber='{}' order by p.date desc".format(user['phone'])

    cur.execute(sql)

    result = cur.fetchall()

    List = ""

    for res in result:

        List += "Date: " + res['date'].strftime("%Y-%m-%d") + " Name: " + res['cname'] + " ID: " + res['cid']

        List += '\n'

    cur.close()

    return render\_template("recent\_clinic.html", list=List, name=user['name'])

* + 거리로 병원을 찾는 함수이다. 입력 받은 거리(dis 변수) 안에 있는 병원들을 모두 출력한다. 만약에 더 많은 병원을 요청하는 경우(wdis 변수), OpenAPI\_Clinic 함수를 불러와 데이터베이스에 추가한다.

@app.route("/search\_clinic", methods=['POST', 'GET'])

def search\_clinic():

    global default\_clinic, user

    if request.method == "POST":

        dis = request.form['dis']

        wdis = request.form['wdis']

        if dis != "":

            dis = float(dis)

            cur = conn.cursor(pymysql.cursors.DictCursor)

            sql = "SELECT name, hpid, address, lng, lat from clinic;"

            cur.execute(sql)

            result = cur.fetchall()

            tmp = []

            mine= (user['lat'], user['lng'])

            for res in result:

                place = (res['lat'], res['lng'])

                if haversine(mine, place) <= dis:

                    res['distance'] = haversine(mine, place)

                    tmp.append(res)

            List = ""

            for res in tmp:

                List += "Name: " + res['name'] + " ID: " + res['hpid'] + " Location: " + res['address'] + " Distance: " + str(res['distance'])

                List += '\n'

            cur.close()

            return render\_template("search\_clinic.html", result=List, name=user['name'])

        elif wdis != "":

            wdis = float(wdis)

            OpenAPI\_Clinic(5, wdis)

            default\_clinic = wdis

            return render\_template("search\_clinic.html", result="More clinics are added into the database", name=user['name'])

    return render\_template("search\_clinic.html", name=user['name'], max=default\_clinic)

* + 진료 과목 리스트가 주어지면, 그중 하나를 입력하여, 그 과목을 진단하는 병원을 찾는다. Clinic 테이블의 treatment에 진료과목 가능한 과목이 문자열로 저장이 되어있다. Result 변수에 저장된 모든 병원들 중, treatment에 내가 원하는 진료과목이 포함되어 있을 경우, tmp 변수에 저장하고 그 병원들을 출력해준다.

@app.route("/search\_clinic\_type", methods=['POST', 'GET'])

def search\_clinic\_type():

    List = "내과 소아청소년과 신경과 정신건강의학과 피부과 외과 흉부외과 정형외과 신경외과 성형외과 산부인과 안과 이비인후과 비뇨기과 재활의학과\n마취통증의학과 영상의학과 치료방사선과 임상병리과 해부병리과 가정의학과 핵의학과 응급의학과 치과 구강악안면외과"

    cur = conn.cursor(pymysql.cursors.DictCursor)

    if request.method == "POST":

        depart = request.form['depart']

        sql = "SELECT name, hpid, address, treatment from clinic"

        cur.execute(sql)

        result = cur.fetchall()

        tmp = []

        for res in result:

            if depart in res['treatment']:

                tmp.append(res)

        line = ""

        for res in tmp:

            line += "Name: " + res['name'] + " ID: " + res['hpid'] + " Location: " + res['address'] + "\nMedical Department: " + res['treatment']

            line += '\n'

        return render\_template("search\_clinic\_type.html", list=List, result=line)

    return render\_template("search\_clinic\_type.html", list=List)

* + 이름을 통해 병원을 검색하는 함수이다. 먼저, SELECT문을 사용하여, clinic 테이블에 저장되어 있는 모든 병원 리스트를 출력해준다. 이후, 빈칸에 병원이름을 입력 받은 경우, SELECT문을 사용하여 clinic 테이블에 같은 이름의 병원을 찾아 출력해준다.

@app.route("/search\_clinic\_name", methods=['POST', 'GET'])

def search\_clinic\_name():

    cur = conn.cursor(pymysql.cursors.DictCursor)

    sql = "SELECT name, hpid, address from clinic"

    cur.execute(sql)

    result = cur.fetchall()

    tmp = ""

    for res in result:

        tmp += "Name: " + res['name'] + " ID: " + res['hpid'] + " Location: " + res['address']

        tmp += '\n'

    if request.method == "POST":

        cname = request.form['cname']

        sql = "SELECT name, hpid, address from clinic where name='{}'".format(cname)

        cur.execute(sql)

        result = cur.fetchall()

        List = ""

        for res in result:

            List += "Name: " + res['name'] + " ID: " + res['hpid'] + " Location: " + res['address']

            List += '\n'

        return render\_template("search\_clinic\_name.html", list=tmp, result=List, name=user['name'])

    return render\_template("search\_clinic\_name.html", list=tmp, name=user['name'])

* + 입력받은 거리 안에 있는 약국을 찾는 함수이다. “search\_clinic()” 함수와 같은 방식으로 구현하였다. 검색을 하고 싶은 경우, 거리를 입력받아 dis 변수에 저장을 해주고, 그 거리 안에 있는 모든 약국을 출력해주었다. 만약에 더 멀리 있는 약국을 찾고 싶은 경우, wdis에 거리를 입력받은 후, OpenAPI\_Pharmacy 함수를 이용하여 그 거리 안에 있는 약국을 데이터베이스에 추가해주었다. 모든 거리의 단위는 km이다.

@app.route("/search\_pharmacy", methods=['POST', 'GET'])

def search\_pharmacy():

    global default\_phar

    if request.method == "POST":

        dis = request.form['dis']

        wdis = request.form['wdis']

        if dis != "":

            dis = float(dis)

            cur = conn.cursor(pymysql.cursors.DictCursor)

            sql = "SELECT name, hpid, address, lng, lat from pharmacy;"

            cur.execute(sql)

            result = cur.fetchall()

            tmp = []

            mine= (user['lat'], user['lng'])

            for res in result:

                place = (res['lat'], res['lng'])

                if haversine(mine, place) <= dis:

                    res['distance'] = haversine(mine, place)

                    tmp.append(res)

            List = ""

            for res in tmp:

                List += "Name: " + res['name'] + " ID: " + res['hpid'] + " Location: " + res['address'] + " Distance: " + str(res['distance'])

                List += '\n'

            cur.close()

            return render\_template("search\_pharmacy.html", result=List, name=user['name'])

        elif wdis != "":

            wdis = float(wdis)

            OpenAPI\_Pharmacy(5, wdis)

            default\_phar = wdis

            return render\_template("search\_pharmacy.html", result="More pharmacies are added into the database", name=user['name'])

    return render\_template("search\_pharmacy.html", name=user['name'], max=default\_phar)

* + 병원을 예약하는 함수이다. POST 방식으로 환자이름, 환자의 전화번호, 병원이름, 병원번호, 예약날짜, 예약시간을 입력 받아 clinic\_reserve 테이블에 저장해준다. if문 밖의 SELECT문은 병원의 정보를 출력해주는 부분이다.

@app.route("/reserve\_clinic", methods=['POST', 'GET'])

def reserve\_clinic():

    global user

    cur = conn.cursor(pymysql.cursors.DictCursor)

    if request.method=='POST':

        pname = request.form['pname']

        pnumber = request.form['pnumber']

        cname = request.form['cname']

        cid = request.form['cid']

        date = datetime.date(int(request.form['year']), int(request.form['mon']), int(request.form['date']))

        time = datetime.time(int(request.form['hour']), int(request.form['min']), 0)

        sql = "INSERT into clinic\_reserve (pname, pnumber, cname, cid, date, time) values ('{}', '{}', '{}', '{}', '{}', '{}')".format(pname, pnumber, cname, cid, date, time)

        cur.execute(sql)

        conn.commit()

        return render\_template("reserve\_clinic.html", name=user['name'],result="Reservation is completed!")

    sql = "SELECT \* from clinic"

    cur.execute(sql)

    result = cur.fetchall()

    List = ""

    for res in result:

        OpenTime = ""

        if res['dutyTime1s'] != '0':

            OpenTime += " Mon: " + res['dutyTime1s'][:-2] + ':' + res['dutyTime1s'][-2:] + " ~ " + res['dutyTime1c'][:-2] + ':' + res['dutyTime1c'][-2:]

        if res['dutyTime2s'] != '0':

            OpenTime += "\n Tue: " + res['dutyTime2s'][:-2] + ':' + res['dutyTime2s'][-2:] + " ~ " + res['dutyTime2c'][:-2] + ':' + res['dutyTime2c'][-2:]

        if res['dutyTime3s'] != '0':

            OpenTime += "\n Wed: " + res['dutyTime3s'][:-2] + ':' + res['dutyTime3s'][-2:] + " ~ " + res['dutyTime3c'][:-2] + ':' + res['dutyTime3c'][-2:]

        if res['dutyTime4s'] != '0':

            OpenTime += "\n Thu: " + res['dutyTime4s'][:-2] + ':' + res['dutyTime4s'][-2:] + " ~ " + res['dutyTime4c'][:-2] + ':' + res['dutyTime4c'][-2:]

        if res['dutyTime5s'] != '0':

            OpenTime += "\n Fri: " + res['dutyTime5s'][:-2] + ':' + res['dutyTime5s'][-2:] + " ~ " + res['dutyTime5c'][:-2] + ':' + res['dutyTime5c'][-2:]

        if res['dutyTime6s'] != '0':

            OpenTime += "\n Sat: " + res['dutyTime6s'][:-2] + ':' + res['dutyTime6s'][-2:] + " ~ " + res['dutyTime6c'][:-2] + ':' + res['dutyTime6c'][-2:]

        if res['dutyTime7s'] != '0':

            OpenTime += "\n Sun: " + res['dutyTime7s'][:-2] + ':' + res['dutyTime7s'][-2:] + " ~ " + res['dutyTime7c'][:-2] + ':' + res['dutyTime7c'][-2:]

        List += "Name: " + res['name'] + " ID: " + res['hpid'] + " Location: " + res['address'] + " Open Time: \n" + OpenTime

        List += '\n'

    return render\_template("reserve\_clinic.html", name=user['name'], list=List)

* + 약국을 예약하는 함수이다. POST 방식으로 환자이름, 환자 전화번호, 약국이름, 약국ID, 병원이름을 입력받은 후 pharmacy\_reserve 테이블에 저장을 해준다. 병원이름은 처방전이 여러 개일 경우를 대비해서 입력받는다. If문 밖의 SELECT문은 pharmacy 테이블에서 모든 약국 리스트를 출력하고, 병원에서 받은 처방전의 리스트를 출력하는 코드이다.

@app.route("/reserve\_pharmacy", methods=['POST', 'GET'])

def reserve\_pharmacy():

    global user

    cur = conn.cursor(pymysql.cursors.DictCursor)

    if request.method=='POST':

        pname = request.form['pname']

        pnumber = request.form['pnumber']

        phname = request.form['phname']

        phid = request.form['phid']

        cname = request.form['cname']

        sql = "INSERT into pharmacy\_reserve (pname, pnumber, phname, phid, cname) values ('{}', '{}', '{}', '{}', '{}')".format(pname, pnumber, phname, phid, cname)

        cur.execute(sql)

        conn.commit()

        sql = "Update Prescription set pharmacy='{}' where pname='{}' and pnumber='{}' and clinic='{}'".format(phname,pname,pnumber,cname)

        cur.execute(sql)

        conn.commit()

        return render\_template("reserve\_pharmacy.html", result="Reservation is completed!")

    sql = "SELECT \* from pharmacy"

    cur.execute(sql)

    result = cur.fetchall()

    List = ""

    for res in result:

        List += "Name: " + res['name'] + " ID: " + res['hpid'] + " Location: " + res['address']

        List += '\n'

    sql = "SELECT \* from prescription where pnumber='{}' and pharmacy is null".format(user['phone'])

    cur.execute(sql)

    result = cur.fetchall()

    List2 = ""

    for res in result:

        List2 += "Name: " + res['clinic'] + " ID: " + res['date'].strftime("%Y-%m-%d")

        List2 += '\n'

    return render\_template("reserve\_pharmacy.html", pre=List2, list=List, name=user['name'])

* + 처방전의 결과를 볼 수 있는 함수이다. POST 방식으로 병원, 약국, 병원에서 처방 받은 시간을 입력하여 처방전을 검색할 수 있다. 이 때, 처방전의 모든 상세항목을 확인할 수 있다. If문 밖의 코드는 나의 모든 처방전 리스트의 날짜, 결과, 약국이름, 병원이름을 SELECT문을 사용하여 가지고 온다. 그 이후, 리스트를 출력해준다. 약국의 처방결과가 1일 경우에는 Checking(확인 중), 결과가 2일 경우에는 Available, 결과가 3일경우에는 Unavailable로 바꾸어 출력해준다.

@app.route("/prescription\_result", methods=['POST', 'GET'])

def prescription\_result():

    cur = conn.cursor(pymysql.cursors.DictCursor)

    if request.method=='POST':

        clinic = request.form['clinic']

        pharmacy = request.form['pharmacy']

        date = datetime.datetime(int(request.form['year']), int(request.form['mon']), int(request.form['date']), int(request.form['hour']), int(request.form['min']))

        sql = "select \* from prescription where clinic='{}' and date='{}'".format(clinic, date)

        cur.execute(sql)

        result = cur.fetchall()

        for res in result:

            pname = res['pname']

            pnumber = res['pnumber']

            date = res['date']

            clinic = res['clinic']

            medicine = res['medicine']

            dosage = res['timeDosage']

            daydosage = res['dayDosage']

            total = res['totalDosage']

            pharmacy = res['pharmacy']

            pdate = res['prescriptedDay']

            extra = res['extra']

            presult = res['result']

            if presult == 1:

                presult = "Checking"

            elif presult == 2:

                presult = "Available"

            elif presult == 3:

                presult = "Unavailable"

            return render\_template("prescription\_result.html", extra=extra, pname=pname, pnumber=pnumber, date=date, clinic=clinic, medicine=medicine, dosage=dosage, daydosage=daydosage, total=total, pharmacy=pharmacy, pdate=pdate, result=presult)

    sql = "SELECT p.date, p.result, p.pharmacy, p.clinic from prescription p where p.pname='{}' and p.pnumber='{}' order by p.date desc".format(user['name'], user['phone'])

    cur.execute(sql)

    result = cur.fetchall()

    List = ""

    for res in result:

        tmp = ""

        if res['result'] == 1:

            tmp = "Checking"

        elif res['result'] == 2:

            tmp = "Available"

        elif res['result'] == 3:

            tmp = "Unavailabe"

        else:

            tmp = "Not Reserve Pharmacy Yet"

        if res['pharmacy'] == None:

            res['pharmacy'] = ""

        List += "Result: " + tmp + " Date: " + res['date'].strftime("%Y-%m-%d, %H:%M") + " Clinic Name: " + res['clinic'] + " Pharmacy: " + res['pharmacy']

        List += '\n'

    return render\_template("prescription\_result.html", list=List)

* + “pharmacy\_reserve” 테이블에서 데이터를 가지고 온 후, 약국을 예약한 환자들의 처방전을 볼 수 있는 함수이다.

@app.route("/pharmacy\_reserve\_list", methods=['GET'])

def pharmacy\_reserve\_list():

    global user, pharmacy

    cur = conn.cursor(pymysql.cursors.DictCursor)

    sql = "SELECT \* from prescription where pharmacy='{}' and result=1".format(pharmacy['name'])

    cur.execute(sql)

    result = cur.fetchall()

    List = ""

    for res in result:

        List += "Name: " + res['pname'] + " Phone: " + res['pnumber'] + " Clinic: "+res['clinic'] + " Date: "+res['date'].strftime("%Y-%m-%d-%H-%M") + " Medicine: " + res['medicine'] + " Dosage Per Time: " + str(res['timeDosage']) + " Dosage Per Day: " +str(res['dayDosage']) + " Total Dosage: " + str(res['totalDosage'])

        List += '\n'

    return render\_template("pharmacy\_reserve\_list.html", list=List)

* + 약국에서 환자를 처방하는 함수이다. POST방식으로 환자의 이름, 전화번호, 병원이름, 조제연월일, 조제내용(웹에서 extra로 표시), 처방결과를 입력을 받는다. 이 내용들을 prescription 테이블에 업데이트를 한다. 그 이후, pharmacy\_reserve 테이블에서 이 환자의 예약 내역을 지운다. 약국 이름은 자동으로 처방전에 업데이트 된다. If문 밖에 있는 코드는 현재 약국으로 예약을 한 사람들의 리스트를 골라 출력하는 부분이다.

@app.route("/prescribe\_patient", methods=['GET', 'POST'])

def prescribe\_patient():

    global pharmacy, user

    result = ""

    cur = conn.cursor(pymysql.cursors.DictCursor)

    if request.method == "POST":

        pname = request.form['pname']

        pnumber = request.form['pnumber']

        cname = request.form['cname']

        if request.form['year'] != "":

            date = datetime.datetime(int(request.form['year']), int(request.form['mon']), int(request.form['date']), int(request.form['hour']), int(request.form['min']))

        result = int(request.form['type'])

        extra = request.form['extra']

        if result == 2:

            sql = "UPDATE prescription set pharmacy='{}', prescriptedDay='{}',result={}, extra='{}' WHERE pname='{}' and pnumber='{}' and result=1 and clinic='{}'".format(pharmacy['name'],date,2,extra, pname, pnumber, cname)

            cur.execute(sql)

            conn.commit()

            result = "Prescription is completed"

        elif result == 3:

            sql = "UPDATE prescription set result=3, pharmacy='{}' where pname='{}' and pnumber='{}' and clinic='{}'".format(pharmacy['name'], pname,pnumber, cname)

            cur.execute(sql)

            conn.commit()

            result = "Prescription is completed"

        sql = "DELETE FROM pharmacy\_reserve where pname='{}' and pnumber='{}' and phname='{}' and phid='{}' and cname='{}'".format(pname, pnumber, pharmacy['name'], pharmacy['hpid'], cname)

        cur.execute(sql)

        conn.commit()

        return render\_template("prescribe\_patient.html", result=result)

    sql = "SELECT pname, pnumber,cname from pharmacy\_reserve pr where phname='{}' and phid='{}'".format(pharmacy['name'], pharmacy['hpid'])

    cur.execute(sql)

    result = cur.fetchall()

    List = ""

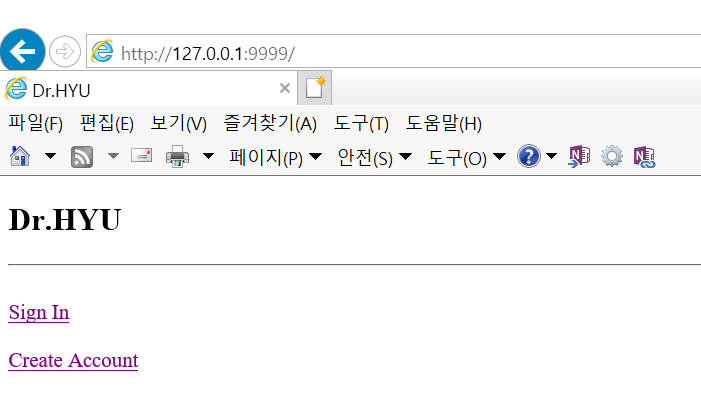
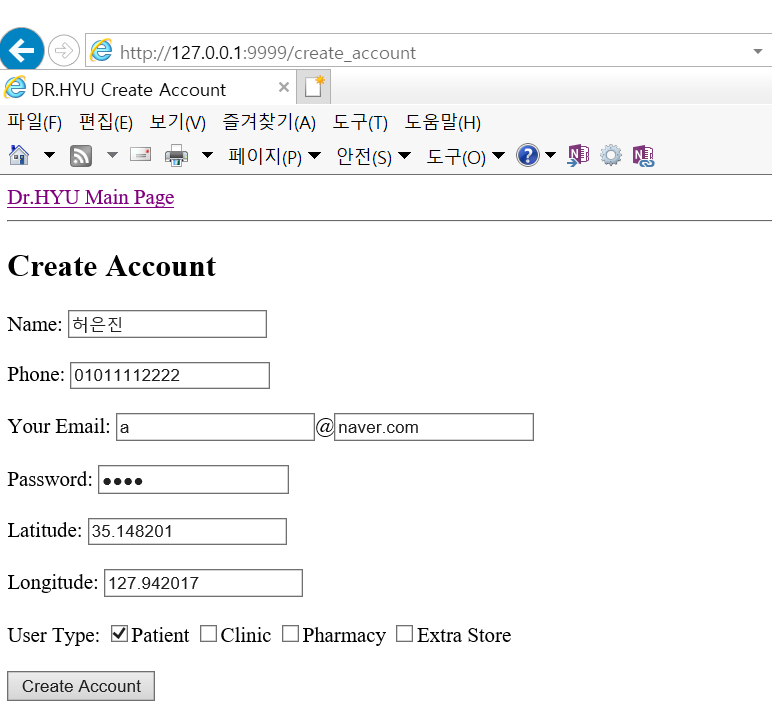
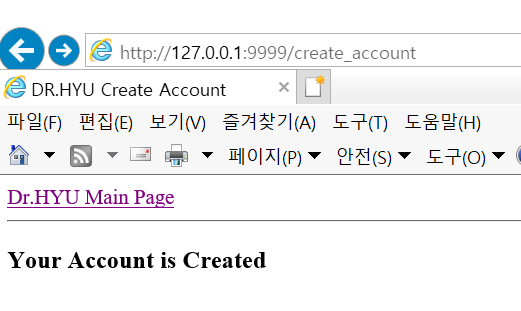
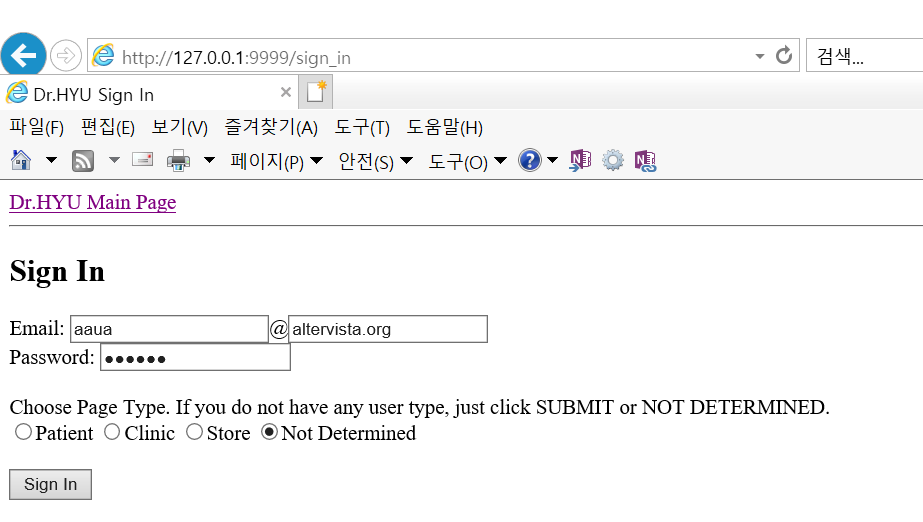
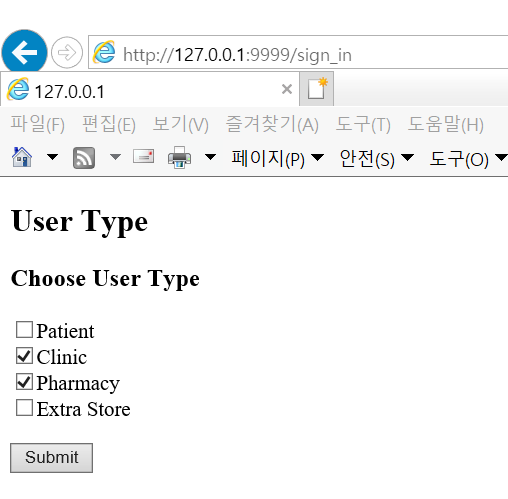
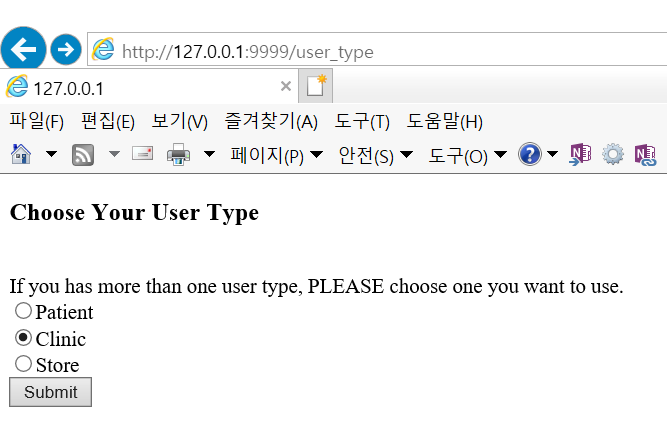
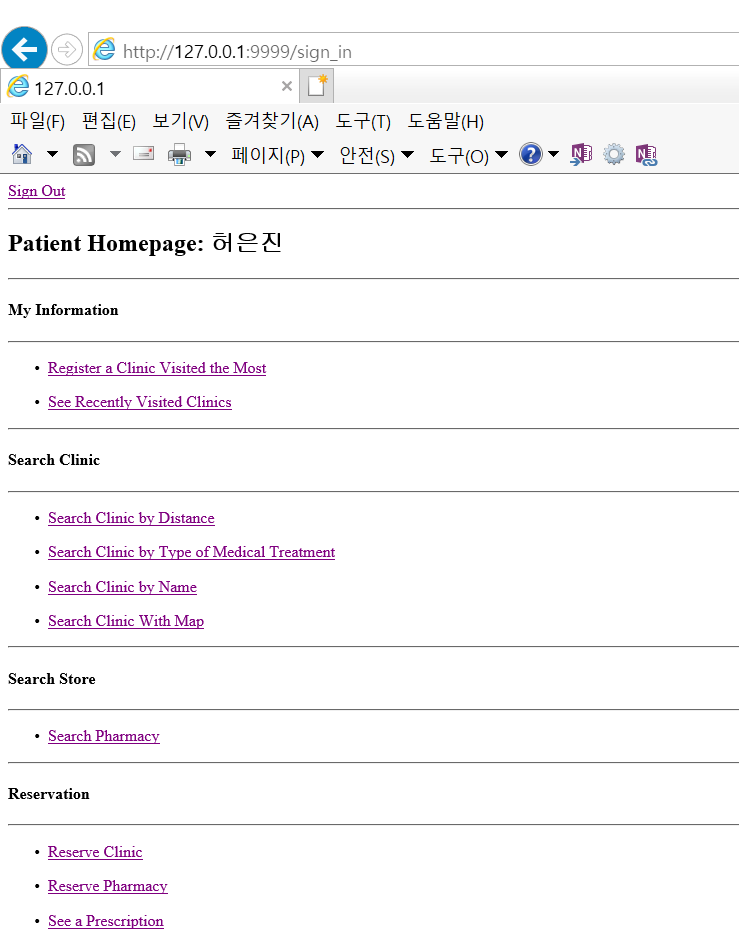
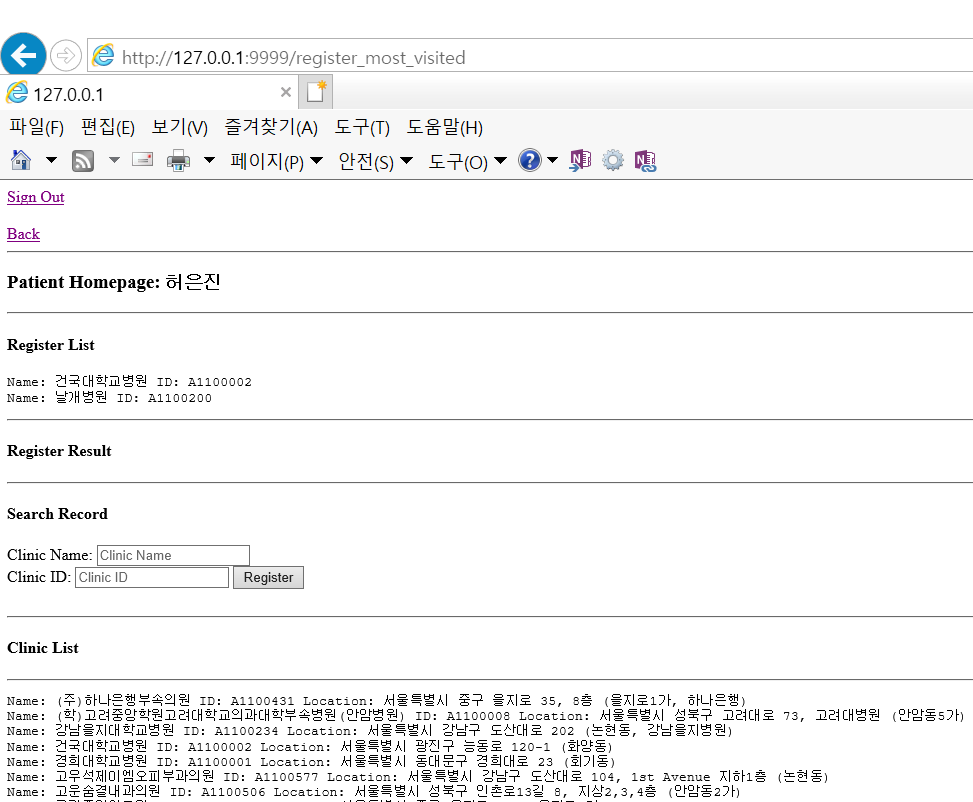
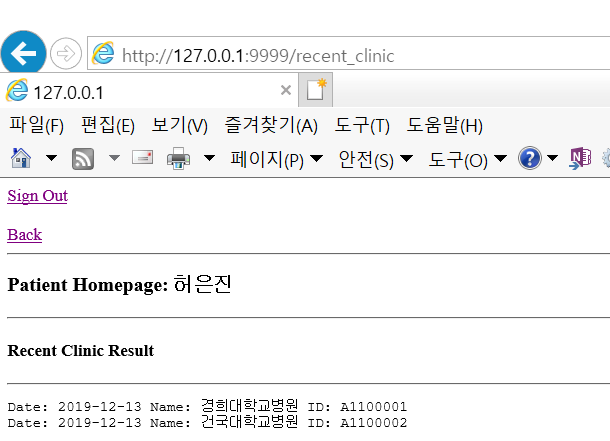
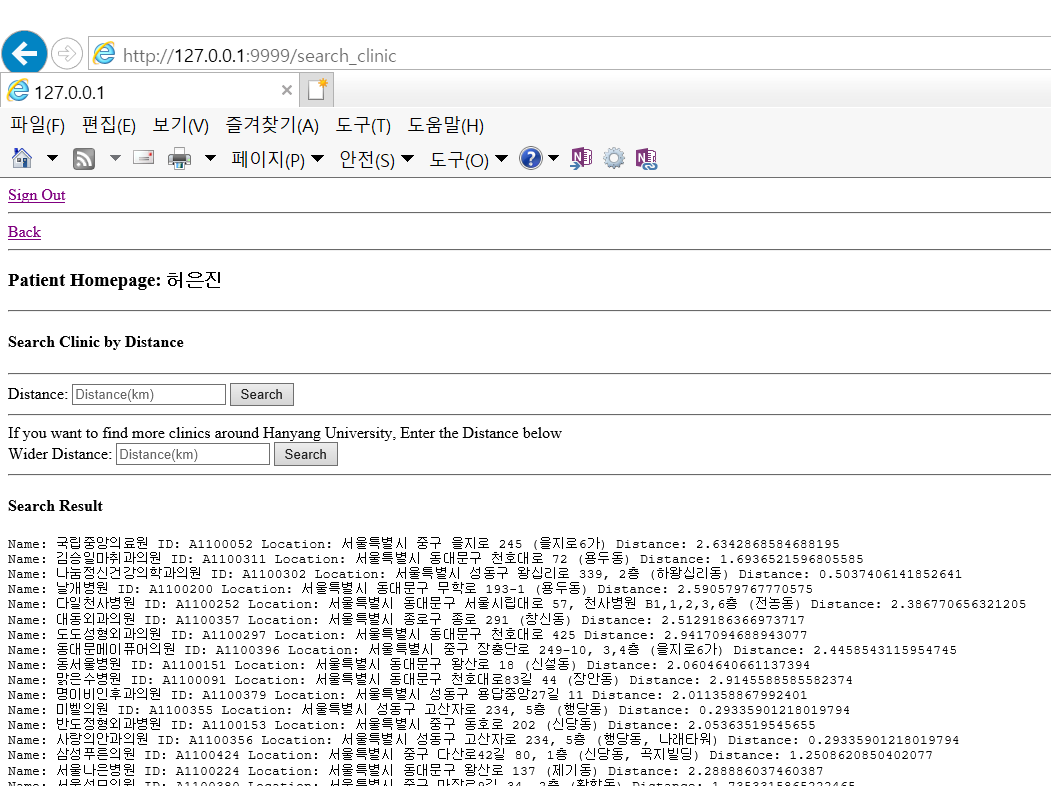
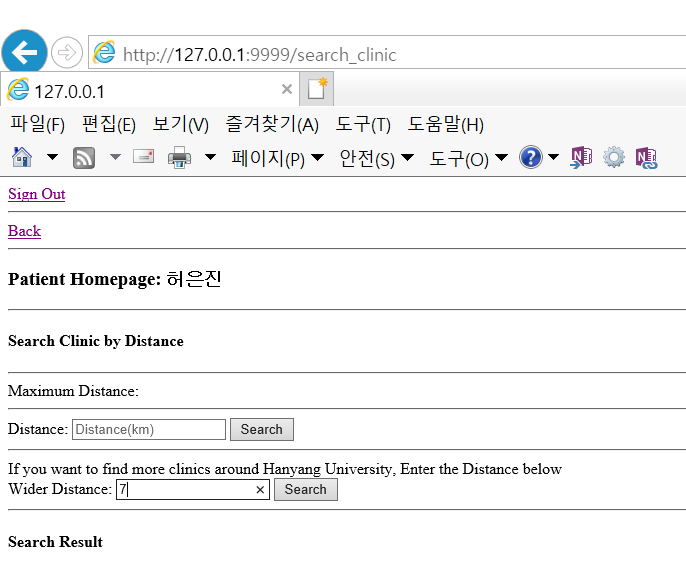
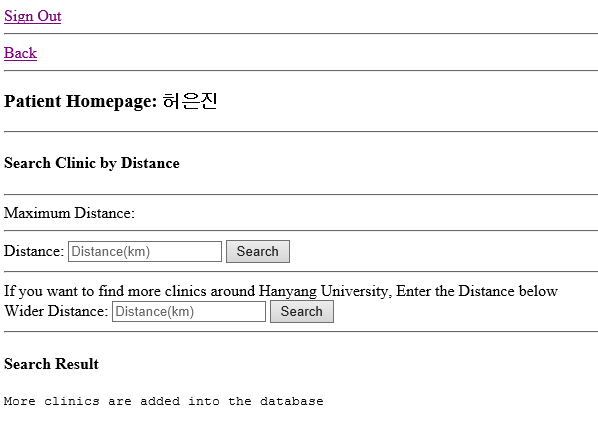
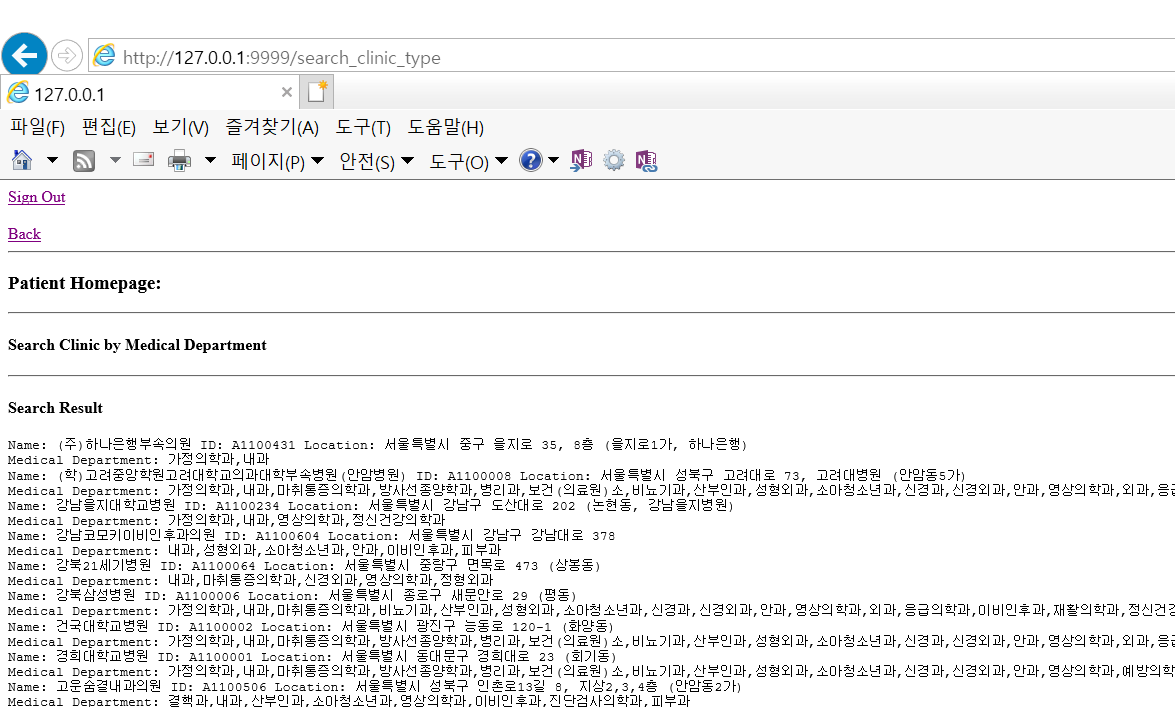
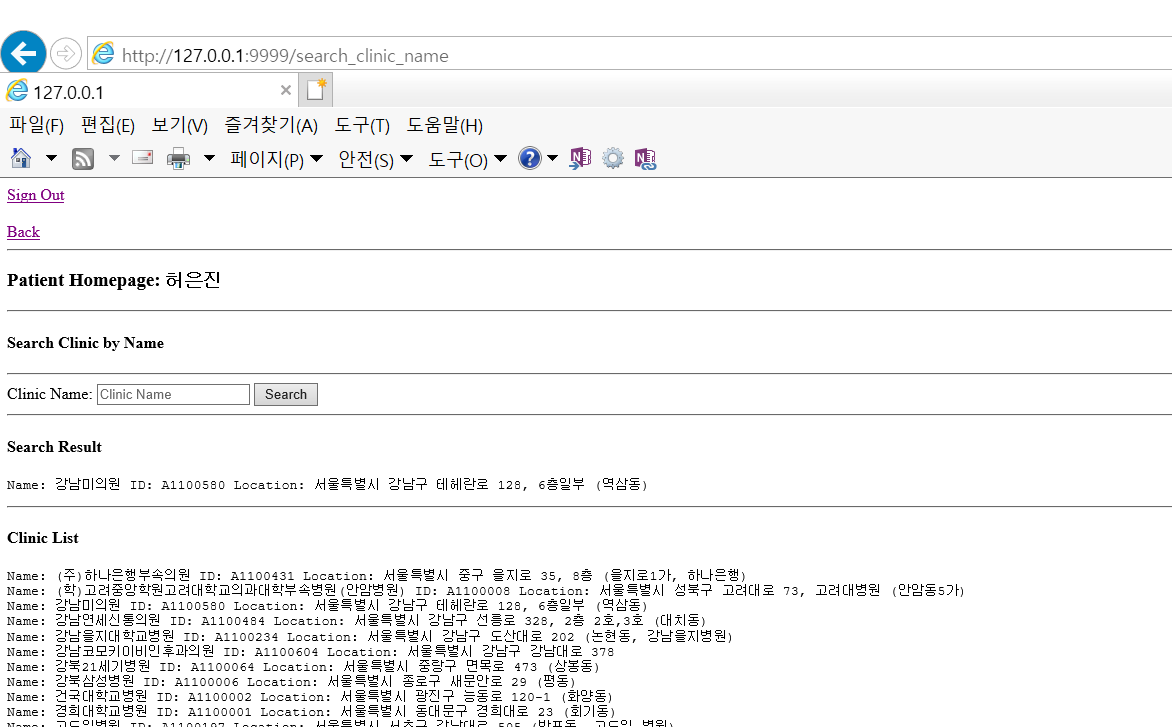
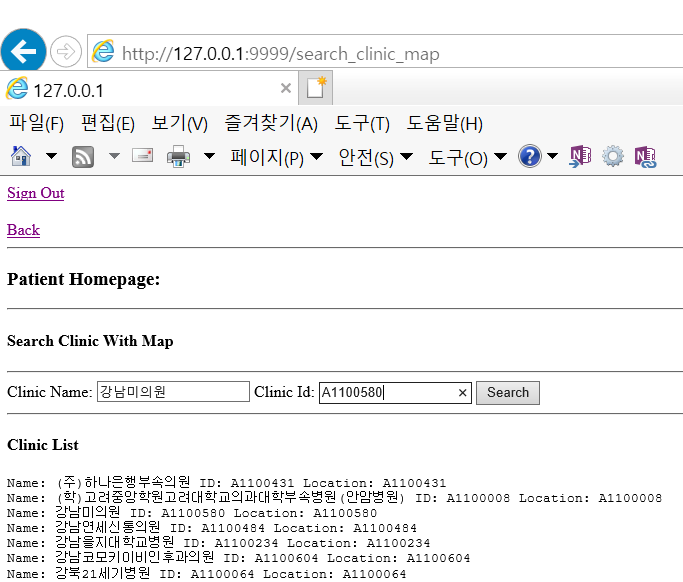
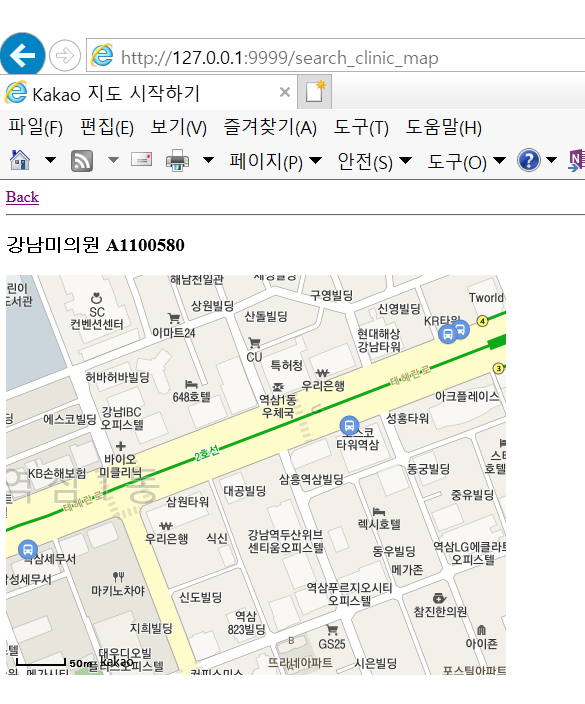
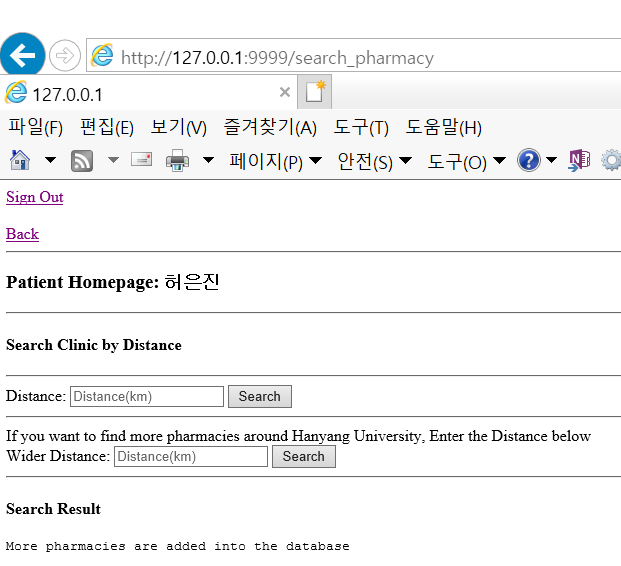
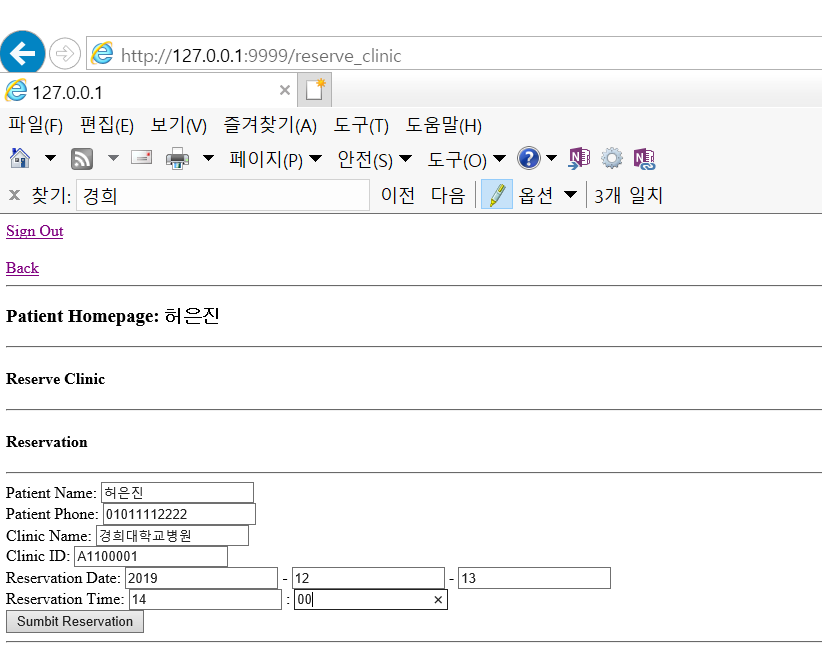
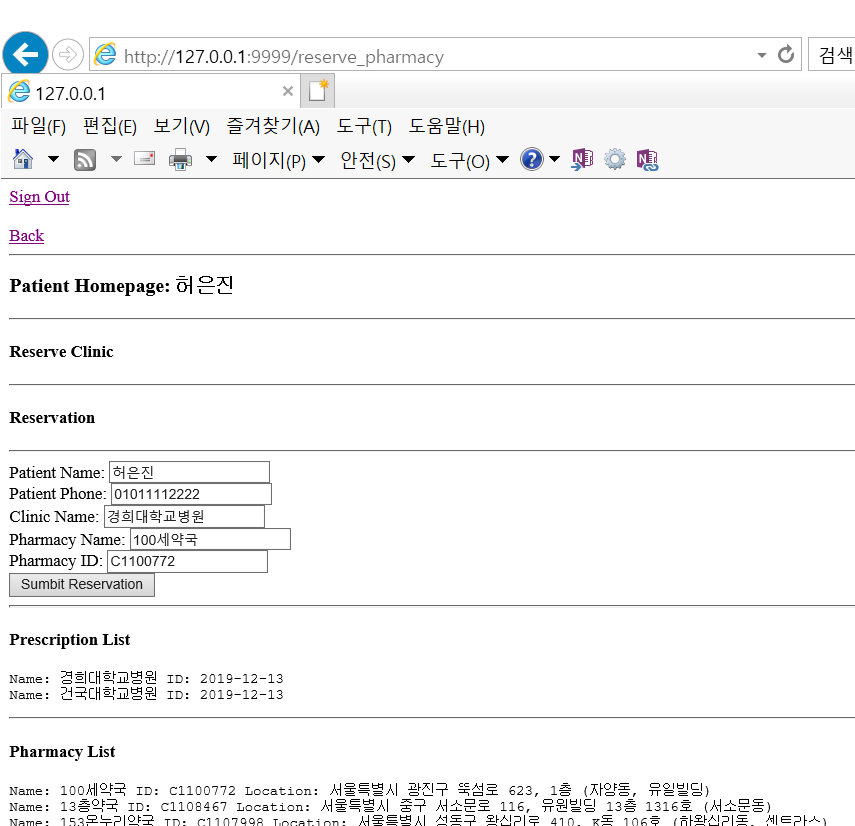
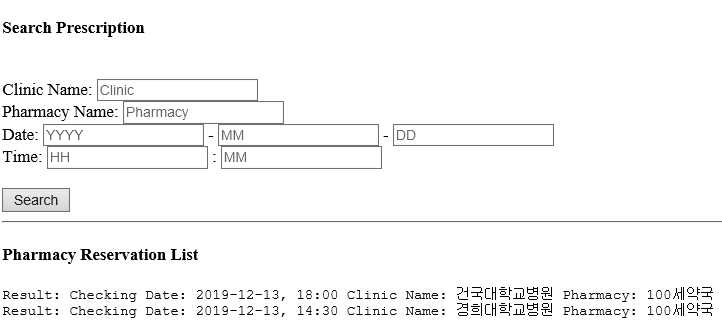
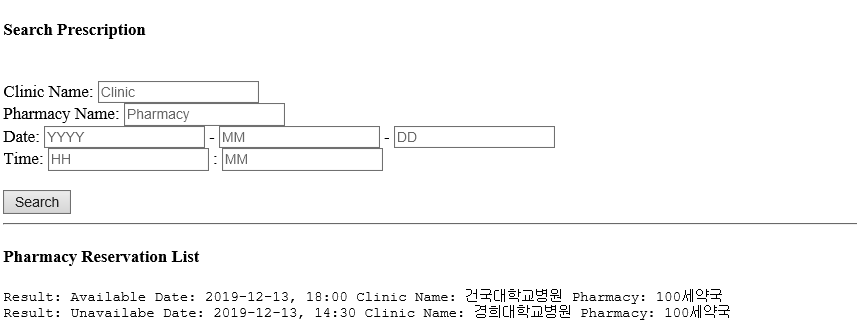
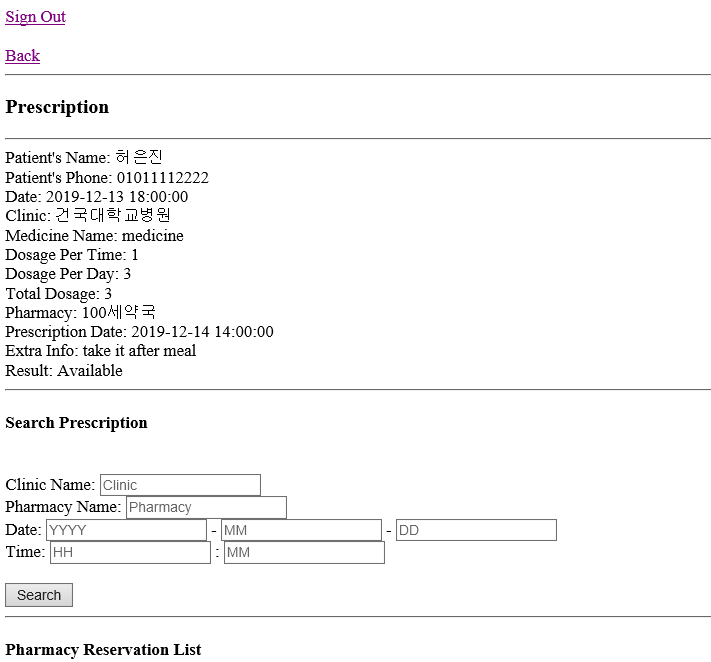
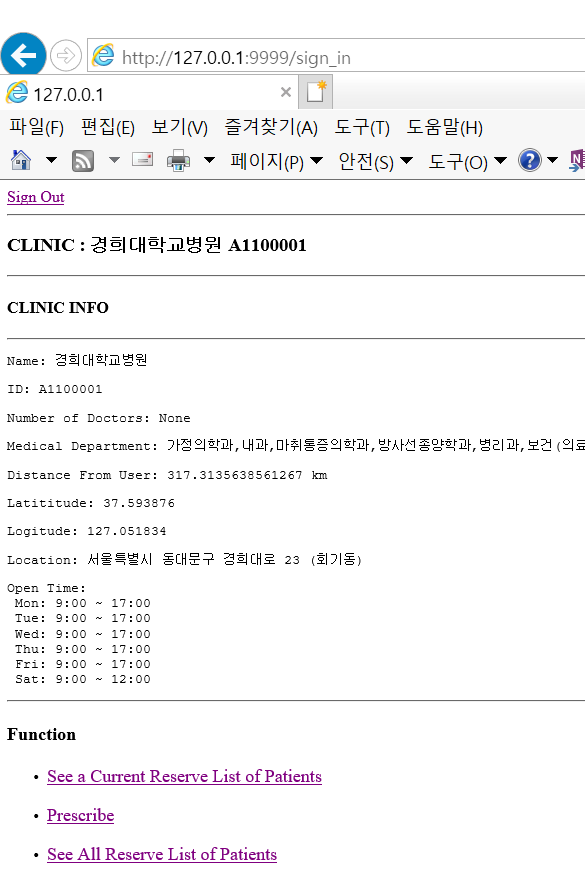
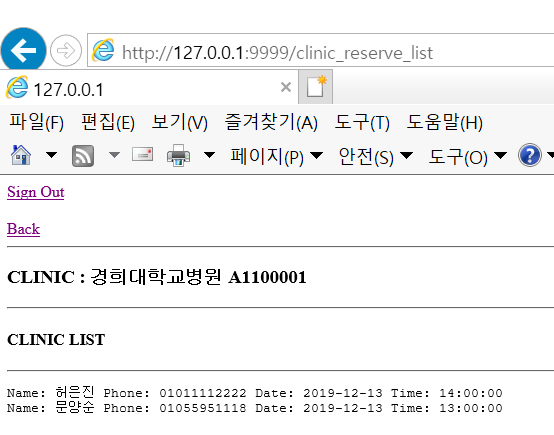
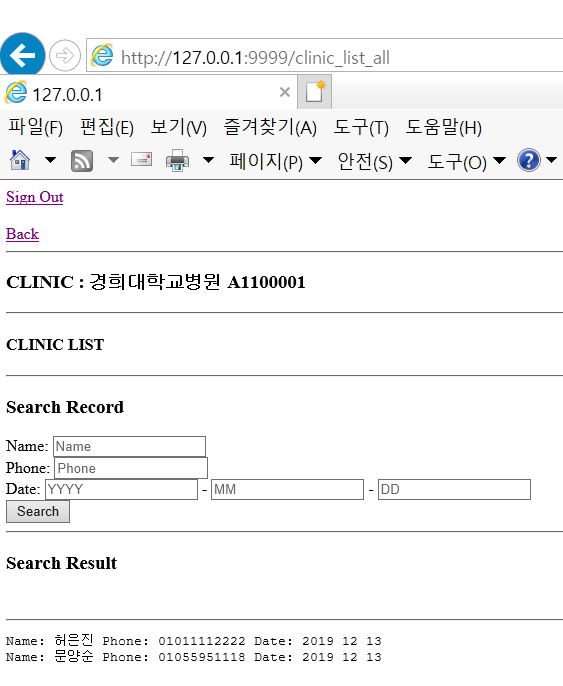
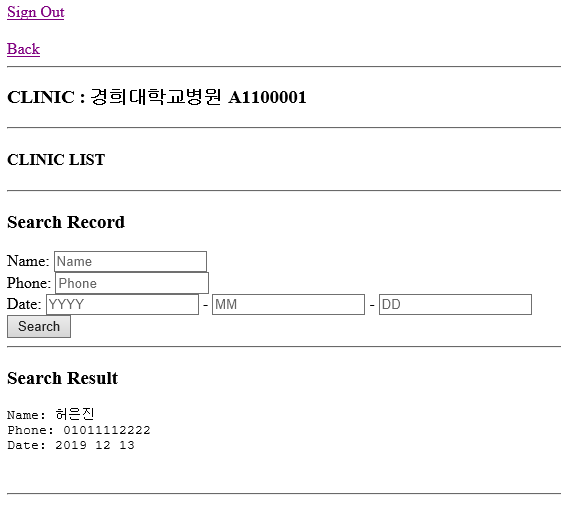
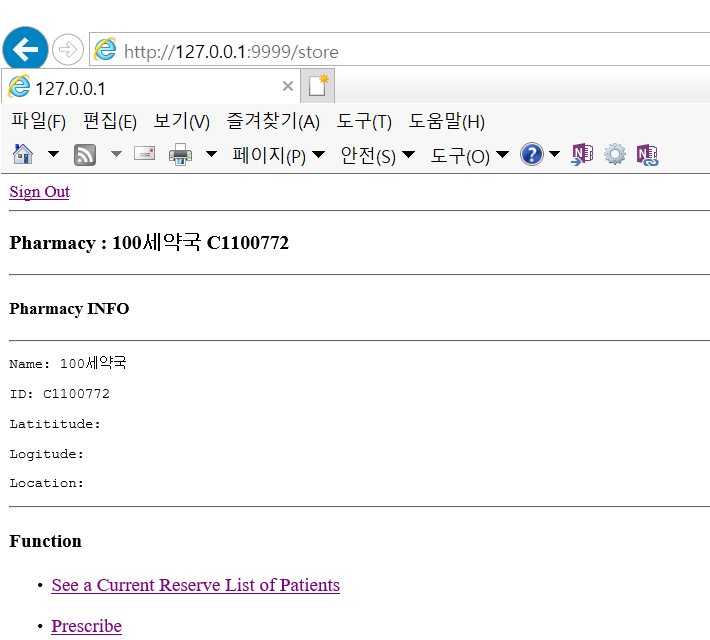
    for res in result:

        List += "Name: " + res['pname'] + " Phone: " + res['pnumber'] + " Clinic: " + res['cname']

        List += '\n'

    return render\_template("prescribe\_patient.html", list=List)

실행화면

* + 처음 실행 화면
  + 
  + 회원가입 화면
  + 
  + 회원가입 완료
  + 
  + 로그인 – 타입이 존재하는 경우
  + 
  + 로그인 – 타입이 결정되지 않은 경우( not determined 클릭 후 submit 또는 그냥 submit 클릭)
  + 
  + 타입 결정 화면
  + 
  + 현재 사용할 페이지 결정
  + 
  + 환자화면
  + 
  + 환자 – 자주 가는 병원 등록
  + 
  + 환자 – 최근에 방문한 병원 확인
  + 
  + 환자 – 거리로 병원 찾기 (3km를 입력한 후 결과화면)
  + 
  + 환자 – 거리로 병원 찾기 – 더 많은 데이터 추가
  + 
  + 
  + 환자 – 진단과목으로 병원 찾기 (내과를 입력한 경우 결과)
  + 
  + 환자 – 이름으로 병원 찾기 (강남미의원 검색 결과)
  + 
  + 환자 – 병원 검색을 통한 지도 보기
  + 
  + 
  + 환자 – 거리로 약국찾기 (3km를 입력한 경우)
  + 
  + 환자 – 거리로 약국 찾기 – 더 많은 데이터 추가한 후 결과 화면
  + 
  + 환자 – 병원 예약
  + 
  + 환자 – 약국 예약
  + 
  + 환자 – 처방전 확인 (약국 처방전 화면)
  + 
  + 환자 – 약국 처방 후 처방전 확인화면
  + 
  + 환자 – 처방전 결과 확인 화면
  + 
  + 병원 화면
  + 
  + 병원 – 현재 예약자보기
  + 
  + 병원 – 처방하기
  + 
  + 병원 – 환자의 처방 기록 확인
  + 
  + 병원 – 환자의 처방기록 확인 (허은진 검색결과)
  + 
  + 약국 화면
  + 
  + 약국 – 환자 예약 리스트와 처방전 확인
  + 
  + 약국 – 처방하기(가능)
  + 
  + 약국 – 처방하기(불가능)
  + 