ADVENTURE DESIGN

ATTENDANCE ORGANIZER

문제 분석 및 설계

20200123
김다은

2021.10.11

Contents

PART
01
문제 분석

1. 소개

01. 입력 02P
02. 출력 02P
03. csv 파일이란? 02P

2. 현재 시스템

01. 개요 03P

3. 제안된 시스템

01. 자동 출결 처리 서비스 요구사항 03P
02. 자동 출결 처리 서비스 유스 케이스 05P

01 문제 분석

1. 소개

온라인 회의 플랫폼에서 제공하는 접속 기록을 사용해 출석부 정리를 한다. 접속 기록을 처리하여 각 학생별 출석상황을 쉽게 알 수 있도록 csv 파일로 저장하는 프로그램에 대한 문제 인식을 다룬다.

○입력

- 수업_분반.csv : 수강 학생 전원의 학번과 이름이 담김
- 수업_분반_날짜.csv : 미팅 참여자의 정보와 들어온 시간 나간 시간이 담김

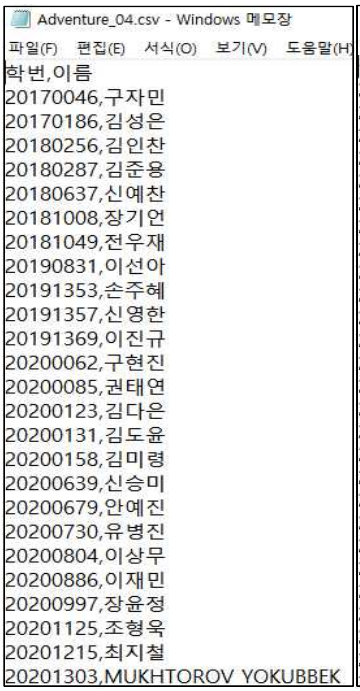
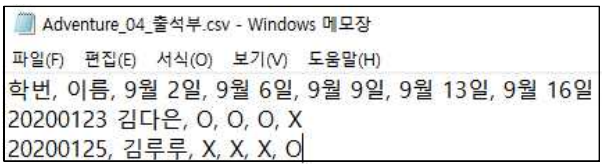


그림 3 학생 명단 기록 csv　　그림 4 접속 기록 csv 파일

○출력

- 수업_분반_출석부.csv : 학번 이름 그리고 날짜별 출석 현황을 알려줌



○ CSV 파일

- CSV(comma separated values) 파일이란 ‘,’로 분리되어 기록된 파일이다.

엑셀을 통해 파일을 읽거나 쓸 수 있다.

2. 현재 시스템

온라인 수업 플랫폼 zoom에서 채팅 기록을 기반으로 한 자동 출결 처리 프로그램이 존재한다. 채팅창에 글을 올린 것을 기반으로 스프레드 시트를 만들어 회의 주최자가 요구한 내용을 바르게 입력하였을 경우 출석으로 처리한다. 이는 특정 키워드로 구분한다. 예를 들어 주최자가 “뇌의 종류 5가지를 적어서 채팅창에 올리시오”라고 요구하였을 경우 “대뇌” 라는 키워드가 반드시 포함되어야 하는 식이다. 학생 구분을 편리하게 하기 위하여 들어오면 반드시 참가자 이름의 형식을 학번과 이름으로 바꾸도록 제한하였다.

○ 출력

그림 6 자동 출석 처리 결과 (출처 : https://sciencelove.com/2497)

3. 제안된 시스템

< 자동 출결 처리 서비스>

자동 출결 처리 서비스 요구 사항

날짜	버전	비고	작성자
2021-10-13	1.0	사용자 요구사항을 1차적으로 분석	김다은

용어 정리

서비스 개요

출석부 자동 처리 프로그램은 csv 파일을 분석하여 학생들의 출석을 자동으로 처리해 준다. 해당 수업의 전 수강생이 담긴 파일과 해당 수업의 접속 기록이 담긴 파일이 입력으로 들어온다. 이 프로그램에서는 학생의 정보를 효과적으로 탐색하기 위하여 내부

적으로 해시 테이블을 사용하기로 한다. 또한 csv 파일이 csv 파일 형식이 아니거나 정보를 조회할 수 없는 형식이라면 예외처리를 해준다. 출석부 자동 처리를 해준 뒤 학생의 이름을 선택하여 출석률을 확인하고 날짜를 선택하여 해당 수업의 출석률을 확인할 수 있다. 최종적인 출석 기록이 담긴 결과물은 수업_분반_출석부.csv 형식으로 저장된다. 출석부 자동 처리 프로그램은 GUI를 구현하여 결과물 csv 파일을 미리 확인하고 출결 확인을 더 용이하게 한다.

기능 요구 사항

○ 문제 해결 범위

- 학생이 표기한 이름에 따라 동일인이 여러 방식으로 표현 될 수 있다. 줌 기록은 이름이 우선이기 때문에 이름으로 먼저 구분한 뒤에 동명이인이 있다면 학번을 통해 분류한다. 출석체크의 편의성을 위해 학번을 사용할 수 있다.
- 한 사람의 이름이 여러 방식으로 표현될 수 있다. 동일인임을 처리하여 줘야한다.
ex) "Daeun Kim", "김다은“ ”20200123 김다은“
- 이름을 인식할 수 없는 경우 적절한 예외처리를 통해 처리 여부를 표시해야 한다.
- 사용자 접속기록이 여러 개일 경우 총 수강 시간을 계산하여 출석을 체크 한다.
- 주어지는 과목 및 접속 기록 파일의 수는 정해져 있지 않다. 프로그램 내부에서 파악하여 처리해 주어야 한다.
- 수업 시작 시간에서 +10분까지는 지각, 그 이후는 접속 기록이 있는 경우에도 결석으로 처리한다.
- 학생별 출석률과 날짜별 출석률을 계산하여 파일에 같이 기록한다.
- 파일을 선택, 수업 시작 시간 설정, 출결 상황 확인 등을 GUI를 통해 제공한다.

○ GUI

그림 7 GUI 구성

대체 흐름

○ 입력 오류

- 파일이 제대로 입력되지 않을 경우 예외처리를 해준다.

ex) 파일의 형식이 올바르지 않을 경우 (csv 파일이 아닐 경우), 파일의 정보가 프로그램의 의도와 다른 경우

- 참여자 파일과 접속 기록 파일의 수업이 일치하지 않을 경우 오류 메시지를 출력한다.

○ 동작 오류

- 수업 시작 시간이 파일의 이름과 다르거나 회의 주최자의 설정 시간과 크게 다를 경우 다시 한 번 확인 하라는 메시지를 출력한다.

- 작업이 완료되지 않은 상태에서 출석률과 csv 파일을 확인하려고 할 시에 메시지를 출력해 준다.

비기능 요구 사항

- 저장이 되지 않은 채로 종료될 때 저장을 하라는 메시지를 띄워준다.

- 확인하고 싶은 수업을 선택할 때 마다 파일 정보가 뜨는 부분 view를 업데이트 해준다.

- 파일을 처리하는 도중 파일이 삭제되거나 훼손되었을 때를 고려하여 예외처리를 해준다.

< 자동 출결 처리 서비스>

자동 출결 처리 서비스 유스 케이스 개요

날짜	버전	비고	작성자
2021-10-13	1.0	사용자 요구사항을 바탕으로 정리	김다은

서비스 개요

자동 출결 처리 서비스는 GUI를 통해 제공되며 입력받은 csv 파일의 정보를 통하여 출결 처리 서비스를 자동으로 제공하는 서비스이다. 출결 처리는 이름을 기반으로 하고 동명이인 잘못된 형식의 입력을 고려해 주어야 한다. 또한 이 서비스는 회의 주최자와 참여자를 구분할 필요가 있다.

시스템 문맥도

이 서비스는 사용자와의 역할을 고려하여 6가지의 유스 케이스로 정의할 수 있다. 파일 입력- 저장과 같은 파일 입출력을 위한 기능적인 부분과 특정 학생, 날짜별 출석률, 과목 선

택과 같은 사용자가 더 편리하게 정보를 확인할 수 있도록 도와주는 view를 바꾸어주는 부분으로 나눌 수 있다.

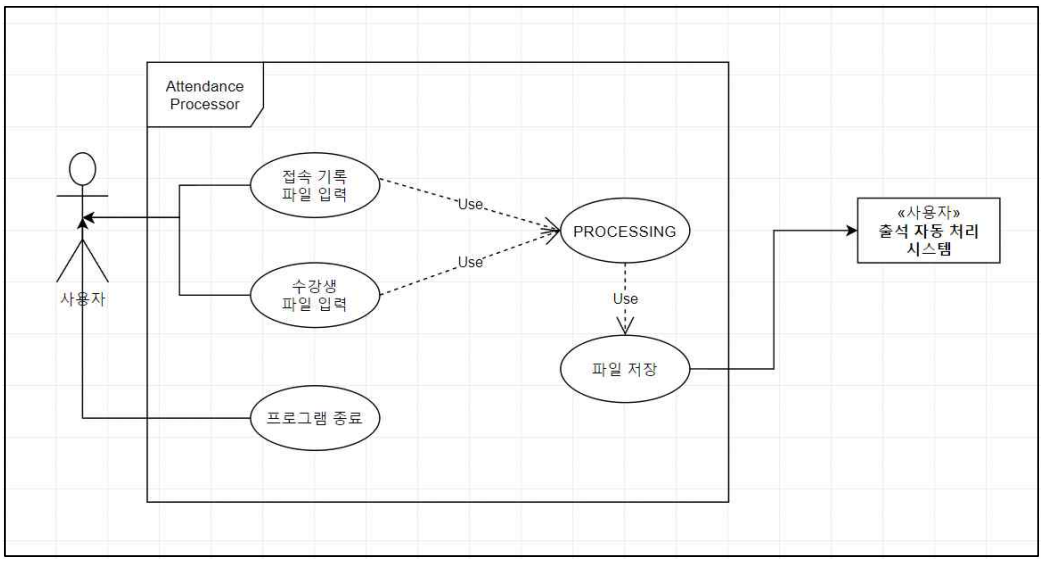


그림 8 Use case diagram

파일 입력 유스 케이스

접속 기록 파일과 수강생 파일을 받는다, 접속 기록을 처리해주고 학생 정보 관리를 위한 해시 테이블을 구성한다.

수업 시작 시간 설정 유스 케이스

사용자가 수업 시작 시간을 설정할 수 있다. 설정된 수업 시작 시간을 기준으로 10분 안까지는 지각 그 이상의 결석 처리된다. 만약 수업 시작 시간이 파일에 입력된 회의 주최자의 처음 시간과 다르다면 메시지를 통해 두 정보가 다름을 알려준다.

어떤 수업의 출석률을 확인할 것인지 선택하기

수업을 선택하면 model에서 선택된 수업에 해당하는 객체를 가져와 view를 업데이트 시켜준다. 만약 processing 이 되지 않았다면 processing 이 필요하다는 메시지를 띄우고 view는 업데이트를 하지 않는다.

특정 학생의 출석률 확인하기

수강생 파일을 통해 combo box 에 학생의 목록이 추가된다. 학생을 선택하고 해당 학생의 출석률을 확인할 수 있다.

특정 날짜별 출석률 확인하기

달력에서 확인하고 싶은 날짜를 클릭하면 해당 일자의 학생들 출석률을 확인할 수 있다. 만약 해당 날짜에 수업이 없었다면 그에 대한 메시지를 띄워주고 view는 업데이트 되지 않는다.

파일 저장하기

정상적으로 processing이 완료되었다면 사용자가 선택한 파일 경로에 파일이 저장되게 된다.

엑터 명세

일반 사용자

자동 출결 처리 서비스의 사용자 대부분을 의미하며 모든 제공되는 서비스에 대한 사용 권한을 가지고 있음.

Contents

PART
02
설계

1. 클래스 다이어그램

01. 클래스 다이어그램 07P

2. 절차 설계

01. 절차 설계 10P

02 설계

1. 클래스 다이어그램

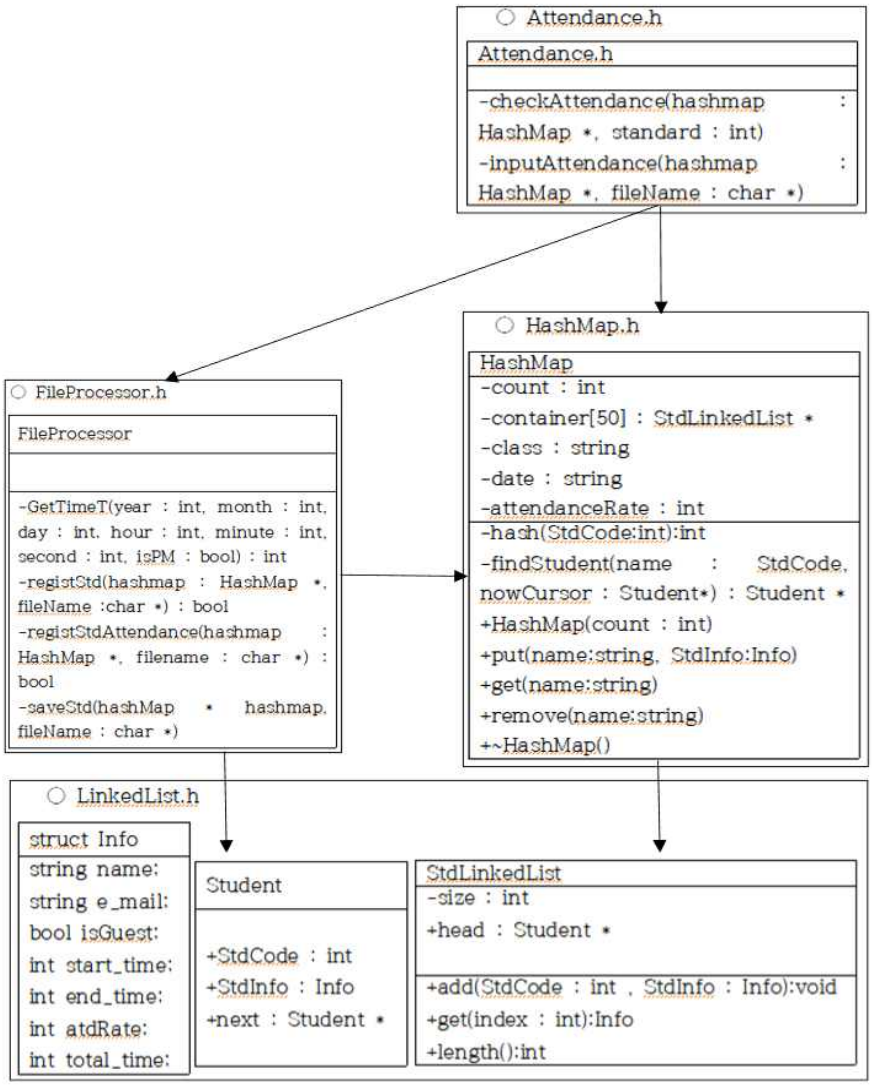


그림 10 클래스 다이어그램

○ LinkedList.h

struct Info
string name; string e_mail; bool isGuest; int start_time; int end_time; int atdRate; int total_time;

Student
+StdCode : int +StdInfo : Info +next : Student *

StdLinkedList
-size : int +head : Student *
+add(StdCode : int , StdInfo : Info):void +get(index : int):Info +length():int

- LinkedList.h 내부에 들어있는 클래스와 구조체들은 모두 해쉬 맵을 위한 것이다. StdLinkedList 는 학생을 구분하기 위한 key 값 그리고 정보가 담긴 Info 구조체로 이루어진 Student class를 가지고 각각의 학생 정보를 연결리스트로 연결, Student 의 값을 가져오기 위한 위한 함수가 담긴 클래스이다.
- StdLinkedList 는 HashMap을 구현할 때 사용되는 연결리스트이다.

○ HashMap.h

HashMap
-count : int -container[50] : StdLinkedList * -class : string -date : string
-hash(StdCode:int):int -findStudent(name : StdCode, nowCursor : Student*) : Student * +HashMap(count : int) +put(name:string, StdInfo:Info) +get(name:string) +remove(name:string) +~HashMap()

- HashMap 객체는 입력되는 접속 파일 하나당 한 개씩 생성되게 된다. 현재 수업에 대한 정보와 날짜, StdLinkedList 클래스 배열 그리고 전체 수강생 수를 멤버 변수로 가진다. 자동 출결 처리 프로그램은 UI에서 학생 정보에 따라 출석률을 보여주고 전체 출석률을 확인하는 등 학생 정보를 탐색할 일이 잦아 O(1) 의 시간 복잡도를 가진 해시 테이블을 사용한다. findStudent 에는 해시 테이블의 키 값을 통해 탐색할 수 있게 해주는 함수이다. put을 통해 해쉬 함수에 값을 추가 get을 통해 값을 가져올 수 있다.

○ FileProcessor.h

FileProcessor
-GetTimeT(year : int, month : int, day : int. hour : int, minute : int, second : int, isPM : bool) : int -registStd(hashmap : HashMap *, fileName :char *) : bool -registStdAttendance(hashmap : HashMap *, filename : char *) : bool -saveStd(hashMap * hashmap, fileName : char *)

- 파일 프로세서는 파일을 읽고 이를 처리하기 쉬운 형태로 바꾸어 hashmap 에 저장해 주는 함수들을 가진 클래스이다.
registStd 는 특정 수업의 수강생에 대한 정보를 가진 파일을 처리하기 위한 함수이다.
registStdAttendance 는 수강생의 회의 총 참여 시간 이름 중복등을 처리해주는 함수이다. saveStd 는 파일을 저장하기 위한 함수이다.

○ Attendance.h

Attendance.h
-attendanceRate : int
-checkAttendance(hashmap : HashMap *, standard : int. fileName : char *) -outputAttendance(hashmap : HashMap *, fileName : char *)

- checkAttendance 함수는 참여자들의 최종 수업 참여 시간과 기준이 되는 수업 시작 시간을 기반으로 출석을 확인해주는 함수이다. 학생들 각각의 출석률을 입력해준다.
outputAttendance 는 입력된 파일의 각각의 수업의 출석률을 저장하여 입력해주는 함수이다.

2. 절차 설계

○ 해시 테이블 체이닝

- 해시 충돌을 막기 위해 shift 연산과 소수를 이용한다

```
int hash(string str){
      int hash = 401;
      while (*str != '\0') {
               hash = ((hash << 4) + (int)(*str)) % this.count;
               str++;
      }
      return hash % this.count;
}
```

○ 해시 테이블 값 가져오기

```
Info get(int StdCode) {
         int StdInfo;
         int hashcode = this->hash(StdCode);

         Student* cursor = this->container[hashcode]->head;
         cursor = this->_findStudentByStdCode(StdCode, cursor);

         // 해쉬코드 컨테이너에 키가 있다면
         if (cursor != nullptr) {
             // 값 리턴
             return cursor->StdInfo;
         }
         // 없다면
         else {
             cout << "키를 찾을 수 없습니다." << endl;
             return cursor->StdInfo;
         }
    }
```

○ 해시 테이블 값 추가하기

```
void put(int StdCode, Info StdInfo) {
         int hashcode = this->hash(StdCode);

         Student* cursor = this->container[hashcode]->head;
```

```
        cursor = this->_findStudentByStdCode(StdCode, cursor);

        // 해쉬코드 컨테이너에 키가 있다면
        if (cursor != nullptr) {
            // 덮어쓰기
            cursor->StdInfo = StdInfo;
        }
        // 없다면
        else {
            // 컨테이너에 노드추가
            this->container[hashcode]->add(StdCode, StdInfo);
        }
    }
```

○ 해시 테이블 값 가져오기

```
Info get(int StdCode) {
        int StdInfo;
        int hashcode = this->hash(StdCode);

        Student* cursor = this->container[hashcode]->head;
        cursor = this->_findStudentByStdCode(StdCode, cursor);

        // 해쉬코드 컨테이너에 키가 있다면
        if (cursor != nullptr) {
            // 값 리턴
            return cursor->StdInfo;
        }
        // 없다면
        else {
            cout << "키를 찾을 수 없습니다." << endl;
            return cursor->StdInfo;
        }
    }
```

○ 수강생 정보 등록

```
bool registStd(HashMap* hashmap, char* fileName) {
int count = 0;
int a = 0;
FILE* pFile = NULL
```

```
 pFile = fopen(fileName, "r");
 if (pFile) {
 char buffer[Max];
      while (!feof(pFile)) {
               fgets(buffer, sizeof(buffer), pFile);
               if (count != 0) {
                        char* ptr = strtok(buffer, ",");
                        a = atoi(ptr);
                        ptr = strtok(NULL, "\n");
                        if (ptr != NULL) {
                                 Info info = { ptr, "", false, 0,0, };
                                 hashmap->put(a, info);
                        }
               }
               count++;
               }
      }
      fclose(pFile);
      return NULL
}
```

○ 수강생 접속 기록

```
 bool registStdAttendance(HashMap* hashmap, char* fileName) {
 int count = 0;
 int a = 0;
 FILE* pFile = NULL
 pFile = fopen(fileName, "r");
 if (pFile) {
 char buffer[Max];
      while (!feof(pFile)) {
               fgets(buffer, sizeof(buffer), pFile);
               if (count != 0) {
                        /*
                                 쉼표를 기준으로 이름, 학번등을 분류해 info 에 입력
                        */
                                 //time stamp를 숫자로 바꾸어 수강 시간 처리
                                 //중복되는 이름에 대한 처리
                                 hashmap->put(a, info);
                        }
               }
```

```
                count++;
                }
        }
        fclose(pFile);
        return NULL
}
```

○ Time stamp를 숫자로 바꿈

```
int GetTimeT(int year, int month, int day, int hour,
        int minute, int second, bool isPM) {
                struct tm t = { 0 };
                t.tm_year = year - 1900;
                t.tm_mon = month - 1;
                t.tm_mday = day
                t.tm_hour = isPM ? 12 + hour : hour
                t.tm_min = minute
                t.tm_sec = second
                return mktime(&t);
        }
```

○ 파일 저장

수강생 전원에 대한 파일이 입력되었을 경우 그 수업에 대한 정보를 Attendance 클래스는 HashMap 배열을 갖게 된다. 하나의 과목당 여러개의 수업 접속 기록 파일이 들어올 수 있으므로 입력받은 과목들마다 배열을 하나씩 생성해 준다. 그리고 전체 과목을 담게될 배열안에 입력 받은 과목이 들어간 이차원 배열이 생성될 것이다. 기존에 존재하였던 과목에 해당하는 HashMap 클래스가 있는 경우 과목 배열에 추가해준다. 파일 저장에서는 배열을 파라미터로 받아 시간 순서대로 정렬한 뒤 저장해준다.과목당 존재하는 배열이 다 없어질 때 까지 이를 반복한다.

○ 여러번 존재하는 접속 기록에 대한 처리

접속기록은 이름을 기준으로 판단한다. 만약 수강생의 목록에서 동명이인이 존재할 경우에는 학번을 기준으로 분류한다. 하지만 접속 기록에서 학번을 입력하지 않은 경우가 존재할 수 있으므로 “김다은 (1) 김다은 (2)”와 같이 처리하여 한쪽이라도 학번을 입력하였을 경우에는 구분이 가능하도록 해준다. 동시 접속의 경우 제일 빨리 접속한 시간을 기준으로 결석 혹은 지각을 판단한다. 전체 수업 수강 시간은 접속한 시간으로부터 나간시간을 기준으로 만약 한쪽에서는 접속을 하였고 한쪽은 접속하지 않았다면 겹치는 시간을 빼주어 전체 수업 참여 시간을 계산하여 준다.

Thank you.

ADVENTURE DESIGN
ATTENDANCE ORGANIZER