

# 소프트웨어공학 2차 과제



소 속 : 공학 대학

과 목 명 : 소프트웨어공학

학 과 : AI소프트웨어학과

학 번 : 202320078

이 름 : 김세현

제 출 일 : 2024.10.21

# 목차

1. 요구분석 ----- 1p~2p
2. 설계 ----- 3p~7p
3. 구현코드 설명 -----8p~10p
4. 결과분석 ----- 11p~13p
5. 유지 보수 ----- 14p

## 비디오 대여 프로그램의 필수요소

비디오 대여 프로그램의 경우 가게에서 실질적으로 사용할 수 있는 프로그램을 만들기 위해 가능한 많은 기능을 구현해 보았습니다.

아래의 추가 기능은 비디오 추가, 삭제, 대여, 검색, 반납, 회원가입, 대여 잔여일 조회 와 같은 기본적인 기능을 제외한 추가적인 기능입니다.

### 추가기능

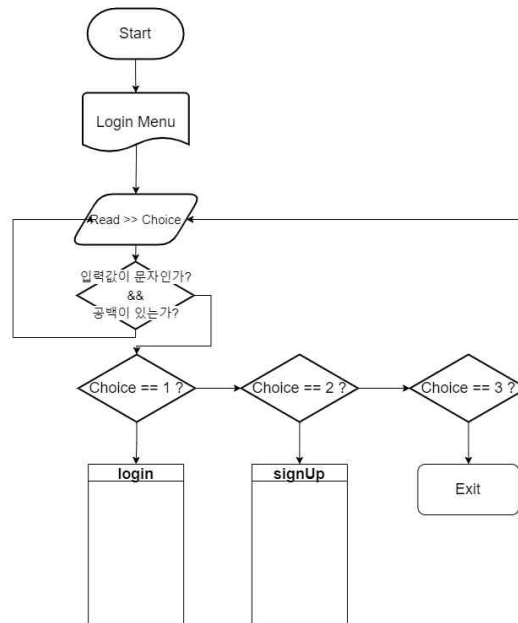
관리자 권한, 사용자 권한
사용자와 관리자의 권한을 다르게 적용하여 각각 다른 기능을 사용할수 있도록 하였습니다.
코인 시스템
사용자가 비디오를 대여할 때 일정한 금액이 과금되도록 하는 시스템입니다. 코인추가는 관리자만 할 수 있습니다.

연체 알림
<p>사용자가 비디오를 7일 이상 반납하지 않을 경우 연체자로 분류되어 로그인 시 주의 알림이 출력되고, 추가적인 대여를 하지 못하도록 하였습니다.</p>

관리자 추가
<p>관리자 추가의 경우 기존의 관리자가 일반사용자 중에서 선택하는 방식이며, 관리자 인원의 경우 제한을 두지 않았습니다.</p>

## 로직설명

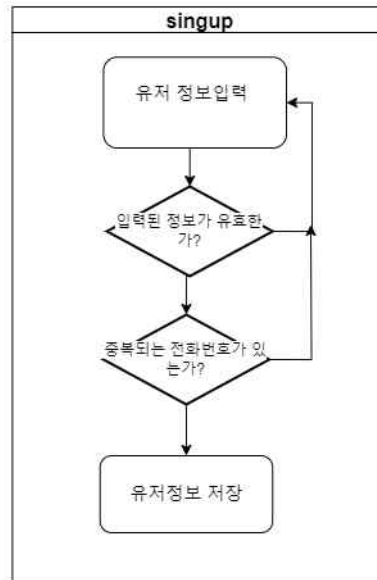
### 1. 초기화면



프로그램을 실행하면 가장 먼저 실행되는 로직입니다.

메뉴에서 1번을 입력하면 로그인으로 넘어가고, 2번을 입력하면 회원가입, 3번은 프로그램 종료로 구성되어 있습니다.

## 2. 회원가입



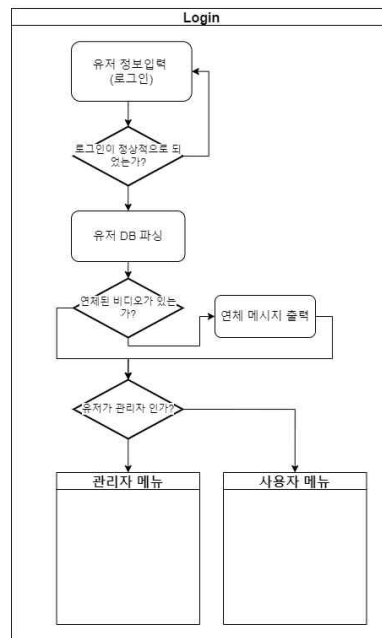
회원가입 에서는 사용자의 id, 비밀번호, 이름, 전화번호, 생년월일을 입력받습니다. 이때 ID와 전화번호가 기존의 데이터와 중복이 될 경우 회원가입이 되지않도록 만들었습니다.

### 2-1. 유저저장 방식

**about7086, 2004, 김세현, 2004-03-04, 010-4238-7086, 1, 0**  
 id, 비밀번호, 이름, 생년월일, 전화번호, 관리자여부, 코인 수  
 (1==관리자, 0==사용자)

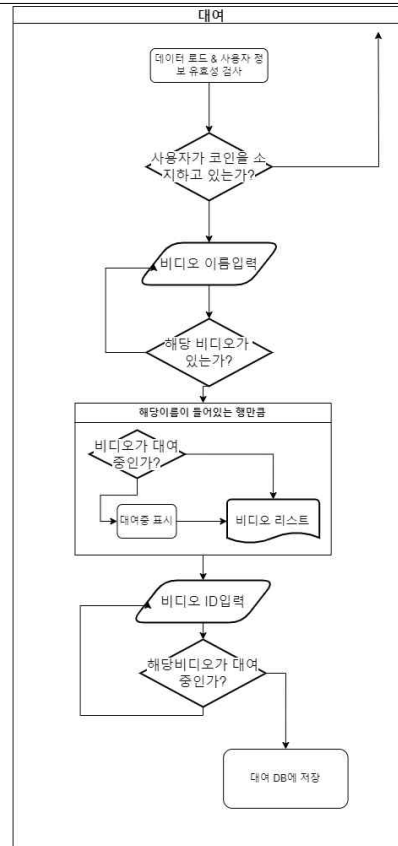
유저 db에 저장되는 방식입니다. 각각 입력받은 정보를 저장하고, 순서대로 관리자여부, 코인수가 입력됩니다. 회원가입을 한 사용자의 경우 코인이 없는 일반 사용자 임으로 마지막 두자리가 ~0,0 으로 저장됩니다.

### 3. 로그인



로그인의 경우 사동적으로 관리자와 사용자를 구분하여 각각의 메뉴로 분해하는 기능으로 만들었습니다. 권한을 나타내는 위치에 1이 있을 경우 관리자 메뉴로 넘어가며, 0일 경우에는 사용자 메뉴로 넘어갑니다.

## 4. 대여



대여함수가 실행되면 가장먼저 프로그램은 사용자의 데이터와 비디오 대여 데이터, 비디오 리스트를 수집합니다. 그후 사용자가 가지고있는 코인의 수가 1 이상일 경우에만 대여 함수가 정상적으로 작동되며, 사용자가 비디오 이름의 일부만 입력해도 유사도 에따라 검색 되도록 만들었습니다. 검색되는 비디오가 있을 경우 리스트가 출력되며, 사용자는 출력물을 보고 해당 비디오의 고유 id를 입력하여 대여할수 있습니다. 대여가 정상적으로되면, 대여db에 사용자의 id와 대여한 비디오의 id, 대여한 날짜가 자동으로 기입되어 저장됩니다.

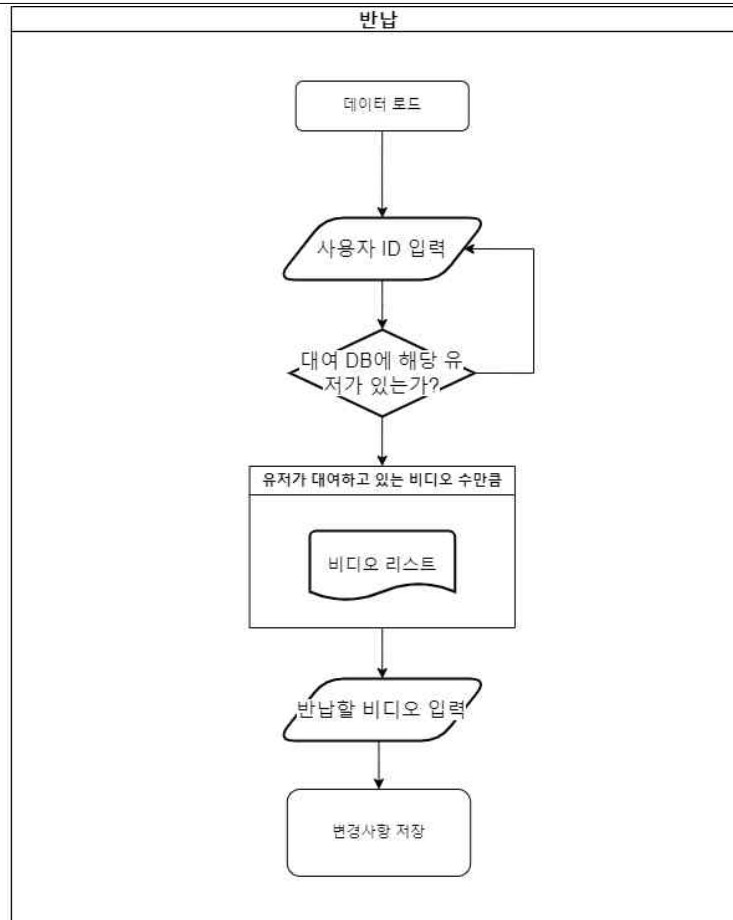
### 4-1. 대여저장 방식

about04,1,2024-10-20

사용자 ID, 비디오 ID, 대여 날짜



## 5. 반납



비디오 반납의 경우 관리자 권한만 사용할수 있는 기능입니다. 관리자가 반납을 하는 사용자의 id를 입력하면고, 해당id의 사용자가 대여db에 있을 경우 정상작동 합니다. 이후 해당 사용자가 대여하고 있는 비디오의 리스트가 출력되며 리스트를 보고 반납할 비디오를 선택하면 반납처리가 완료됩니다. 반납처리가 완료되면 대여db에서 해당 정보가 삭제됩니다.

## 구현 코드

### 데이터로드

#### 대여DB 로드

```
std::vector<std::tuple<std::string, std::string, std::time_t>> loadRentals() {
    std::vector<std::tuple<std::string, std::string, std::time_t>> rentals;
    std::ifstream infile("rentals.txt");
    std::string line;
    while (std::getline(infile, line)) {
        std::stringstream ss(line);
        std::string userId, videoId, dateStr;
        std::getline(ss, userId, ',');
        std::getline(ss, videoId, ',');
        std::getline(ss, dateStr, ',');

        std::tm tm = {};
        std::stringstream dateStream(dateStr);
        dateStream >> std::get_time(&tm, "%Y-%m-%d");
        std::time_t rentalDate = std::mktime(&tm);

        rentals.push_back(std::make_tuple(userId, videoId, rentalDate));
    }
    return rentals;
}
```

위의 코드는 세 개의 로드 코드중 한가지인 대여정보DB를 로드하는 코드입니다.  
DB상에서 , 로 칼럼을 구분하고있기 때문에 이를 구분자로 설정해주었고, stl의 벡터를 사용하여 주기억장치로 올리는 프로세스입니다.

## 회원가입 중복 & 유효성 검사

### 회원가입 함수의 일부분

```
std::regex phoneRegex("^(010-[0-9]{4})-[0-9]{4}|02-[0-9]{3,4}-[0-9]{4}|[0-6]{3}-[0-9]{3,4}-[0-9]{4}$"); // '-'가 포함된 전화번호의 정규 표현식
std::regex phoneRegexNoDash("^(010[0-9]{8}|02[0-9]{7,8}|[0-6]{3}[0-9]{7,8})$"); // '-'가 미포함된 전화번호의 정규 표현식
std::regex nameRegex("^[0-9]{4}$"); // 숫자가 포함되지 않은 이름의 정규 표현식
std::regex passwordRegex("^[a-zA000-907A9]+$"); // 한글이 포함되지 않은 비밀번호의 정규 표현식
std::regex birthDateRegex("^[0-9-]{4}$"); // 한글과 영어가 포함되지 않은 생년월일의 정규 표현식
```

```
std::vector<User> users = loadUsers();

std::cout << "회원가입을 진행합니다." << std::endl;
std::cout << "사용자 ID: ";
std::cin >> userId;
std::cin.ignore(); // 버퍼 비우기

if (!isUserIdUnique(userId, users)) {
    std::cout << "이미 존재하는 사용자 ID입니다, 다시 시도하십시오." << std::endl;
    return;
}

while (true) {
    std::cout << "비밀번호: ";
    std::getline(std::cin, password);

    if (!std::regex_match(password, passwordRegex)) { // 비밀번호에 한글이 포함된 경우
        std::cout << "비밀번호에 한글이 포함될 수 없습니다, 다시 입력하세요." << std::endl;
    }
    else {
        break;
    }
}
```

```
std::getline(std::cin, phoneNumber);

if (phoneNumber.empty()) {
    std::cout << "전화번호가 비어 있습니다, 다시 입력해 주세요." << std::endl;
}
else if (std::regex_match(phoneNumber, phoneRegexNoDash)) { // - 추가
    if (phoneNumber.substr(0, 2) == "02" && phoneNumber.length() == 9) { // 서울지역번호 2자리-3자리-4자리
        phoneNumber = phoneNumber.substr(0, 2) + "-" + phoneNumber.substr(2, 3) + "-" + phoneNumber.substr(5, 4);
    }
    else if (phoneNumber.substr(0, 2) == "02" && phoneNumber.length() == 10) { // 서울지역번호 2자리-4자리-4자리
        phoneNumber = phoneNumber.substr(0, 2) + "-" + phoneNumber.substr(2, 4) + "-" + phoneNumber.substr(6, 4);
    }
    else if ((phoneNumber.substr(0, 3) == "031" || phoneNumber.substr(0, 3) == "032" || phoneNumber.substr(0, 3) == "042" ||
        phoneNumber.substr(0, 3) == "051" || phoneNumber.substr(0, 3) == "052" || phoneNumber.substr(0, 3) == "053" ||
        phoneNumber.substr(0, 3) == "062" || phoneNumber.substr(0, 3) == "064" || phoneNumber.substr(0, 3) == "033" ||
        phoneNumber.substr(0, 3) == "041" || phoneNumber.substr(0, 3) == "043" || phoneNumber.substr(0, 3) == "054" ||
        phoneNumber.substr(0, 3) == "055" || phoneNumber.substr(0, 3) == "061" || phoneNumber.substr(0, 3) == "063") &&
        phoneNumber.length() == 10) { // 지역번호 3자리-3자리-4자리
        phoneNumber = phoneNumber.substr(0, 3) + "-" + phoneNumber.substr(3, 3) + "-" + phoneNumber.substr(6, 4);
    }
    else if ((phoneNumber.substr(0, 3) == "031" || phoneNumber.substr(0, 3) == "032" || phoneNumber.substr(0, 3) == "042" ||
        phoneNumber.substr(0, 3) == "051" || phoneNumber.substr(0, 3) == "052" || phoneNumber.substr(0, 3) == "053" ||
        phoneNumber.substr(0, 3) == "062" || phoneNumber.substr(0, 3) == "064" || phoneNumber.substr(0, 3) == "033" ||
        phoneNumber.substr(0, 3) == "041" || phoneNumber.substr(0, 3) == "043" || phoneNumber.substr(0, 3) == "054" ||
        phoneNumber.substr(0, 3) == "055" || phoneNumber.substr(0, 3) == "061" || phoneNumber.substr(0, 3) == "063") &&
        phoneNumber.length() == 11) { // 지역번호 3자리-4자리-4자리
        phoneNumber = phoneNumber.substr(0, 3) + "-" + phoneNumber.substr(3, 4) + "-" + phoneNumber.substr(7, 4);
    }
    else if (phoneNumber.substr(0, 3) == "010" && phoneNumber.length() == 11) {
        phoneNumber = phoneNumber.substr(0, 3) + "-" + phoneNumber.substr(3, 4) + "-" + phoneNumber.substr(7, 4); // 휴대전화번호 010-4자리-4자리
    }
    else {
        std::cout << "잘못된 전화번호 형식입니다, 다시 입력해 주세요." << std::endl;
        continue;
    }
    break;
}
else if (std::regex_match(phoneNumber, phoneRegex)) { // - 있는 경우 필수
    break;
}
else {
    std::cout << "잘못된 전화번호 형식입니다, 전화번호는 010-xxxx-xxxx 또는 010xxxxxx 이거나 지역전화의 경우 (지역번호)-(x)xxx-xxxx, (지역번호)(x)xxxxxx 형식이어야 합니다." << std::endl;
}

if (!isPhoneNumberUnique(phoneNumber, users)) {
    std::cout << "이미 존재하는 사용자 전화번호입니다, 다시 시도하십시오." << std::endl;
    return;
}
```

회원가입에서 사용자의 id와 전화번호는 중복되지 못하도록 만들었습니다. 뿐만 아니라 생년월일 입력에 문자(특수문자 제외)를 입력하거나, 이름에 숫자가 들어가는 일을 방지하기 위해 정규표현식을 만들어 유효성을 검사한후 저장하도록 만들었습니다. 전화번호의 경우 1차 과제에 코드를 인용하여 사용하였습니다.

## 연체 알림 메시지

### 연체판별 함수

```
bool hasOverdueVideos(const User& user, const std::vector<std::tuple<std::string, std::string, std::time_t>>& rentals) {
    std::time_t now = std::time(nullptr);
    const int rentalPeriod = 7 * 24 * 60 * 60;

    for (const auto& rental : rentals) {
        if (std::get<0>(rental) == user.userId) {
            std::time_t rentalDate = std::get<2>(rental);
            if (now - rentalDate > rentalPeriod) {
                return true;
            }
        }
    }

    return false;
}
```

### 알림 메시지

```
void login() {
    std::string userId, password;
    clearScreen();

    std::vector<User> users = loadUsers();

    std::cout << "로그인을 진행합니다." << std::endl;
    std::cout << "사용자 ID: ";
    std::cin >> userId;
    std::cout << "비밀번호: ";
    std::cin >> password;

    clearScreen();
    User* user = authenticateUser(userId, password, users);
    if (user) {
        currentUser = user;

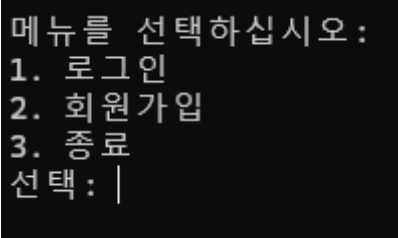
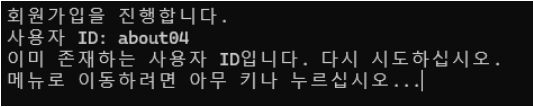
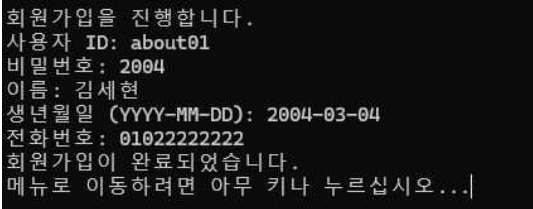
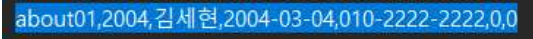
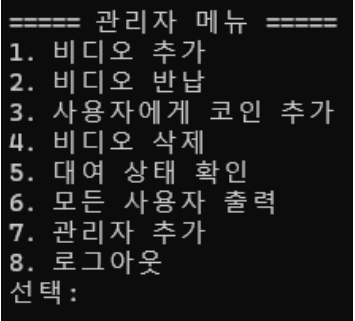
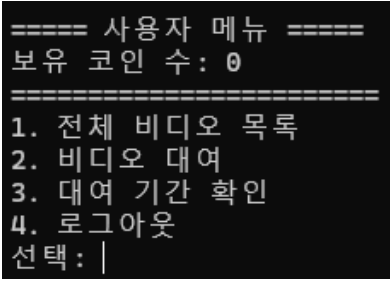
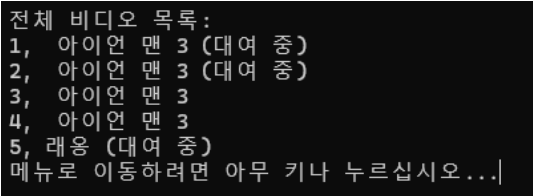
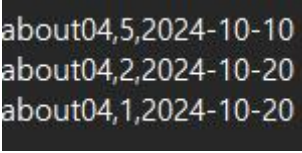
        // 대여 비디오 정보를 로드하여 연체 여부 확인
        std::vector<std::tuple<std::string, std::string, std::time_t>> rentals = loadRentals();
        if (hasOverdueVideos(*user, rentals)) {
            std::cout << "***경고*** !!연체된 비디오가 있습니다!! ***경고***" << std::endl;
            Sleep(4000);
        }

        if (user->isAdmin) {
            adminMenu();
        }
        else {
            userMenu();
        }
    }
    else {
        std::cout << "로그인 실패, 사용자 ID 또는 비밀번호가 잘못되었습니다." << std::endl;
    }
}
```

반납을 7일 이상 하지 않을 경우 로그인시 알림을 출력하는 기능입니다.

bool 함수를 이용하여 T/F로 연체여부를 판별하는 함수를 만들었으며, True일 경우 4초 간 메시지가 출력되도록 만들었습니다.

## 테스트 결과

<p>메뉴 UI</p> 	<p>&gt; 2번 - 회원가입 (중복된 id 입력)</p>  <p>&gt; enter 입력 후 다시 메뉴로 진입</p>
<p>&gt; 2번 - 회원가입 (정상적인 가입)</p>   <p>(유저정보 db 결과)</p> <p>&gt; enter 입력 후 다시 메뉴로 진입</p>	<p>&gt; 초기 메뉴에서 1번 - 관리자 로그인</p> 
<p>&gt; 초기 메뉴에서 1번 - 사용자 로그인</p>  <p>&gt; 수정하고 싶은 요소 선택</p>	<p>&gt; 사용자 메뉴 1번 - 비디오 전체 출력</p>  <p>(아래: 대여DB)</p>  <p>&gt; enter 입력 후 다시 메뉴로 진입</p>

<p>&gt; 관리자 메뉴 3번 - 코인 추가</p> <p>코인을 추가할 사용자 ID: about01          추가할 코인 수: 10          코인이 추가되었습니다.          메뉴로 이동하려면 아무 키나 누르십시오... </p> <p>about01,2004,김세현,2004-03-04,010-2222-2222,0,10</p> <p>코인이 10개로 수정되걸 볼수있다</p>	<p>&gt; 사용자 메뉴 2번 - 비디오 대여</p> <p>비디오 이름: 아이언 맨          대여 가능한 비디오 목록:          비디오 ID: 4, 이름: 아이언 맨 3, 감독: 세인 블랙          비디오 ID: 6, 이름: 존워, 감독: 채드          대여할 비디오 ID를 입력하세요: 4          비디오가 성공적으로 대여되었습니다.          메뉴로 이동하려면 아무 키나 누르십시오... </p> <p>about01,4,2024-10-22</p> <p>대여DB에 정보가 저장됨</p>
<p>&gt; 사용자 메뉴 3번 - 대여 기간 확인</p> <p>현재 대여 중인 비디오 목록 및 남은 일수:          비디오 ID: 3, 이름: 아이언 맨 3, 대여 날짜: 2024-10-21, 남은 일수: 7일          메뉴로 이동하려면 아무 키나 누르십시오... </p>	<p>&gt; 만약 연체된경우</p> <p>***경고*** !!연체된 비디오가 있습니다!! ***경고***</p> <p>알림이 출력되며</p> <p>===== 사용자 메뉴 =====          보유 코인 수: 519          =====</p> <p>1. 전체 비디오 목록          2. 비디오 대여          3. 대여 기간 확인          4. 로그아웃          선택: 2          연체 중인 비디오가 있어 대여할 수 없습니다.          메뉴로 이동하려면 아무 키나 누르십시오... </p> <p>대여가 불가능함</p>
<p>▶ 관리자 메뉴 1번 - 비디오 추가</p> <p>비디오 이름: 존워          감독 이름: 채드          비디오 추가가 완료되었습니다.          메뉴로 이동하려면 아무 키나 누르십시오... </p> <p>1, 아이언 맨 3,세인 블랙          2, 아이언 맨 3,세인 블랙          3, 아이언 맨 3,세인 블랙          4, 아이언 맨 3, 세인 블랙          5,레옹, 튀크 베송          6,존워,채드</p> <p>(비디오 DB)비디오가 추가된걸 볼수있다</p>	<p>&gt; 관리자 메뉴 2번 - 반납</p> <p>비디오를 반납할 유저 ID: about01          현재 대여 중인 비디오 목록:          비디오 ID: 3, 이름: 아이언 맨 3, 대여 날짜: 2024-10-21          반납할 비디오 ID: 3          비디오가 성공적으로 반납되었습니다.          메뉴로 이동하려면 아무 키나 누르십시오... </p> <p>about04,5,2024-10-10          about04,2,2024-10-20          about04,1,2024-10-20           </p> <p>대여DB에서 삭제된걸 볼 수 있다.</p>

<p>&gt; 관리자 메뉴 4번 - 비디오 삭제</p> <div> 비디오 삭제를 진행합니다.  비디오 이름: 존워  삭제할 비디오를 선택하십시오  -----  1. 존워  선택: 1  비디오가 삭제되었습니다.  메뉴로 이동하려면 아무 키나 누르십시오...  </div> <div> 1, 아이언 맨 3,세인 블랙  2, 아이언 맨 3,세인 블랙  3, 아이언 맨 3,세인 블랙  4, 아이언 맨 3, 세인 블랙  5,레옹, 뤼크 베송 </div> <p>(비디오 DB)</p>	<p>&gt; 관리자 메뉴 5번 - 대여중인 비디오 리스트</p> <div> 원래 대여 중인 비디오 목록 및 남은 일수:  유저 ID: about04, 비디오 ID: 5, 이름: 레옹, 감독: 뤼크 베송, 대여 날짜: 2024-10-10, 남은 일수: 5일  유저 ID: about04, 비디오 ID: 2, 이름: 아이언 맨 3, 감독: 세인 블랙, 대여 날짜: 2024-10-20, 남은 일수: 5일  유저 ID: about01, 비디오 ID: 1, 이름: 아이언 맨 3, 감독: 세인 블랙, 대여 날짜: 2024-10-20, 남은 일수: 5일  유저 ID: about05, 비디오 ID: 3, 이름: 아이언 맨 3, 감독: 세인 블랙, 대여 날짜: 2024-10-22, 남은 일수: 7일  유저 ID: about01, 비디오 ID: 4, 이름: 아이언 맨 3, 감독: 세인 블랙, 대여 날짜: 2024-10-22, 남은 일수: 7일  메뉴로 이동하려면 아무 키나 누르십시오...  </div> <p>모든 사용자의 정보가 출력됨</p>
<p>&gt; 관리자 메뉴 6번 - 사용자 정보</p> <div> ===== 모든 사용자 목록 =====  ID: about7086  이름: 김세현  생년월일: 2004-03-04  전화번호: 010-4238-7086  관리자 여부: 예  코인: 0  -----  ID: about04  이름: 123  생년월일: 123  전화번호: 123  관리자 여부: 아니오  코인: 519  -----  ID: about05  이름: 김세현  생년월일: 2004-03-04  전화번호: 010-4234-7086  관리자 여부: 아니오  코인: 0  -----  ID: about01  이름: 김세현  생년월일: 2004-03-04  전화번호: 010-2222-2222  관리자 여부: 아니오  코인: 9  -----  메뉴로 이동하려면 아무 키나 누르십시오...  </div>	<p>&gt; 관리자 메뉴 7번 - 관리자 추가</p> <div> 관리자 추가를 진행합니다.  추가할 관리자 ID: about01  관리자가 추가되었습니다.  메뉴로 이동하려면 아무 키나 누르십시오...  </div> <div> about01,2004,김세현,2004-03-04,010-2222-2222,1,9 </div> <p>유저DB에서 권한이 관리자로 수정된걸 볼 수 있다.</p>



## 예외 처리문

```
회원이입을 진행합니다.  
사용자 ID: about02  
비밀번호 : 2004  
이름 : 123  
이름에 숫자가 포함될 수 없습니다. 다시 입력하세요.  
이름 : |
```

1. 이름 유효성 확인 - 이름의 경우 숫자가 들어가는 경우가 없기 때문에  
숫자가 들어갈 경우 예외처리 시킴

```
생년월일 (YYYY-MM-DD): ㄹㅇㅇ  
생년월일에 한글과 영어가 포함될 수 없습니다. 다시 입력하세요.  
생년월일 (YYYY-MM-DD): |
```

2. 생년월일 유효성 확인 - 생년월일 의 경우 문자가 들어갈 이유가 없기  
때문에 예외처리 시킴

```
전화번호: 01022  
잘못된 전화번호 형식입니다. 전화번호는 010-xxxx-xxxx 또는 010xxxxxxxxx 이거나 지역전화의 경우 (지역번호)-(x)xxx-xxxx, (지  
역번호)(x)xxxxxxx 형식이어야 합니다.  
전화번호: |
```

3. 전화번호 유효성 확인 - 실재 존재하는 전화번호인지 파악하고 불충족시  
예외처리시킴

```
if (std::cin.fail()) { // 문자가 들어올경우 실행(정수형입력에 실패한 경우)  
    std::cin.clear();  
    std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n'); // 메뉴에서 문자 입력 무시  
    continue;  
}
```

4. 메뉴에서 문자입력 예외처리