history_server.py

```
def socket_create():
    try:
        global host
        global port
        global s
        host = ''
        port = 8000
        s = socket.socket()
    except socket.error as err:
        print('socket create err : ' + str(err))

def socket_bind():
    try:
        global host
        global port
        global s
        print('bind socket to port : ' + str(port))
        s.bind((host,port))
        s.listen(5)
    except socket.error as err:
        print('socket create err : ' + str(err))
        # print('retry...')
        # socket_bind()

def socket_accept():
    conn, addr = s.accept()
    print('connection success')
    print('ip : ' + addr[0] + ' / port : ' + str(addr[1]))
    csv_transfer(conn)

    conn.close()
```

socket create : 서버쪽 소켓을 생성합니다, 포트번호 9999(수정하세요 사진)
socket bind : 주어진 포트번호와 호스트 주소를 바탕으로 해당 소켓에 포트번호를 바인딩 줍니다
socket accpet : 클라이언트가 접속하게 되면 connection을 만들고 주어진 작업 수행, 여기선 csv 파일을 클라이언트가 서버로 전송해주는 csv_transfer 함수 호출
conn.close : 다 완료되면 연결 종료

```python
def csv_transfer(conn):
    data = conn.recv(1024)
    if not data:
        print("no data")
        sys.exit()
    with open('web_history.csv', 'wb') as f:
        try:
            while data:
                f.write(data)
                data = conn.recv(1024)
        except Exception as e:
            print(e)
    print('csv file ready')

def history_process():

    good_things=[]
    bad_things=[]



    with open('web_history.csv','r', encoding='utf-8') as file:
        datas = csv.reader(file)

    # '클라우드' or '도커' or 'Splunk' or 'sql' or 'SQL' or 'Python' or '파이썬'
        for row in datas:
            if '클라우드' in row[0] or '도커' in row[0] or 'Splunk' in row[0] or 'sql' in row[0] or\
                'SQL' in row[0] or 'Python'in row[0] or '파이썬' in row[0] or 'splunk' in row[0] or 'vscode' in row[0]:
                good_things.append((row[0],row[1],))
            else:
                bad_things.append((row[0],row[1],))

    cr.db_update(good_things,bad_things)

    print('db update complete')
```

csv_transfer : 서버쪽에선 클라이언트가 연결된 후 클라이언트가 보내는 csv 데이터를 recv 함수를 통해 받아서 web_history.csv 파일 생성

history_process : 이건 정민씨가 한거니깐, 제가 설명해야할 부분은 맨 밑에 cr.db_update 함수, import 에 보시면 crawling_csv.py를 import 시킨 후 거기에 있는 db_update 함수를 불러오는것, 히스토리 프로세스 함수에서 처리한 굿띵즈와 배드띵즈 리스트를 crawling_csv.py의 db_update 함수에다 넘겨주고 db에 연동시키는 역할

history_client.py

```python
def get_chrome_history():
    history_path = 'C:\\Users\\' +os.getlogin()+'\\AppData\\Local\\Google\\Chrome\\User Data\\Default\\History'
    con = sqlite3.connect(history_path)
    cursor = con.cursor()

    q = """
        SELECT title,datetime(last_visit_time/1000000-11644473600,'unixepoch','localtime') \
                    FROM urls where last_visit_time >= ? order by last_visit_time asc
        """
    # cur_time = (datetime.datetime(2021,
    #                               datetime.datetime.today().month,
    #                               datetime.datetime.today().day,
    #                               9,
    #                               0
    #                               ).timestamp()+11644473600)*1000000
    cur_time = (datetime.datetime(2021,
                                  4,
                                  5,
                                  9,
                                  0
                                  ).timestamp()+11644473600)*1000000

    cursor.execute(q,(cur_time,))
    urls = cursor.fetchall()
    try:
        with open('web_history.csv', 'w', newline='', encoding='utf8') as hist:
            history_writer = csv.writer(hist)
            for line in urls:
                history_writer.writerow(line)
        print('chrome history parsing complete')

    except Exception as e:
        print('chrome history parsing error')
        print(e)
        sys.exit()
```

겟 크롬 히스토리 : 크롬 브라우저의 방문기록은 위에 주어진 history_path 경로에 sqlite3 DB 파일로 저장되어 있음, os.getlogin()은 현재 로컬피씨 유저의 id를 가져오는 함수 (윈도우 로그인 할때 이름 보이는거, 그거 따오는거. user폴더라고 함) path 경로를 통해 sqlite3 파이썬 내장 패키지를 이용하여 히스토리 DB를 불러오도록 함

q : 히스토리 정보 긁어오는 sqlite3 select 쿼리문.

select title, datetime(last_visit_time 어쩌고 저쩌고 괄호닫고 까지 : 여기 잘 설명해야하는데 크롬 히스토리는 webkit(웹킷) 시간으로 설정되어있음 (1601년 1월 1일 0시 0분 0초 기준 누적 시간을 마이크로세컨드로 표현) 이를 EPOCH 시간(에포크 시간 or 유닉스 시간) 으로 변환해주는게 저 숫자로 된 공식 (1970년 1월 1일 0시 0분 0초 기준 누적 시간을 초(seconds)로 표현), 에포크 시간으로 변환해야 datetime 함수로 우리가 쉽게 볼 수 있는 human date로 변환이 가능함,
  즉 last_visti_time은 webkit 시간이니깐 이를 우선 뒤에 수식을 통해 EPOCH 시간으로 변환 후 datetime 함수에 넣어서 human date로 표현하도록 만들었다
그걸 select 쿼리문으로 히스토리의 방문 사이트 이름(title)과 방문날짜(datetime 이하)를 수집함

cur_time : 이건 현재 시간을 webkit 시간으로 변경하는 용도. 크롬 히스토리에서 쿼리문 전

송할때 현재 시간(현재 월, 일 오전 9시 기준) 이후로 쌓인 history를 긁어오기 위해서. q에 보시면 where last_visit_time >= ?로 된 부분이 있는데 거기에 cur_time 값이 들어가서 그 시간 보다 큰(현재 날짜 오전 9시 이후) last_visit_time 값을 가진 데이터를 뽑아오게 한다 이렇게 설명하심 됩니다. 왜 또 webkit 시간으로 바꿔야 하나? -> 크롬에서 webkit을 쓰니깐.. 어쨌든 history 긁어오는 쿼리문 돌릴려면 webkit 시간 기준으로 해야하니깐...

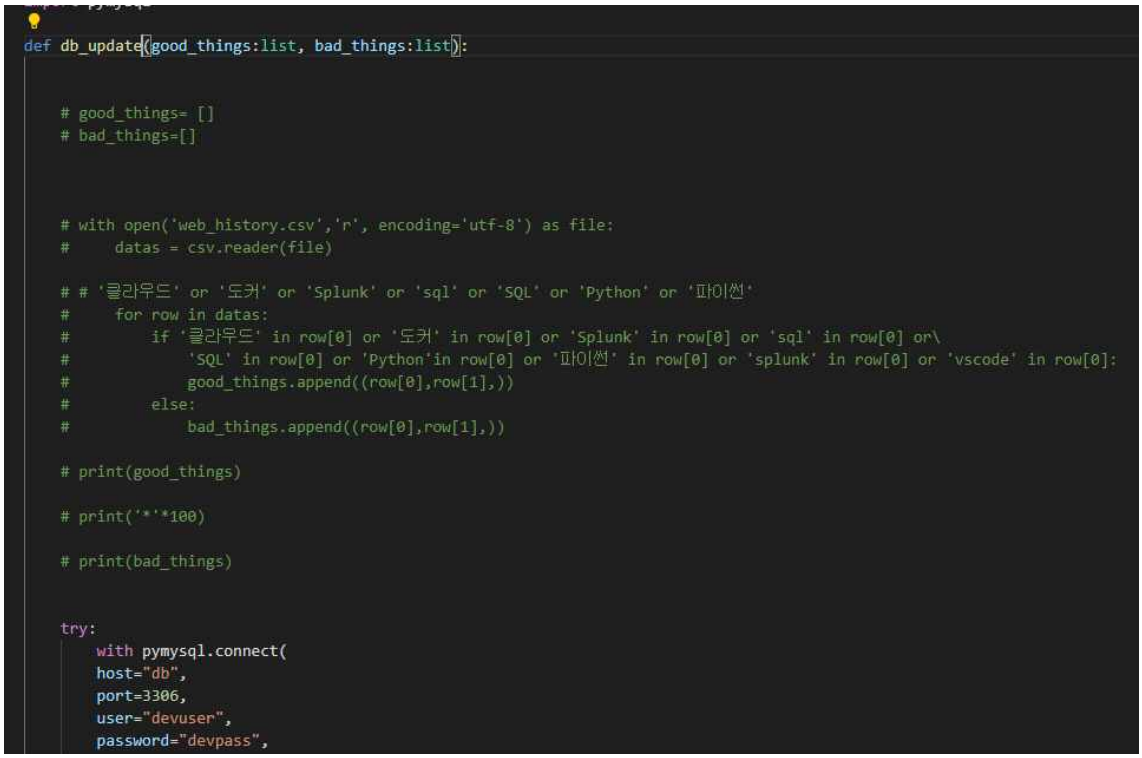
주의 : 주석처리된 부분이 진짜 코드입니다 (현재 월, 일의 오전 9시 이후로 히스토리 수집) 밑에 부분은 지우시고 주석된 코드 주석 해제하셔야해요. 밑에부분은 새벽에 저희 만들고 있을때 히스토리 자료가 없으니깐 임의로 한거, csv 파일 모자라다 싶으면 월이랑 일 조절해가면서 csv 파일 수집하세요

try 밑으로는 web_history.csv 파일 만들어서 쿼리로 불러온 히스토리 정보를 작성하는것

```
def connection():
    try:
        s = socket.socket()

        host = 'localhost'
        port = 8000

        s.connect((host,port))
        try:
            with open('web_history.csv','rb') as f:
                data = f.read(1024)
                while data:
                    s.sendall(data)
                    data = f.read(1024)

            print('file transfer complete')
        except Exception as e:
            print('file transfer error')
            print(e)



    except Exception as e:
        s.close()
        print('Connection Close')
        sys.exit()
```

connection : 크롬 히스토리 파일을 csv로 다 만들고 나면 서버와 연결(s.connect), 포트번호 9999로 수정하세요 사진. csv파일을 오픈해서 1024바이트씩 읽어온 후 sendall 함수를 통해 서버로 데이터 전송(바이트 스트림으로 전송)

crawling_csv.py



```
def db_update(good_things:list, bad_things:list):


    # good_things= []
    # bad_things=[]



    # with open('web_history.csv','r', encoding='utf-8') as file:
    #     datas = csv.reader(file)

    # # '클라우드' or '도커' or 'Splunk' or 'sql' or 'SQL' or 'Python' or '파이썬'
    #     for row in datas:
    #         if '클라우드' in row[0] or '도커' in row[0] or 'Splunk' in row[0] or 'sql' in row[0] or\
    #             'SQL' in row[0] or 'Python'in row[0] or '파이썬' in row[0] or 'splunk' in row[0] or 'vscode' in row[0]:
    #             good_things.append((row[0],row[1],))
    #         else:
    #             bad_things.append((row[0],row[1],))

    # print(good_things)

    # print('*'*100)

    # print(bad_things)


    try:
        with pymysql.connect(
        host="db",
        port=3306,
        user="devuser",
        password="devpass",
```

주의 : 주석처리된 부분은 제가 임의로 서버쪽에서 처리하도록 만든거라 crawling_csv 파일에선 사용하지 않습니다. crawling_csv 파일은 DB서버(mysql)와 연동하는 용도다 라고 설명하심 될거 같습니다. 정민씨가 코드 짰으니깐 잘 설명하시리라 믿습니다 문서작업 하실때는 저기 주석처리된 부분 지우시고 하세요!

추가요청사항

혹시 시간 되시면 크론탭이랑 윈도우 작업 스케쥴러 좀 해보셔야할거 같습니다. 이번 프로젝트가 '주기적'으로 정보를 크롤링 해야한다니깐 크롤링 서버에서는 히스토리 서버 파일을 크론탭을 통해 뭐 오전 9시 기준, 1시간단위나 10분단위 등으로 주기적으로 실행시키도록, 히스토리 클라이언트 파일은 윈도우 작업스케줄러를 통해 오전 9시 기준 1시간 단위든 10분단위든 주기적으로 실행시키도록 했다고 말하세요 아마 준형씨가 이거 잘 알거 같습니다. 크론탭 준형씨에게 물어보세요. 크게 어렵지는 않고 그냥 명령어만 치면 해결되니깐 이 부분까지 잘 소명하세요! 발표 잘하시고 건승하십쇼