# 디지털 포렌식 증거 시각화 프로젝트

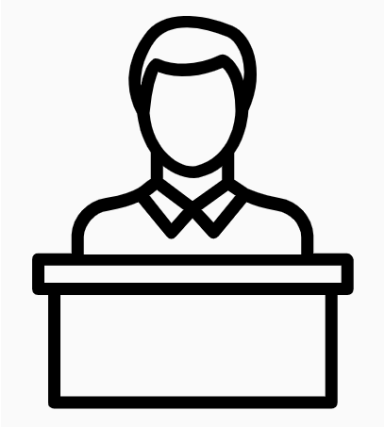
2020-1학기 종합설계프로젝트2

**11팀**

2017113030 김예린
2017114553 김경숙
2016112088 이현섭
2016113900 채한빈

# 팀원 소개

## 멘토



**강대명**
BeNX / SW엔지니어

## 멘티

**김예린 (팀장)**
컴퓨터학부 17학번

**김경숙**
컴퓨터학부 17학번

**이현섭**
컴퓨터학부 16학번

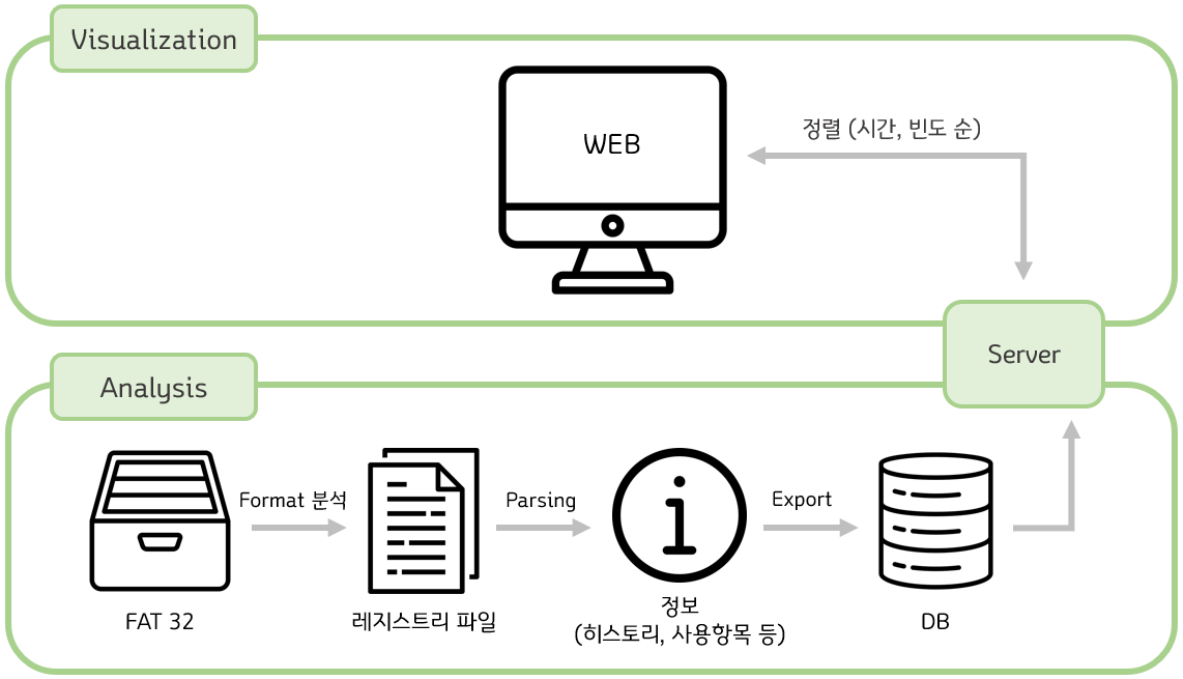
**채한빈**
컴퓨터학부 16학번

# Contents

# 1. 프로젝트 분석 및 설계 내용

## 1 프로젝트 분석

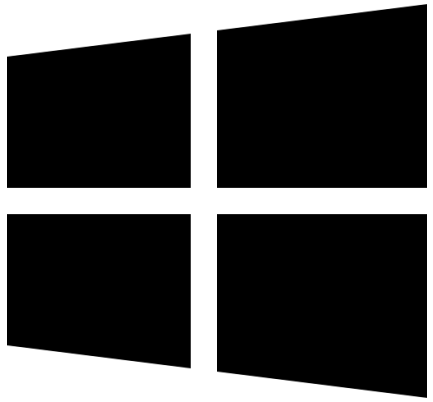
- 디스크 이미지 파일에서 레지스트리 하이브 파일 찾기
- 시스템 안에서 일정 시간동안 파일이 변화된 모습을 제공
- DB와 서버를 사용해서 접근 기록, 레지스트리 정보 등을 정렬해서 웹에 시각화



# 1. 프로젝트 분석 및 설계 내용

## 1 프로젝트 분석

- 하이브 파일 분석의 필요성



“윈도우 시스템 분석의 필수 요소”

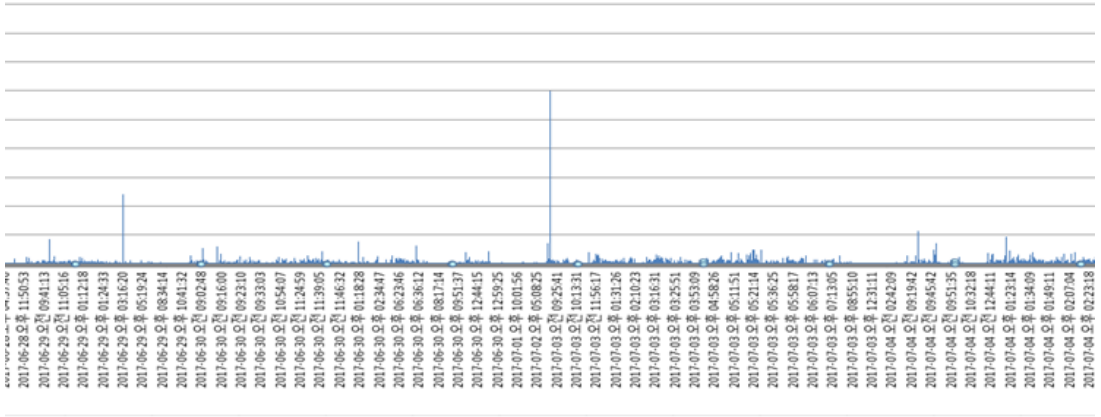
- 운영체제 정보, 사용자 계정 정보, 시스템 정보, 응용프로그램 실행 흔적, 최근 접근 문서 등을 분석 가능
- 자동 실행 항목 분석, 악성코드 탐지
- 저장매체 사용 흔적 분석(하드디스크, CD-ROM, USB 등)

➡ 시스템 안에서 파일이 변화된 모습을 보여줌

# 1. 프로젝트 분석 및 설계 내용

## 1 프로젝트 분석

- 타임라인 분석



- 파일 정보 및 로그 분석

| Serial number | Usb device name | Hardware Id | Last write time (UTC) |
|---|---|---|---|
| 4EB30900FFFF03CE | USB DISK Pro USB Device | USBSTOR₩Disk_____USB | 2017-06-20 오전 10:29:58 |
| B290409762 | BUFFALO HD-PCTU3 USB Device | USBSTOR₩DiskBUFFALO_HD | 2017-06-20 오전 10:29:58 |
| 0350900000000F34 | General USB Flash Disk USB Device | USBSTOR₩DiskGeneral_USB | 2017-06-20 오전 10:29:58 |
| AA04012700019688 | IOCELL Castella USB Device | USBSTOR₩DiskIOCELL__Cas | 2017-06-20 오전 10:29:58 |
| G265M8B10TG9M182 | ipTIME External USB Device | USBSTOR₩DiskipTIME__Exte | 2017-06-20 오전 10:29:58 |
| 14006429030OA8 | PHD 3.0 Silicon-Power USB Device | USBSTOR₩DiskPHD_3.0_Silic | 2017-06-20 오전 10:29:58 |
| 0708467E1A90FF66 | Philips USB Flash Drive USB Device | USBSTOR₩DiskPhilips_USB_ | 2017-06-20 오전 10:29:58 |
| 90277DDC17000027 | Samsung S3 Portable USB Device | USBSTOR₩DiskSamsung_S3 | 2017-06-20 오전 10:29:58 |

- **타임 라인**
  : 파일의 생성, 수정, 삭제, 접근 시간이나 웹 접속 기록 등 시간 정보 시각화

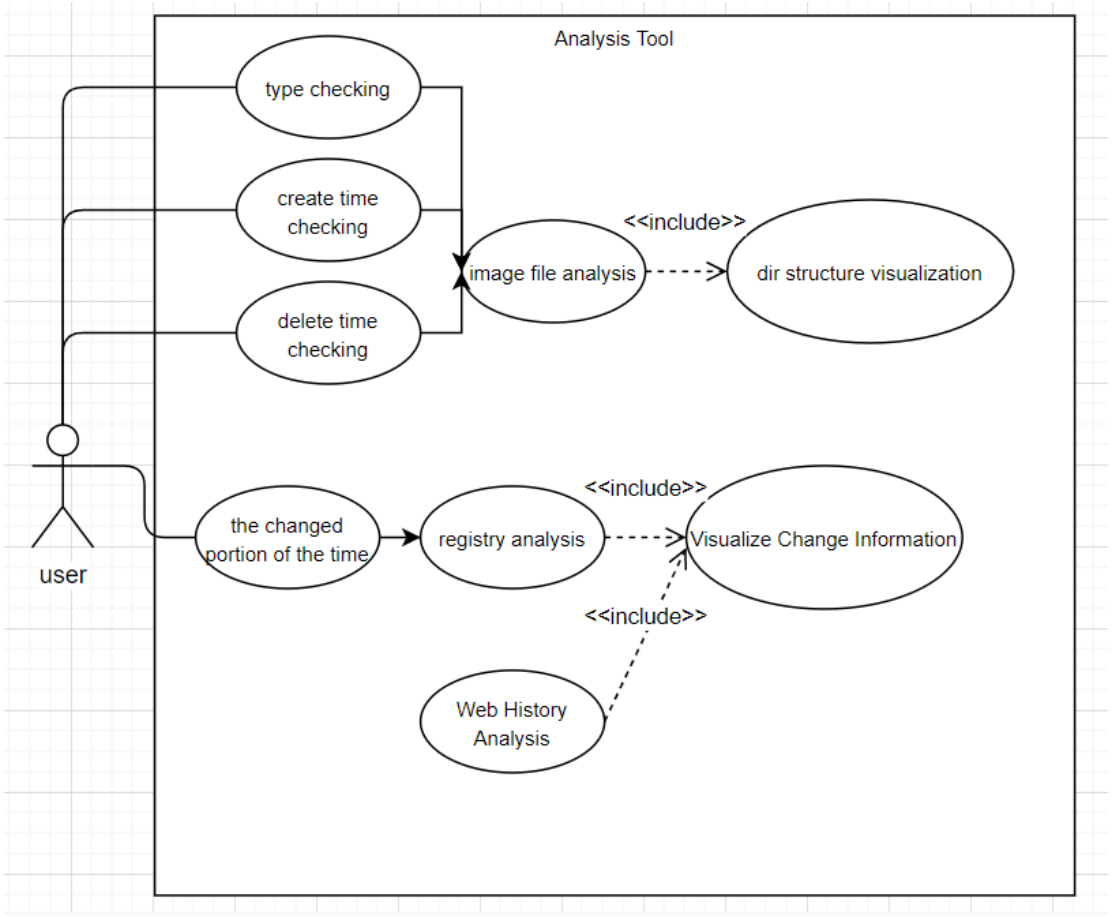
- **저장매체의 대용량화**
  : 모든 파일을 Eye Check로 분석하는 것은 현실적으로 불가능하기에 이러한 기능이 요구됨

- **파일 정보 제공**
  : 파일의 이름, 확장자, 크기, 위치 등

- **파일 로그 분석**
  : 파일 변경 정보, 삭제 유무, 확장자 변경 유무 등

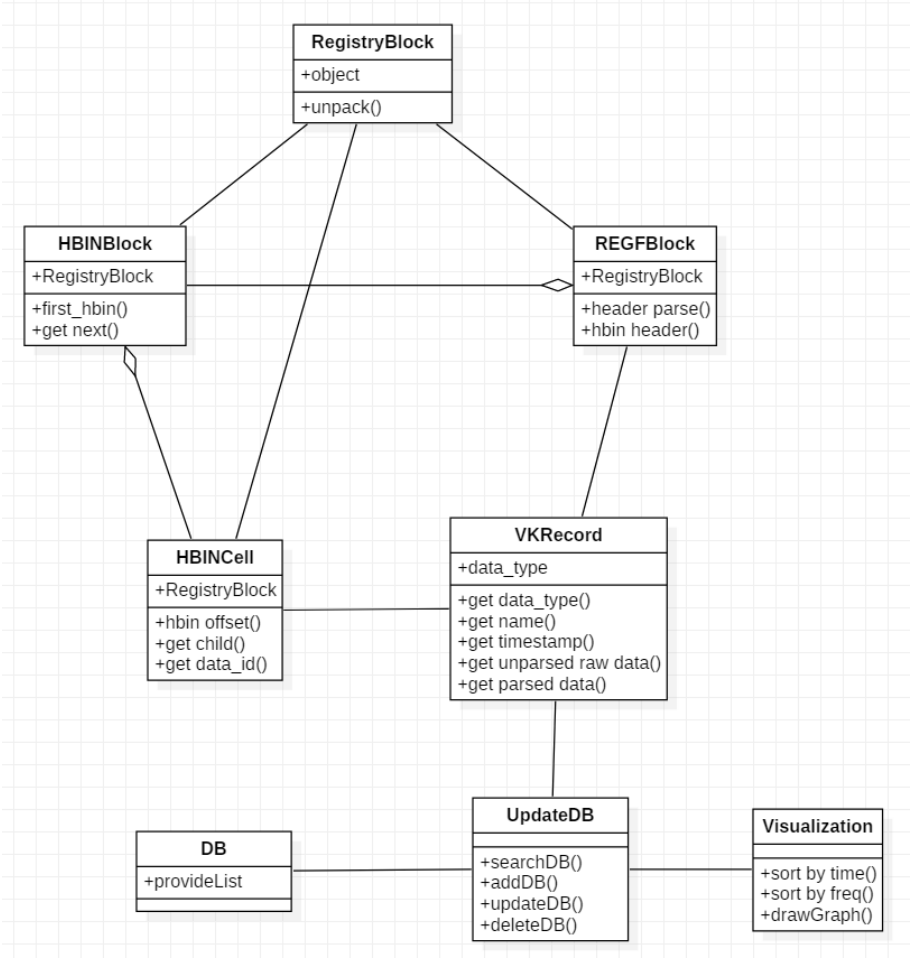
# 1. 프로젝트 분석 및 설계 내용

## 2 프로젝트 설계 – Use-case Diagram



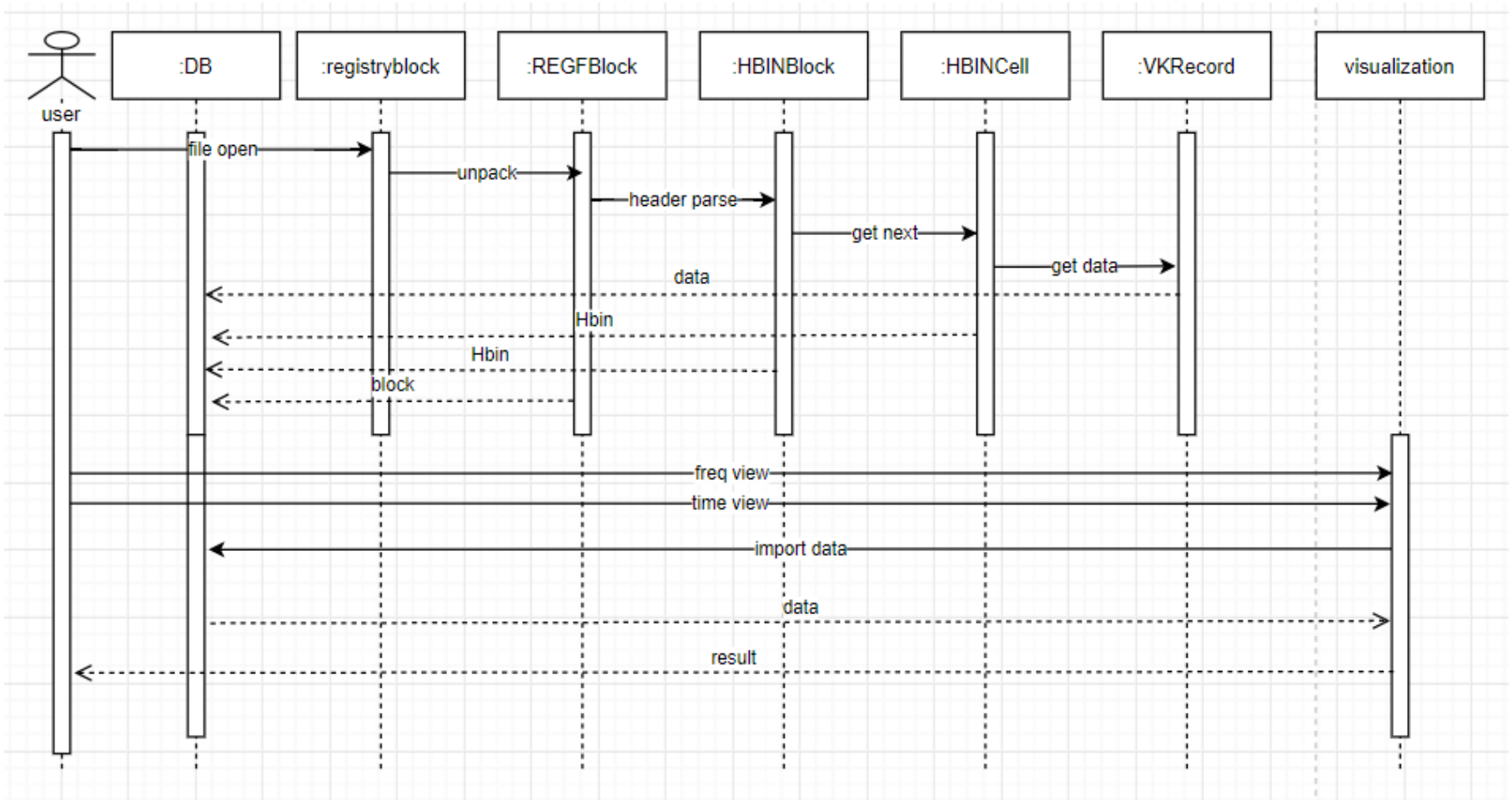
# 1. 프로젝트 분석 및 설계 내용

## 2 프로젝트 설계 – Class Diagram



# 1. 프로젝트 분석 및 설계 내용

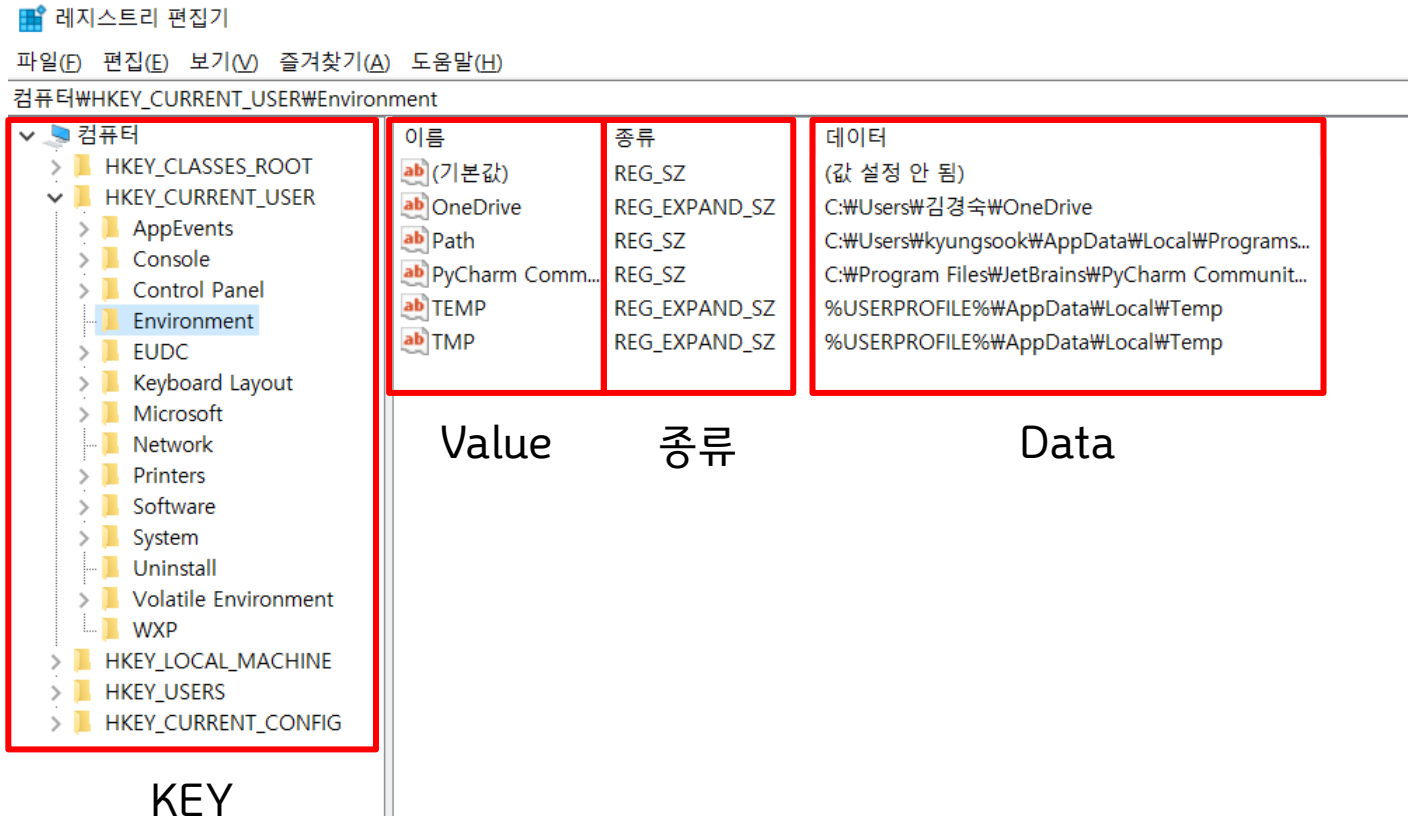
3 프로젝트 설계 – 하이브 파일 분석 Sequence Diagram



# 2. 진행 상황 및 개발 현황

## 1 진행 상황

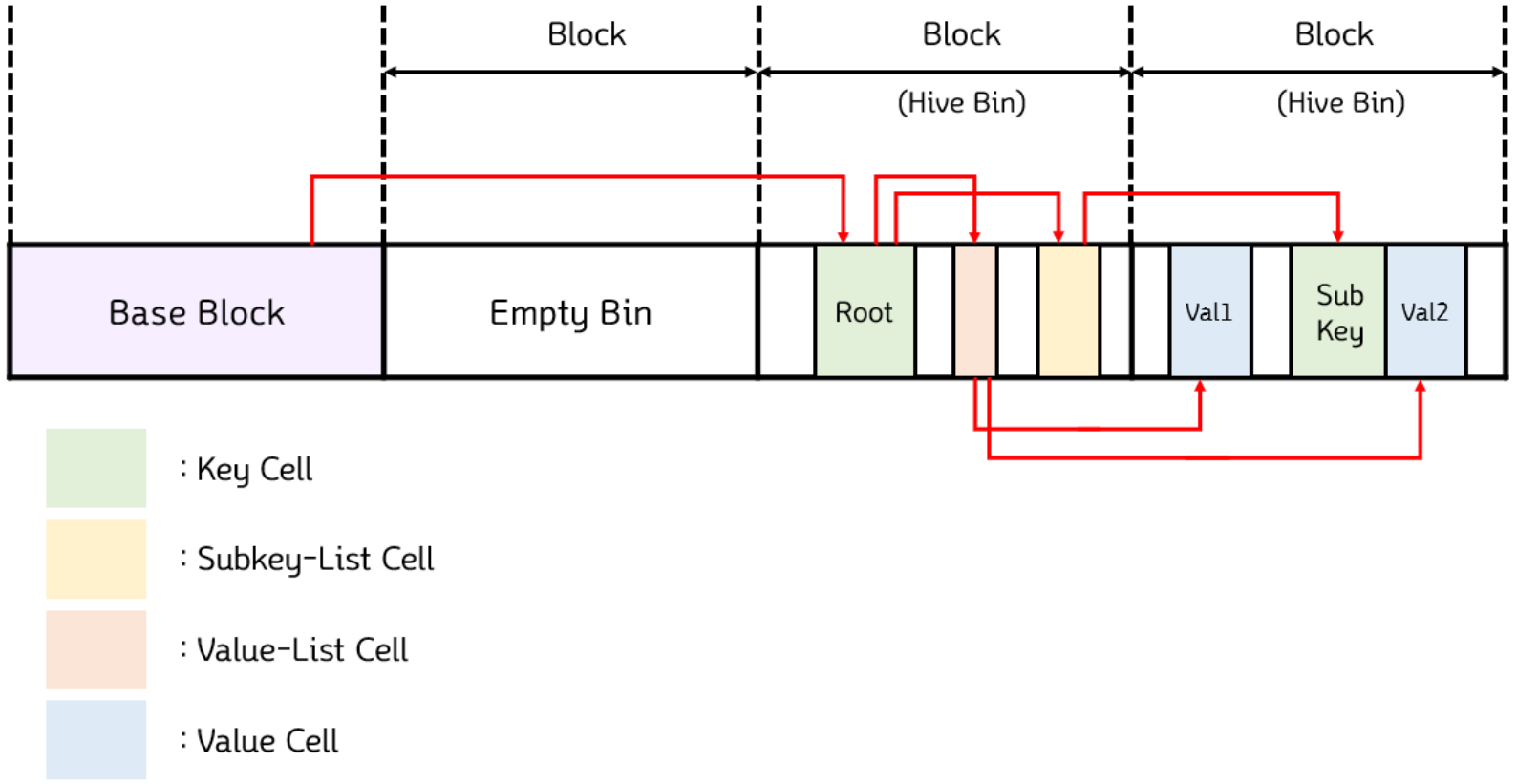
- Regedit을 통해 레지스트리 하이브 계층 구조 이해



# 2. 진행 상황 및 개발 현황

## 1 진행 상황

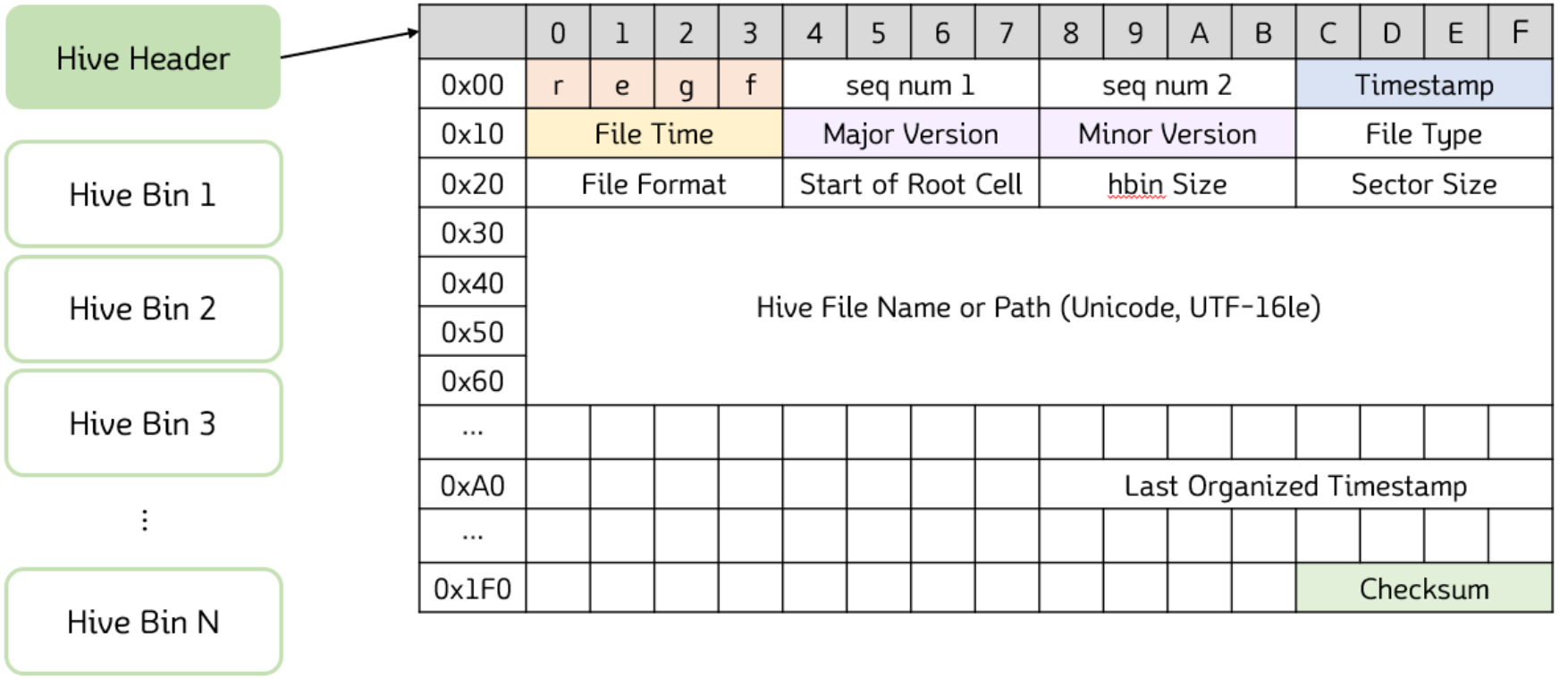
- 레지스트리 하이브 파일 구조 분석



# 2. 진행 상황 및 개발 현황

## 1 진행 상황

- Hive Header 분석을 통해 레지스트리 정보 추출




| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | r | e | g | f | seq num 1 | | | | seq num 2 | | | | Timestamp | | | |
| 0x10 | File Time | | | | Major Version | | | | Minor Version | | | | File Type | | | |
| 0x20 | File Format | | | | Start of Root Cell | | | | hbin Size | | | | Sector Size | | | |
| 0x30 | Hive File Name or Path (Unicode, UTF-16le) | | | | | | | | | | | | | | | |
| 0x40 | | | | | | | | | | | | | | | | |
| 0x50 | | | | | | | | | | | | | | | | |
| 0x60 | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | |
| 0xA0 | | | | | | | | | Last Organized Timestamp | | | | | | | |
| ... | | | | | | | | | | | | | | | | |
| 0x1F0 | | | | | | | | | | | | | Checksum | | | |

# 2. 진행 상황 및 개발 현황

## 2 개발 현황 (1/5)

**RegistryBlock()** : input으로 받은 윈도우 레지스트리 파일의 내용 unpack

(읽어온 바이너리를 offset부터 length만큼 잘라서 버퍼에 저장)

```
class RegistryBlock(object):
    def __init__(self, buf, offset, parent):
        self._buf = buf
        self._offset = offset
        self._parent = parent

    def unpack_binary(self, offset, length):
        return self._buf[self._offset + offset:self._offset + offset + length]
```

# 2. 진행 상황 및 개발 현황

## 2 개발 현황 (2/5)

```
class REGFBlock(RegistryBlock):
    def __init__(self, buf, offset, parent):
        super(REGFBlock, self).__init__(buf, offset, parent)
        _id = self.unpack_dword(0)

    def file_type(self):
        return FileType(self.unpack_dword(0x1C))

    def file_format(self):
        return self.unpack_dword(0x20)

    def hbins(self):
        h = HBINBlock(self._buf, self.first_hbin_offset(), self)
        yield h

        while h.has_next():
            h = h.next()
            yield h

    def first_log_entry_offset(self):
        return 0x200

    def log_entries(self):
        expected_seqnum = c_uint32(self.hive_sequence2())
        h = HvLEBlock(self._buf, self.first_log_entry_offset(), self)
        if h.sequence() == expected_seqnum.value and h.validate_log_entry():
            yield h

            while h.has_next():
                h = h.next()
                expected_seqnum.value += 1
```

**REGFBlock()**
: unpack한 윈도우 레지스트리 파일에서 header (base block)를 추출하여 내용 분석

- File type: 파일 타입
- File format: 파일 포맷
- Hive bins: 모든 hive bin들을 get
- Log entries: log 기록
- Timestamp: 마지막으로 수정된 시간

# 2. 진행 상황 및 개발 현황

## 2 개발 현황 (3/5)

```
class HBINBlock(RegistryBlock):
    def __init__(self, buf, offset, parent):
        super(HBINBlock, self).__init__(buf, offset, parent)
        _id = self.unpack_dword(0)
        self._reloffset_next_hbin = self.unpack_dword(0x8)
        self._offset_next_hbin = self._reloffset_next_hbin + self._offset

    def first_hbin(self):
        reloffset_from_first_hbin = self.unpack_dword(0x4)
        return HBINBlock(self._buf, (self.offset() - reloffset_from_first_hbin), self.parent())

    def has_next(self):
        regf = self.first_hbin().parent()
        if regf.hbins_size() + regf.first_hbin_offset() == self._offset_next_hbin:
            return False
        else:
            self.next()
            return True

    def next(self):
        return HBINBlock(self._buf, self._offset_next_hbin, self.parent())

    def cells(self):
        c = HBINCell(self._buf, self._offset + 0x20, self)

        while c.offset() < self._offset_next_hbin:
            yield c
            if c.offset() + c.size() == self._offset_next_hbin:
                break
            c = c.next()
```

**HBINBlock()**
: 윈도우 레지스트리 파일의 첫 hive bin block부터 연결된 모든 hive bin 찾음

- First hive bin: 제일 앞 hive bin
- Next hive bin: 연결된 다음 hive bin
- Cells: hive bin의 cell을 get

# 2. 진행 상황 및 개발 현황

## 2 개발 현황 (4/5)

```
class HBINCell(RegistryBlock):
    def __init__(self, buf, offset, parent):
        super(HBINCell, self).__init__(buf, offset, parent)
        self._size = self.unpack_int(0x0)

    def child(self):
        id_ = self.data_id()

        if id_ == b"vk":
            return VKRecord(self._buf, self.data_offset(), self)
        elif id_ == b"nk":
            return NKRecord(self._buf, self.data_offset(), self)
        elif id_ == b"lf":
            return LFRecord(self._buf, self.data_offset(), self)
        elif id_ == b"lh":
            return LHRecord(self._buf, self.data_offset(), self)
        elif id_ == b"li":
            return LIRecord(self._buf, self.data_offset(), self)
        elif id_ == b"ri":
            return RIRecord(self._buf, self.data_offset(), self)
        elif id_ == b"sk":
            return SKRecord(self._buf, self.data_offset(), self)
        elif id_ == b"db":
            return DBRecord(self._buf, self.data_offset(), self)
        else:
            return DataRecord(self._buf, self.data_offset(), self)
```

**HBINCell()**
: 윈도우 레지스트리 파일의 Hive bin 안에 있는 Cell을 파싱하여 signature에 따라 분류

- Size: cell의 크기
- Next cell: 다음 cell을 반환
- Raw data: cell에 포함된 raw data를 get
- Child: signature를 비교해 레코드 별로 처리

# 2. 진행 상황 및 개발 현황

## 2 개발 현황 (5/5)

```python
class VKRecord(Record):
    def __init__(self, buf, offset, parent):
        super(VKRecord, self).__init__(buf, offset, parent)
        _id = self.unpack_string(0x0, 2)

    def name(self):
        name_length = self.unpack_word(0x2)
        unpacked_string = self.unpack_string(0x14, name_length)
        if self.has_ascii_name():
            return unpacked_string.decode("windows-1252")
        return unpacked_string.decode("utf-16le")

    def timestamp(self):
        if self.has_timestamp():
            return parse_windows_timestamp(struct.unpack_from(str("<Q"), self.raw_data()[-8:])[0])

    def data_type_str(self):
        data_type = self.data_type()

        if data_type == RegSZ:
            return "RegSZ"
        elif data_type == RegExpandSZ:
            return "RegExpandSZ"
        elif data_type == RegBin:
            return "RegBin"
        elif data_type == RegDWord:
            return "RegDWord"
        elif data_type == RegMultiSZ:
            return "RegMultiSZ"
        elif data_type == RegQWord:
            return "RegQWord"
```

**VKRecord()**

: Hive bin cell의 key, value 값을 포함한 레코드를 파싱하고 분석

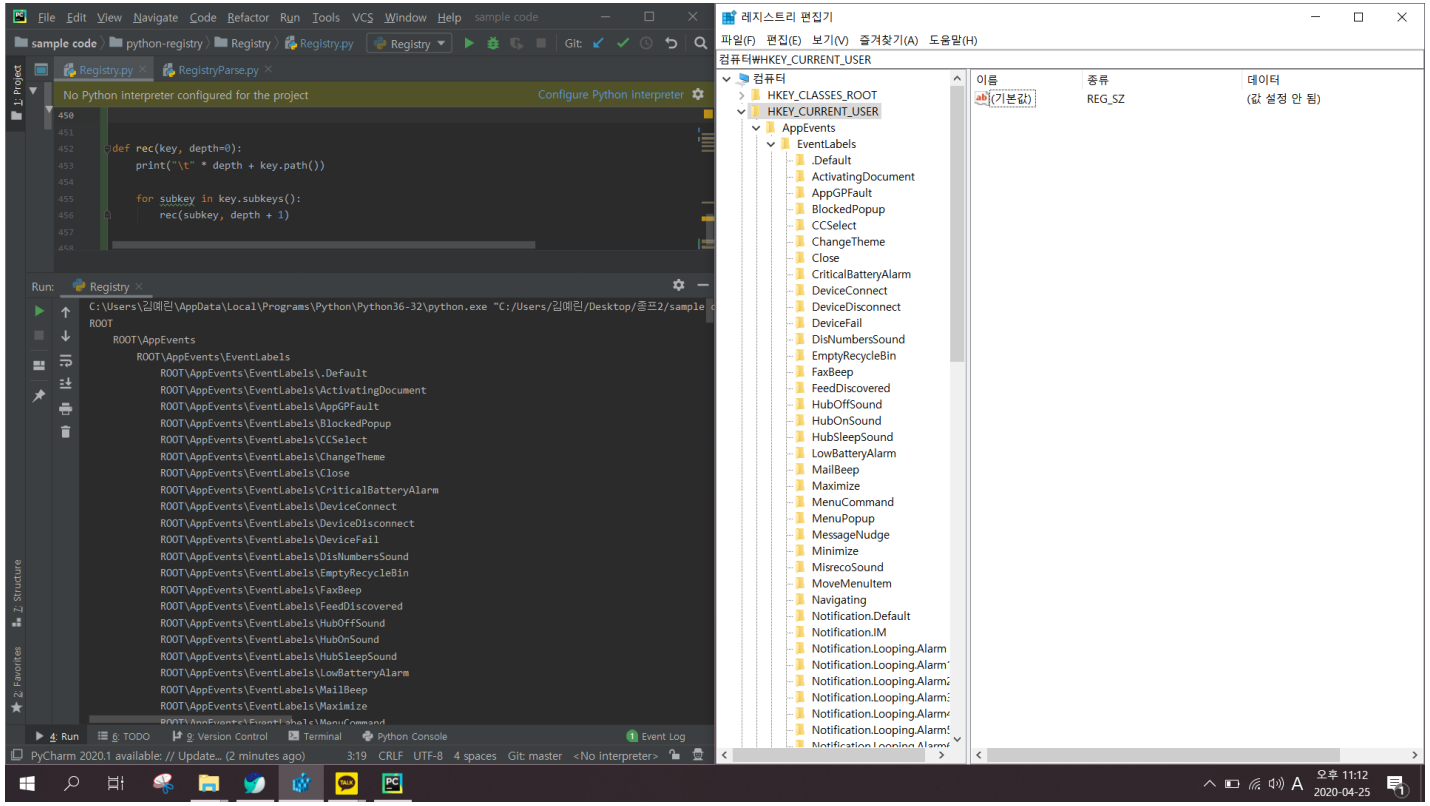
- Data type: value의 데이터 타입
- Name: value의 이름
- Timestamp: 수정된 시간
- Data length: 길이
- Data: 내용

# 2. 진행 상황 및 개발 현황

## 3 실행 결과 화면

RootKey인 **HKEY_CURRENT_USER**의 하이브 파일 분석

: 윈도우에서 제공하는 레지스트리 편집기의 트리구조와 동일한 결과 출력 확인 가능



# 3. 과제 추진 일정

1. 현재 진행 상황 (3월~4월)

| | 3월 | | 4월 | | | | |
|---|---|---|---|---|---|---|---|
| | 3주 | 4주 | 1주 | 2주 | 3주 | 4주 | 5주 |
| 프로젝트 환경 구축 | | | | | | | |
| 기존 코드 리팩토링 | | | | | | | |
| 윈도우 레지스트리 구조 공부 | | | | | | | |
| 레지스트리 분석 기능 구현 | | | | | | | |
| 기존 코드와 merge 작업 | | | | | | | |
| 시각화 기능 구현 (DB, 웹 연동) | | | | | | | |
| 논문 작성 | | | | | | | |
| Testing & Debugging | | | | | | | |
| 문서작업 (계획서, 보고서) | | | | | | | |

# 3. 과제 추진 일정

## 2 추후 진행 상황 (5월~6월)

| | 5월 | | | | | 6월 | |
|---|---|---|---|---|---|---|---|
| | 1주 | 2주 | 3주 | 4주 | 5주 | 1주 | 2주 |
| 프로젝트 환경 구축 | | | | | | | |
| 기존 코드 리팩토링 | | | | | | | |
| 윈도우 레지스트리 구조 공부 | | | | | | | |
| 레지스트리 분석 기능 구현 | | | | | | | |
| 기존 코드와 merge 작업 | | | | | | | |
| 시각화 기능 구현 (DB, 웹 연동) | | | | | | | |
| 논문 작성 | | | | | | | |
| Testing & Debugging | | | | | | | |
| 문서작업 (계획서, 보고서) | | | | | | | |

# 3. 과제 추진 일정

## 3 프로젝트 세부 계획

(1) 레지스트리 분석 기능 구현

- 레지스트리 파일 파싱
- 레지스트리 트리 구조 get
- Key값과 그에 해당하는 Value값 get

(2) 기존 코드에 merge 작업

(3) 시각화 기능 구현 (DB, 웹 연동)

- Get한 정보를 DB에 저장
- AWS를 이용하여 서버 구축 후 웹 제작
- DB에 저장된 내용을 웹에 시각화 하여 출력

(4) 논문 작성

- IPACT 2020 국내학술대회 (20.06.26)
- 논문 제출 마감 : 20.05.22 (금)

(5) Testing & Debugging

- 샘플 디스크 이미지 파일을 이용하여 테스트 및 기능 개선

# 3. 과제 추진 일정

## 3 팀원 업무 분장

### 김예린

Team Leader

- 전체 프로젝트 관리
- 레지스트리 파일 포맷 분석

### 김경숙

Designer

- 시각화 UI/UX 연구개발
- 웹 히스토리 분석

### 이현섭

Programmer

- 빈도 시각화 기능 구현
- 증거 타임라인 구현

### 채한빈

Programmer

- 변경 정보 시각화 기능 구현
- 웹/DB 연동

# 4. 이슈사항

(1) 코로나19

- 멘토님과 오프라인 미팅을 가지지 못하여 초반 과제에 대한 이해도 부족
- 타지역에 사는 팀원이 있어서 오프라인 회의에 어려움이 있음

(2) 디스크 이미지 파일

- 레지스트리 파일과 웹 히스토리가 저장된 디스크 이미지 파일이 없어서 테스트가 어려움
- 레지스트리 편집기에서 RootKey의 하이브 파일을 따서 직접 디스크 이미지 파일 제작

(3) DB와 웹 구현 및 AWS

- DB와 웹 구축 경험이 없어서 구현 전 공부가 필요함
- AWS 경험 부족

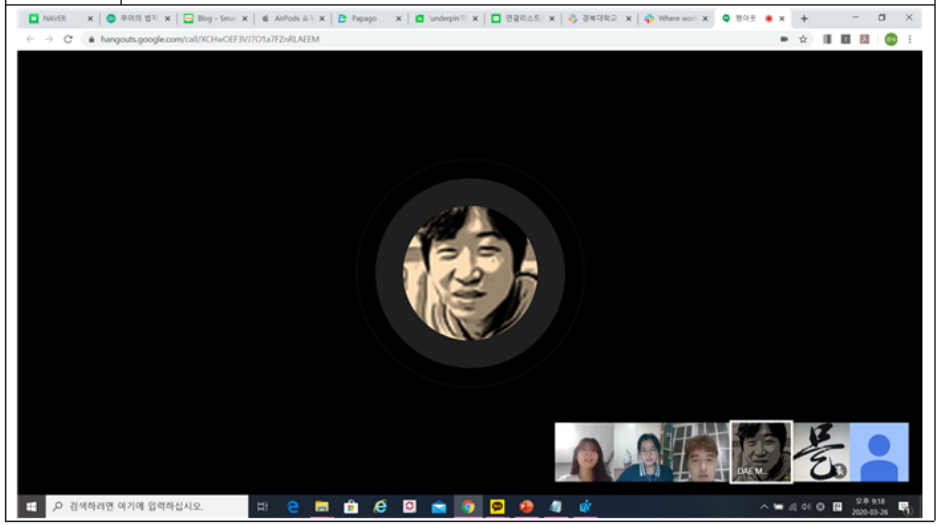
# 5. 멘토링 내용

## Mentoring #1 (2020.03.17)

| 2차시 | 날짜 | 2020.03.17.(화) | 장소 | 각자 자택 | 기록자 | 김경숙 |
|---|---|---|---|---|---|---|
| 회의내용 | 1. 프로젝트 방향: 종프 1에서 만든 프로그램 기반으로 한다. fat32에 저장되어있는 파일을 분석해서 시각화<br>2. 개발은 UI와 분석으로 진행<br>3. 레지스트리 찾는 건 직접 개발해야할 듯<br>4. 멘토님 참여 정기 화상회의 일정 잡기: 목요일 저녁 9시<br>5. 개발, 분석은 파이썬으로, 보여주는 것은 웹으로 해도 될 듯<br>6. 중간중간에 학습하고 있는 것을 블로그 등에 남기는게 좋을 듯 | | | | | |

## Mentoring #2 (2020.03.26)

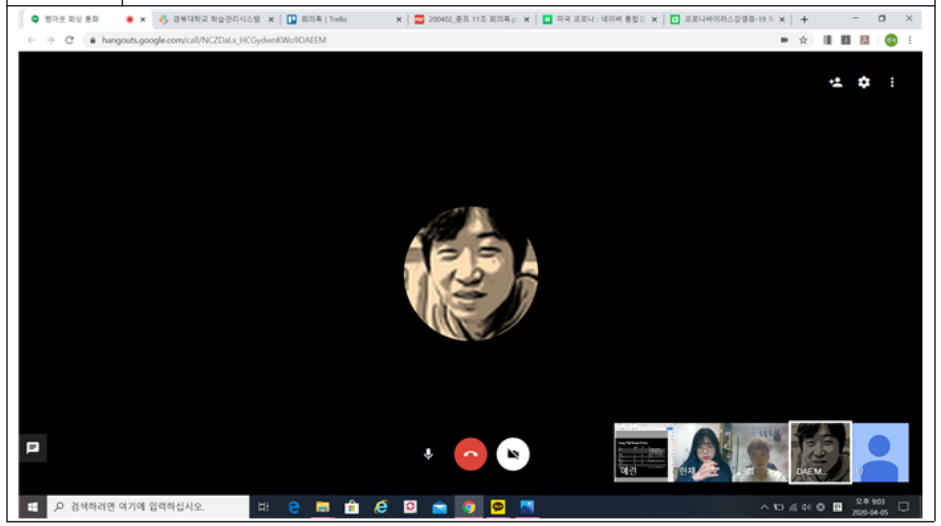
| 5차시 | 날짜 | 2020.03.26.(목) | 장소 | 각자 자택 | 기록자 | 김경숙 |
|---|---|---|---|---|---|---|
| 회의내용 | 1. fat32 파일 시그니처를 분석해서 레지스트리 파일을 찾아낸다.<br>2. 레지스트리 파일을 파싱해서 정보를 뽑아내야 한다.<br>- 웹 히스토리<br>- 윈도우의 어떤 파일을 사용했는지<br>- 실행 정보가 어디있는지<br>3. export 기능: zip파일로 뽑고, web에 보여줘야 하기 때문에 DB에 저장해야함<br>4. 시각화를 어떻게 해야할지는 고민을 해야할 듯! | | | | | |



# 5. 멘토링 내용

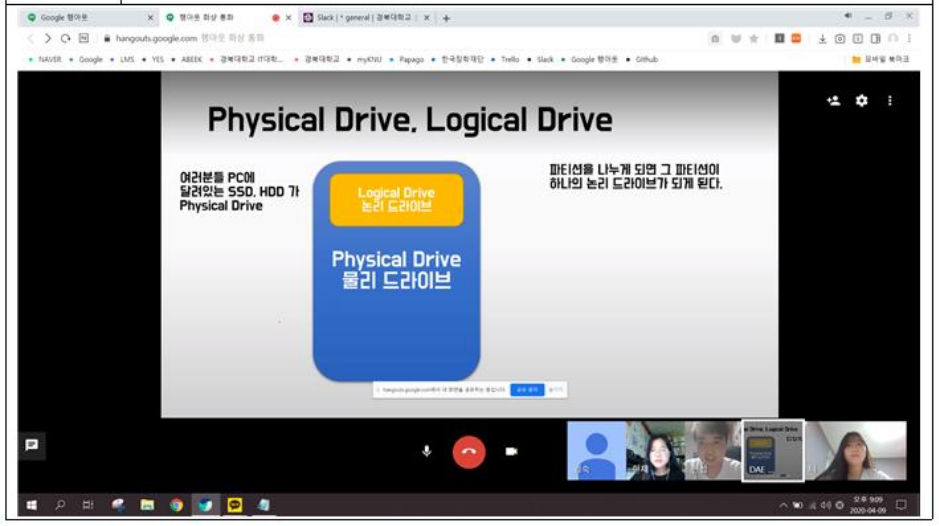
## Mentoring #3 (2020.04.05)

| 9차시 | 날짜 | 2020.04.05.(일) | 장소 | 각자 자택 | 기록자 | 김경숙 |
|---|---|---|---|---|---|---|
| 회의내용 | 1. full file list 전체 파일 목록 다 가져오는거<br>2. 모든 파일 목록 파일 시그니처 뽑아서 어떤 타입인지 알아내는거<br>3. 목록에서 시간정보로 특정기간에 선택한거 접근순서같은거 알아내기<br>4. 웹 히스토리 내용 어떻게 저장되고 원래 경로가 윈도우에서 어디 있는지<br><br>+ 필터기능 - 이미지파일만, 한글파일만 볼수있도록<br><br>목적 : 특정 시간, 어떤 식으로 파일에 접근했느냐? | | | | | |



## Mentoring #4 (2020.04.09)

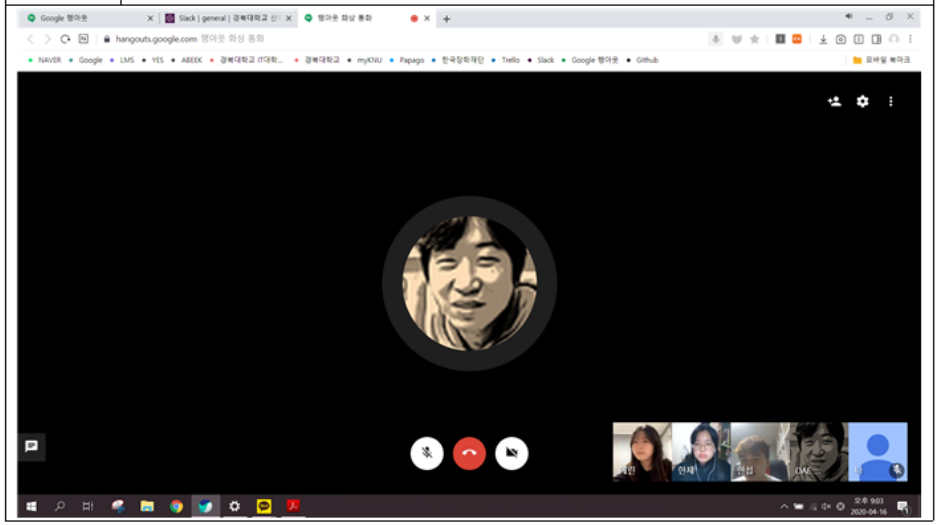
| 11차시 | 날짜 | 2020.04.09.(목) | 장소 | 각자 자택 | 기록자 | 김경숙 |
|---|---|---|---|---|---|---|
| 회의내용 | 1. 멘토님 레지스트리 강의<br>2. 레지스트리 파일 시그니처<br>3. 레지스트리 분석 방법 | | | | | |



# 5. 멘토링 내용

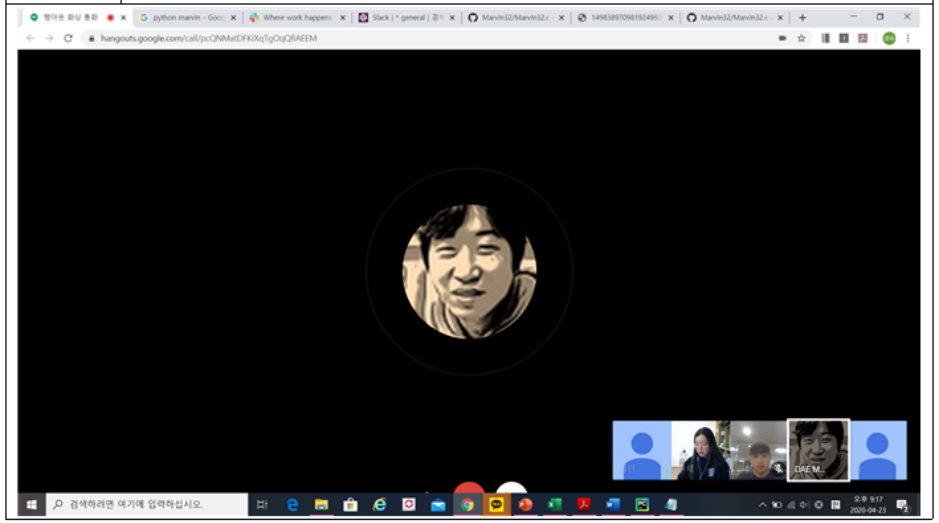
## Mentoring #5 (2020.04.16)

| 13차시 | 날짜 | 2020.04.16.(목) | 장소 | 각자 자택 | 기록자 | 김예린 |
|---|---|---|---|---|---|---|
| 회의내용 | 1. 모르는 부분 질문<br>- 디스크 이미지 파일 읽는 방법 : 전부 다 읽어서 하는 방향으로 진행<br>- DB에는 어떤 정보를 넣는지 질문<br>: 시간정보, 파일 정보, 어떤 키인지, 파일 위치, 어떤 값이 나왔는지<br>- 루트키 모두 봐야하는지, HKEY_CURREN_USER만 보면 되는지<br>: 모두 봐야한다. 다섯 개의 루트 키 하이브 파일은 모두 같은 구조 | | | | | |



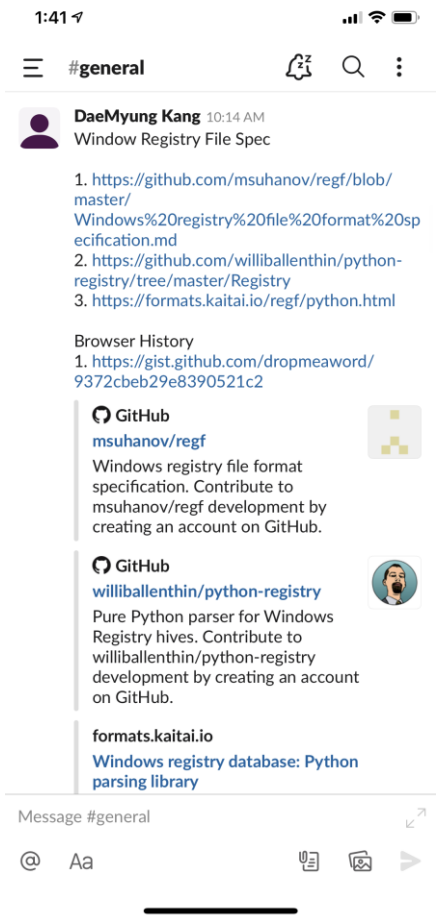
## Mentoring #6 (2020.04.23)

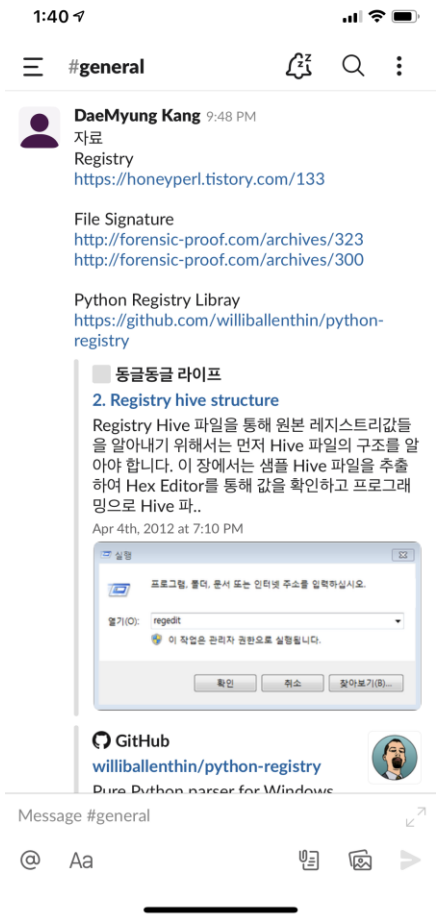
| 16차시 | 날짜 | 2020.04.23.(목) | 장소 | 각자 자택 | 기록자 | 김예린 |
|---|---|---|---|---|---|---|
| 회의내용 | 1. 모르는 부분 질문<br>- marvin32가 무엇인지<br>2. winDD로 디스크 이미지 제작이 안 되는 부분 해결<br>- USB를 100MB정도로 파티션 분할하여 제작해볼 것 (FAT32기반)<br>- 레지스트리 하이브 파일 우선 제작 후 담아서 이미지 파일 생성<br>3. 2020.04.25.(토)에 중간 발표 리허설 같이 봐주신 후 피드백 주신다 함 | | | | | |



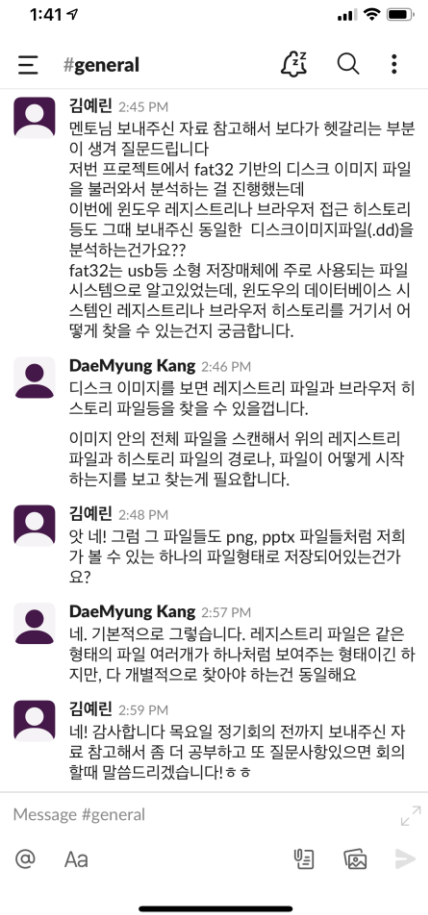
# 5. 멘토링 내용

## 1 Slack을 통한 참고 자료 공유




## 2 과제 수행 중 모르는 부분 질문



# 들어주셔서 감사합니다

11조