프로젝트

최종보고서

19013139 최명수

담당 교수님 : 신동일 교수님

과목 : 데이터베이스 2분반

목차

1. 기능적 요구사항 분석

<관리자 기능>

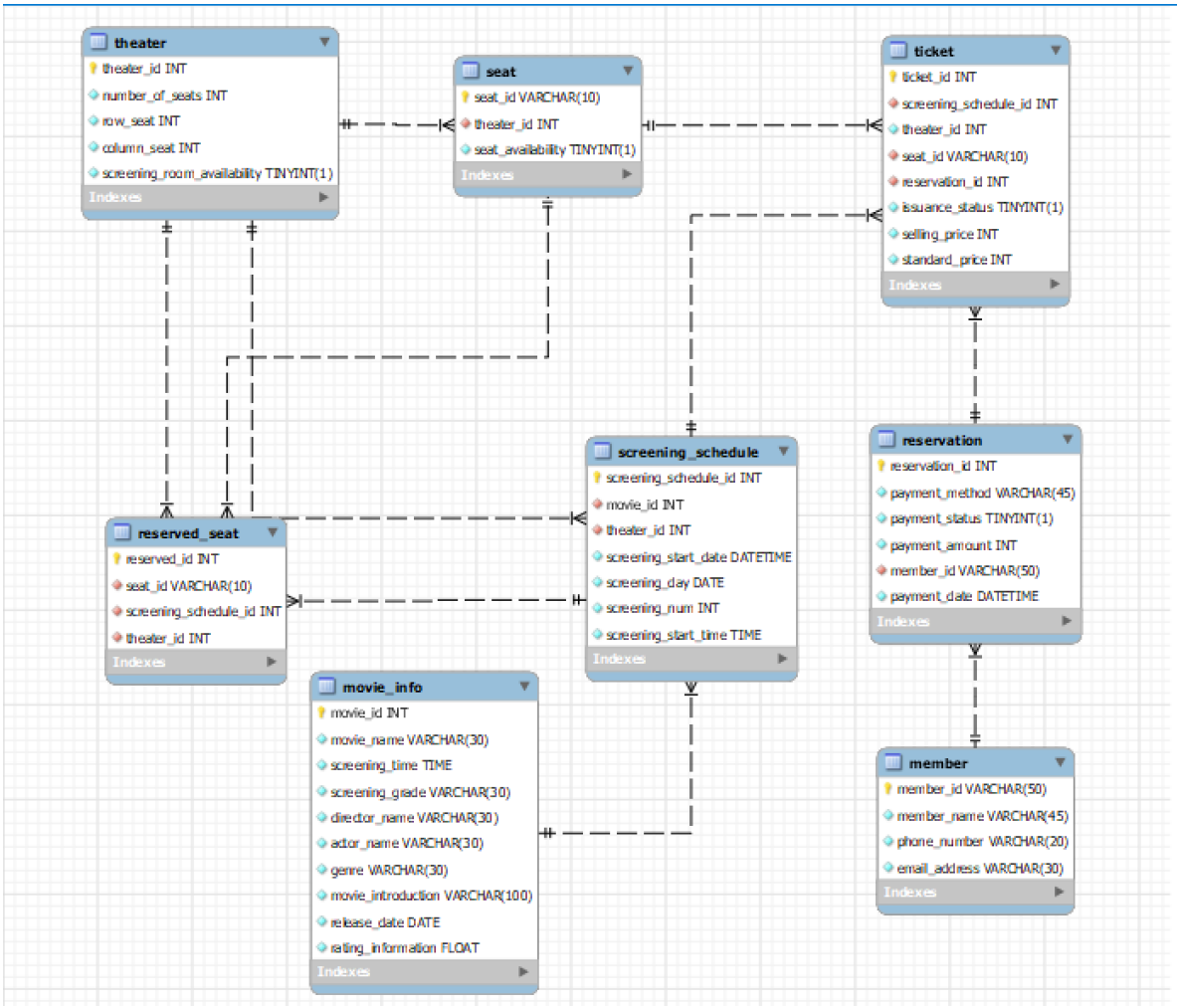
- 데이터베이스를 초기화할 수 있는 기능을 제공한다. 초기화 시에는 샘플 데이터로 초기화 된다.
- 데이터베이스에 포함된 모든 테이블에 대한 입력/삭제/변경 기능을 제공한다.

  단, 삭제/변경은 "조건식" 을 입력 받아서 삭제/변경하는 방식으로 구현한다. 입력 기능의 경우 반드시 하나의 윈도우 안에서 모든 데이터를 기입하고 버튼 클릭 한번으로 입력되도록 구현한다. 예) 입력해야할 속성이 4개라면, 속성1/속성2/속성3/속성4에 대한 입력값을 하나의 GUI 윈도우에서 입력받아야 하며, 저장/취소 버튼이 있어서 저장(즉, 입력 실행) 혹은 취소(입력 취소) 기능이 구현되어야 한다.
- 전체 테이블을 볼 수 있는 기능을 제공한다.

<회원 기능>

- 모든 영화에 대한 조회 기능을 제공한다. 영화명, 감독명, 배우명, 장르의 검색을 이용해 조회하고 입력되지 않은 정보는 무시하고 조회한다.
- 위에서 조회한 영화에 대한 예매 기능을 제공한다. 영화 예매 기능에서는 상영관 좌석을 반드시 GUI 로 표시하고 해당 좌석을 마우스로 선택할 수 있는 기능을 구현해야 한다.
- 본인이 예매한 영화에 대해서 영화명, 상영일, 상영관번호, 좌석번호 및 판매가격 정보를 보여주는 기능을 제공한다.
- 위에서 표시된 예매 정보 중에서 하나를 클릭하면 해당 예매에 대해서 모든 상영일정, 상영관, 티켓에 관한 정보를 보여주는 기능을 제공한다.
- 본인이 예매한 영화에 대하여 조회하여 다른 영화로 예매를 변경하는 기능을 제공한다.
- 본인이 예매한 영화에 대하여 조회하여 다른 상영 일정으로 변경하는 기능을 제공한다.

2. 스키마 설계



테이블 간의 관계 요약

1. 회원(member) 와 예약(reservation)
   - "reservation.member_id"는 "member.member_id" 를 참조한다.
   - 한 회원은 여러 예약을 가질 수 있다.
2. 영화(movie_info) 와 상영 일정(screening_schedule)
   - "screening_schedule.movie_id" 는 "movie_info.movie_id" 를 참조한다.
   - 한 영화는 여러 상영 일정을 가질 수 있다.

3. 상영관(theater) 와 상영 일정(screening_schedule)
    - "screening_schedule.theater_id" 는 "theater.theater_id" 를 참조한다.
    - 한 상영관은 여러 상영 일정을 가질 수 있다.
4. 상영관(theater) 와 좌석(seat)
    - "seat.theater_id" 는 "theater.theater_id" 를 참조한다.
    - 한 상영관은 여러 좌석을 가질 수 있다.
5. 좌석(seat)와 예약된 좌석(reserved_seat)
    - "reserved_seat.seat_id" 는 "seat.seat_id" 를 참조한다.
    - 한 좌석은 여러 예약된 좌석 기록을 가질 수 있다.
6. 상영 일정(screening_schedule) 와 예약된 좌석(reserved_seat)
    - "reserved_seat.screening_schedule_id" 는 "screening_schedule.screening_schedule_id" 를 참조한다.
    - 한 상영 일정은 여러 예약된 좌석 기록을 가질 수 있다.
7. 상영관(theater)와 예약된 좌석(reserved_seat)
    - "reserved_seat.theater_id" 는 "theater.theater_id" 를 참조한다.
    - 한 상영관은 여러 예약된 좌석 기록을 가질 수 있다.
8. 예약(reservation) 와 티켓(ticket)
    - "ticket.reservation_id" 는 "reservation.reservation_id" 를 참조한다.
    - 한 예약은 여러 티켓을 가질 수 있다.
9. 상영 일정(screening_schedule)와 티켓(ticket)
    - "ticket.screening_schedule_id" 는 "screening_schedule.screening_schedule_id" 를 참조한다.
    - 한 상영 일정은 여러 티켓을 가질 수 있다.
10. 좌석(seat)와 티켓(ticket)
    - "ticket.seat_id" 는 "seat.seat_id" 를 참조한다.
    - 한 좌석은 여러 티켓을 가질 수 있다.

3. 프로그램 구조

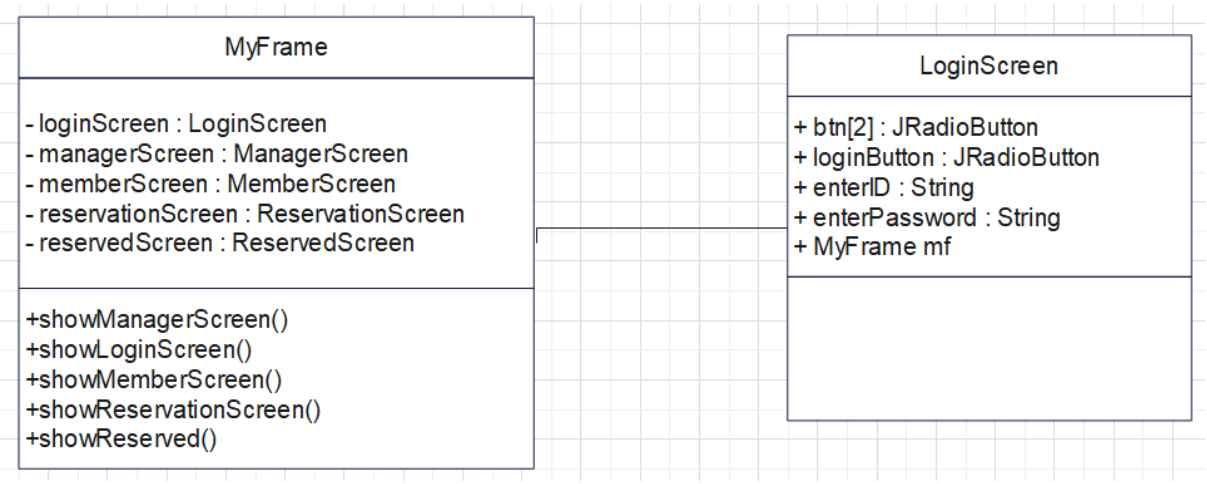
3 – 1. DBMS, 개발언어 및 개발도구

DBMS: MySQL 8.0.36

개발언어: JAVA (JDK 17)

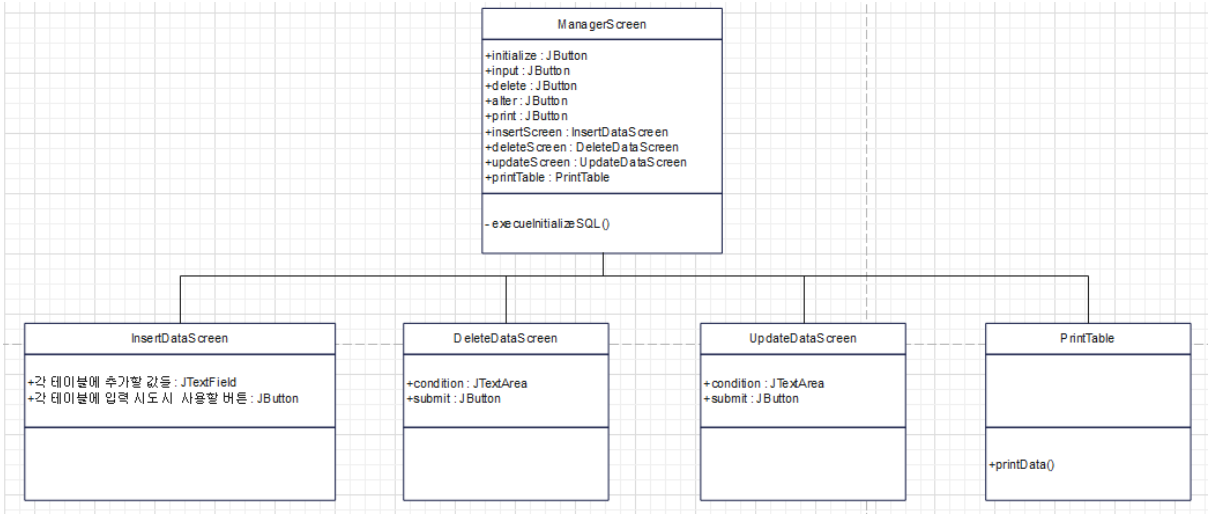
개발도구: Eclipse

3 – 2. Class 소개 (설명에 필요한 변수와 함수만을 표시)



LoginScreen 클래스에서 JRadioButton 타입의 btn 변수 2개로 관리자로 로그인 할 것인 것, 일반 회원으로 로그인할 것인지를 선택한다.

enterID 와 enterPassword 변수에 사용자가 입력한 아이디와 비번을 저장하고 이를 member 테이블의 member_id 열 값들(본 프로젝트에서는 아이디와 비번이 똑같도록 설정)과 비교하여 member 테이블에 존재하는 값들일 경우에 관리자일 경우에 MyFrame 클래스의 showManagerScreen 함수를 호출하여 관리자 화면을 띄운다. 일반 회원일 경우에는 showMemberScreen() 함수를 호출하여 일반 회원의 화면을 띄운다. member 테이블에 존재하지 않는 입력 값일 경우 "아이디나 비밀번호 정보가 맞지 않습니다." 라는 알림 창을 띄운다.



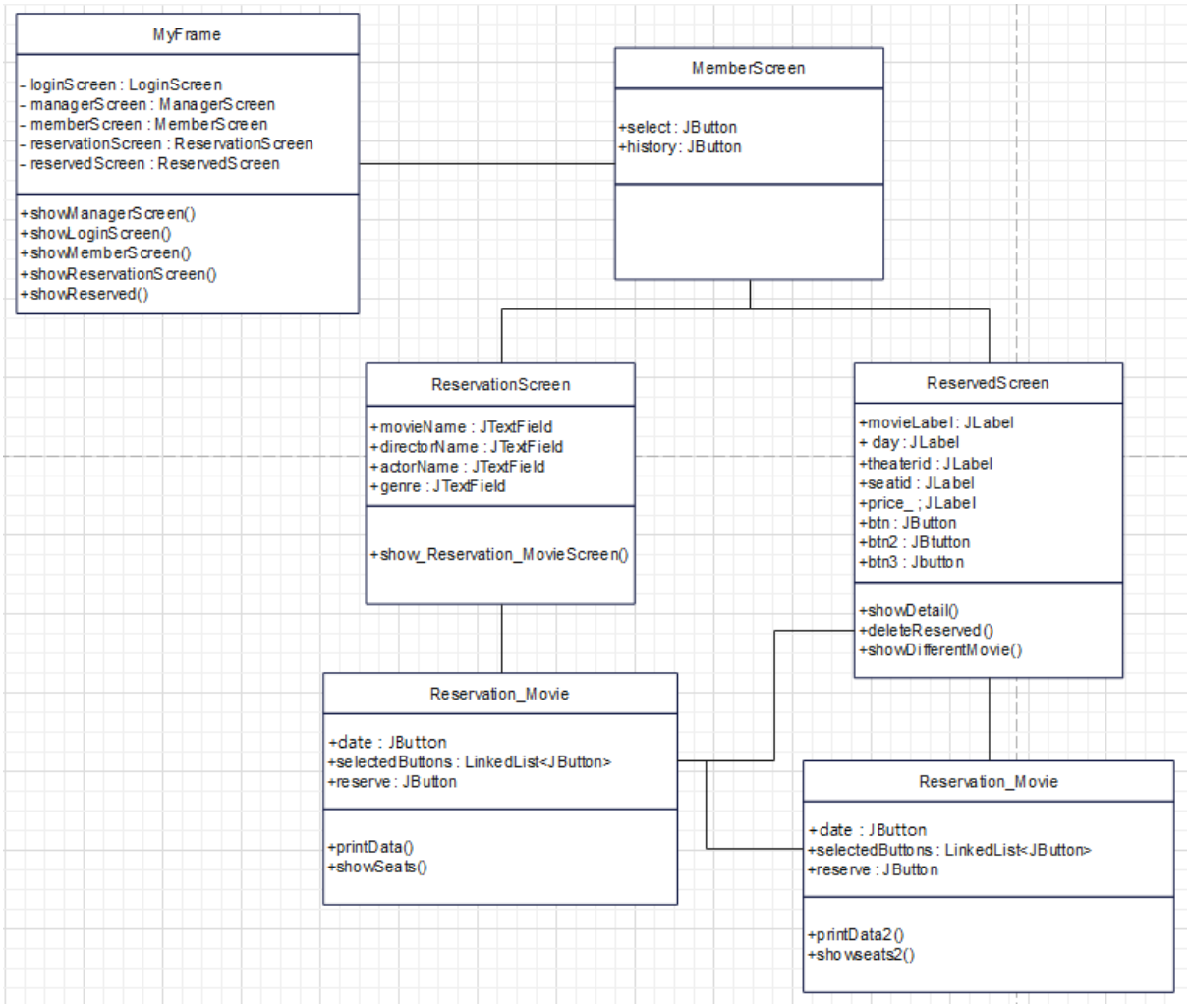
ManagerScreen 에서는 initialize 버튼을 누르면 executeInitializeSQL 을 통해서 초기화 기능이 구현되도록 하였고. Input, delete, alter, print 버튼을 누르면 InsertDataScreen, DeleteDataScreen, UpdateDataScreen, printTable 클래스가 화면에 표시되도록 하였다.

InsertDataScreen 에서는 총 8개의 테이블 각각에 데이터를 삽입할 수 있도록 하였고 삽입 취소 기능을 구현하기 위해 삽입 버튼을 누르면 "데이터를 삽입하시겠습니까?" 라는 문구와 함께 확인 창이 뜨도록 구현하였다.

DeleteDataScreen 에서는 JTextArea 타입의 condition 변수에 조건식을 입력 받게 하였고 JButton 타입의 submit 버튼을 누르면 조건식이 데이터베이스에 전달되도록 구현하였다. 해당 SQL 조건식이 외래키 제약 조건 등 오류 메시지를 띄우도록 하였고, "그래도 실행하시겠습니까" 라는 문구와 함께 창을 띄우도록 구현하였다.

UpdateDataScreen 은 DeleteDataScreen 과 같은 구현 방식으로 구현하였다.

PrintTable 에서는 printData() 함수를 overloading 하여 각 테이블의 모든 값들을 볼 수 있도록 구현하였다.



MemberScreen 에서 JButton 타입의 select 버튼을 클릭하면 MyFrame의 showReservationScreen 함수를 이용하여 영화를 조회할 수 있는 ReservationScreen 클래스를 화면에 띄운다. history 버튼을 클릭하면 showReserved() 함수를 이용하여 영화 예매 내역을 볼 수 있는 ReservedScreen 클래스를 화면에 띄운다.

ReservationScreen 클래스에서는 영화명(movieName), 감독명(directorName), 배우명(actorName), 장르(genre) 를 검색하여 영화를 조회한다. 영화를 예매하러 가는 버튼을 누르면 show_Reservation_MovieScreen() 함수를 호출하여 Reservation_Movie 클래스를 화면에 띄운다.

Reservation_Movie 클래스에서는 printData() 함수를 이용하여 JButton 타입의 date 변수로 해당 영화의 상영 날짜들이 나타나고, 그 중에 하나를 클릭하면 showSeats() 함수가 이용되어 해당 상영관의 이름과 좌석 배치도가 화면에 표시된다. 이미 예약된 좌석들은 비활성화된 상태로 회색 배경색을 가진 체로 나타난다. 예매하고자 하는 좌석들을 선택하면 빨간색 배경색으로 변경되고 LinkedList<JButton> 타입의 변수 selectedButtons 변수 안에 저장된다. JButton 타입의 submit 버튼을 클릭하면 해당 정보들이 reserved_seat,

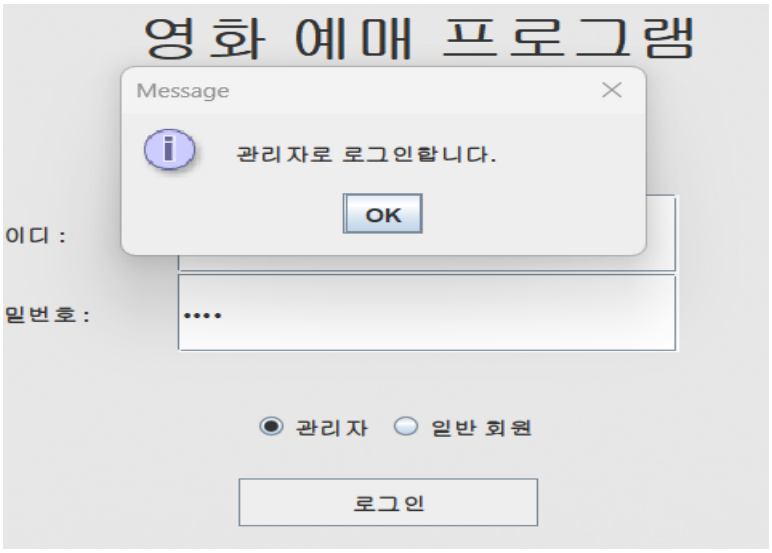
ticket, reservation 테이블에 각각 저장된다.

ReservedScreen 클래스에서는 예매된 정보들이 표시되는데, 영화이름(movieLabel), 상영날짜(day), 상영관번호(theaterid), 좌석 번호(seated), 가격(price) 정보들이 각각 표시된다. 본 프로젝트에서는 영화 한 표의 가격을 16000으로 지정하였다. 상세보기(btn), 예매 취소(bnt2), 예매 변경(btn3) 버튼 들을 클릭하면 showDetail(), deleteReserved(), showDifferentMovie() 함수가 이용되어 해당 기능에 맞는 화면들이 표시된다. 그 중 예매를 변경하는 기능은 영화를 예매할 때와 같은 화면이 표시되고 예매할 영화를 클릭하면 printData2() 함수가 이용되어 해당 영화의 상영날짜들이 표시된다. 예매할 상영날짜를 클릭하면 showseats2() 함수가 이용되어 해당 상영관과 좌석 배치도가 화면에 표시된다. 예약했던 좌석은 파란색 배경으로 표시되고 예매 변경할 좌석을 클릭하면 빨간색으로 표시된다. JButton 타입의 select 버튼을 클릭하면 변경된 좌석이 파란색 배경으로 바뀌고 예매했던 좌석은 빈 좌석으로 다시 표시된다.

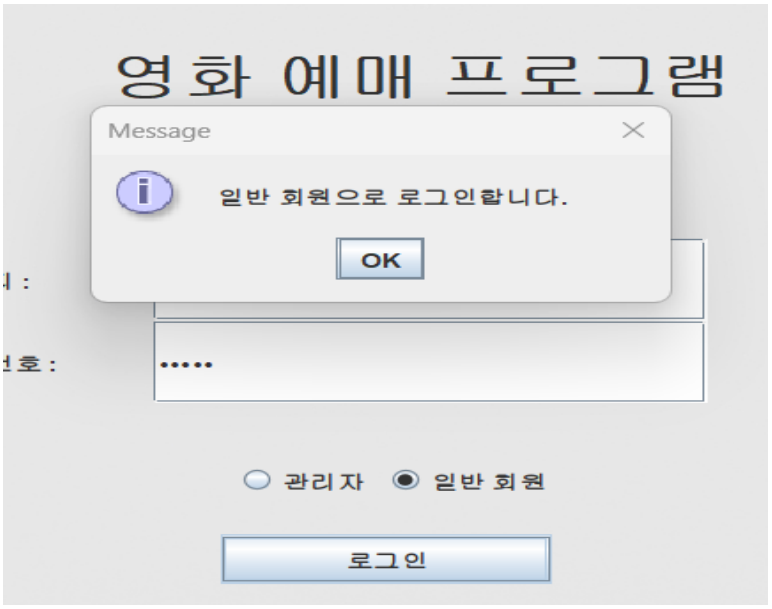
4. 테스팅 방법 및 GUI 구조와 DB 구조

1 – 1. 로그인 시 테스팅 방법 및 GUI 구조

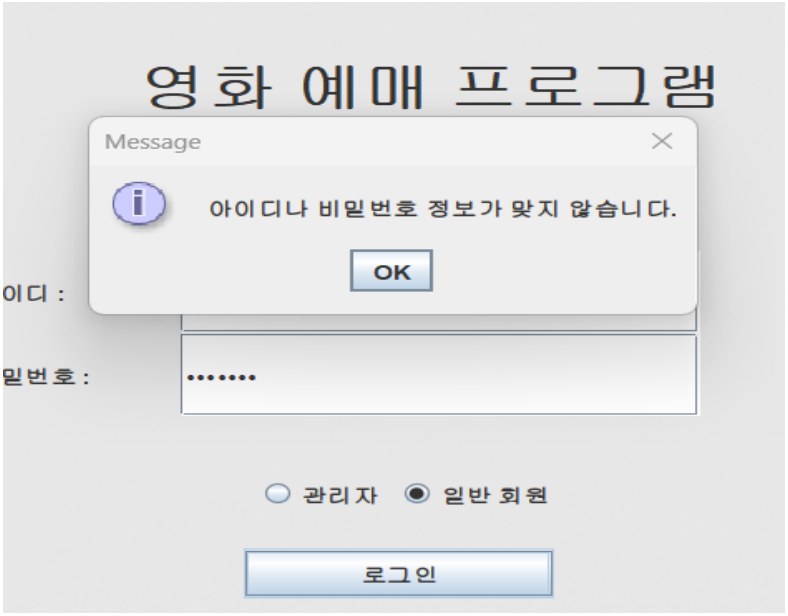
다음은 로그인 시에 화면이다. 관리자로 로그인 할 경우 root / 1234 (아이디 / 비밀번호) 를 입력하고, 일반 회원으로 로그인 할 경우 user1 / user1(아이디 / 비밀번호) 를 입력하여 로그인한다.



관리자로 로그인할 경우 다음과 같은 메시지와 함께 OK 버튼을 누르면 로그인이 가능하다.



일반 회원으로 로그인 할 경우에도 다음과 같은 메시지와 함께 OK 버튼을 누르면 로그인이 가능하다.



존재하지 않는 아이디와 비밀번호 일 경우 다음과 같은 메시지를 띄운다.

1 – 2. 로그인 시 DB 구조

```
16 •   select * from member;
```

Result Grid | Filter Rows: | Edit:

member_id	member_name	phone_number	email_address
user1	최명수	010-9770-6915	audtn0099@naver.com
user10	박수용	010-1453-1236	qweqwe@gmail.com
user11	양강현	010-5556-4534	fdgdfg@gmail.com
user12	김지원	010-4356-8964	asd@gmail.com
user2	김지원	010-1234-5678	zywon@gmail.com
user3	장경준	010-1231-3565	abcde@gmail.com
user4	김보미	010-6787-3454	sdfasd@gmail.com
user5	최나영	010-4145-5676	dfgddf@gmail.com
user6	최지영	010-4564-2345	asfsdf@gmail.com
user7	최정득	010-5676-0898	asdggd@gmail.com
user8	최인종	010-3454-2342	sdfsd@gmail.com
user9	김지원	010-5678-2222	erwee@gmail.com
NULL	NULL	NULL	NULL

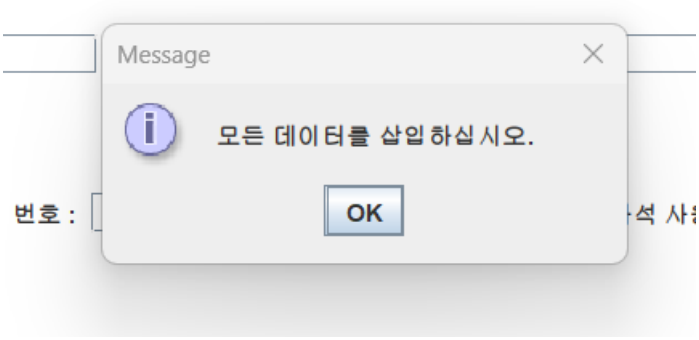
관리자 모드로 로그인을 시도할 경우에는 root / 1234 (아이디 / 비밀번호) 로만 로그인이 가능하게 코드를 구성하였고, 일반 회원으로 로그인을 시도할 경우에는 member 테이블 안에 존재하는 회원만 로그인이 가능하게 구현하였다.

2 – 1. 관리자 모드 시 테스팅 방법 및 GUI 구조

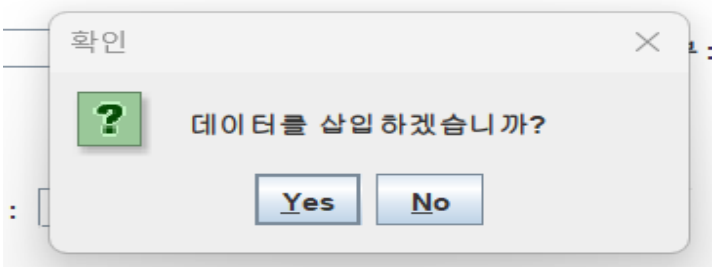
초기화 버튼을 누르면 내부적으로 초기화를 진행한다. 모든 테이블을 drop 하고 새로 만든 후에 초기화 데이터들을 삽입시킨다.

<데이터 삽입>

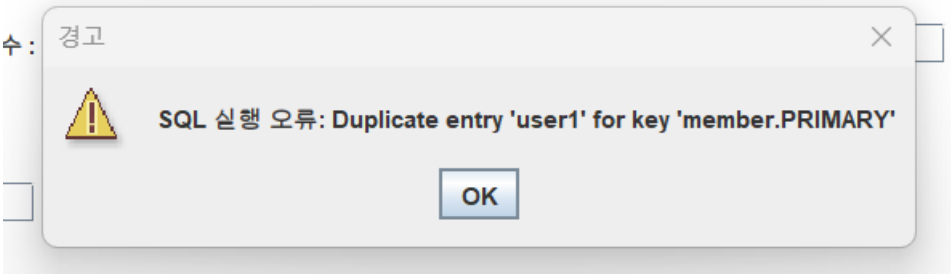
삽입 버튼을 클릭하였을 경우의 화면이다.



모든 속성들을 not null 로 구성하였기 때문에 하나의 속성이라도 입력되지 않으면 "모든 데이터를 삽입하십시오." 라는 문구와 함께 메시지창을 띄운다.

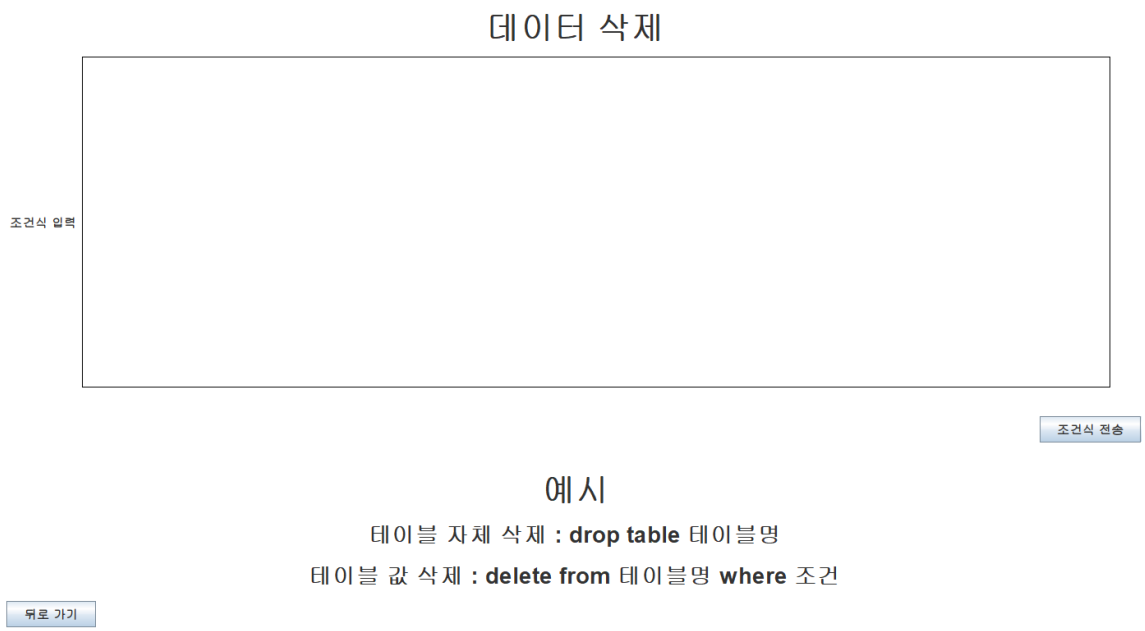


모든 속성들을 기입 한 후에 삽입 버튼을 클릭하면 다음과 같은 메시지를 띄우며 최종 의사를 물어본다.

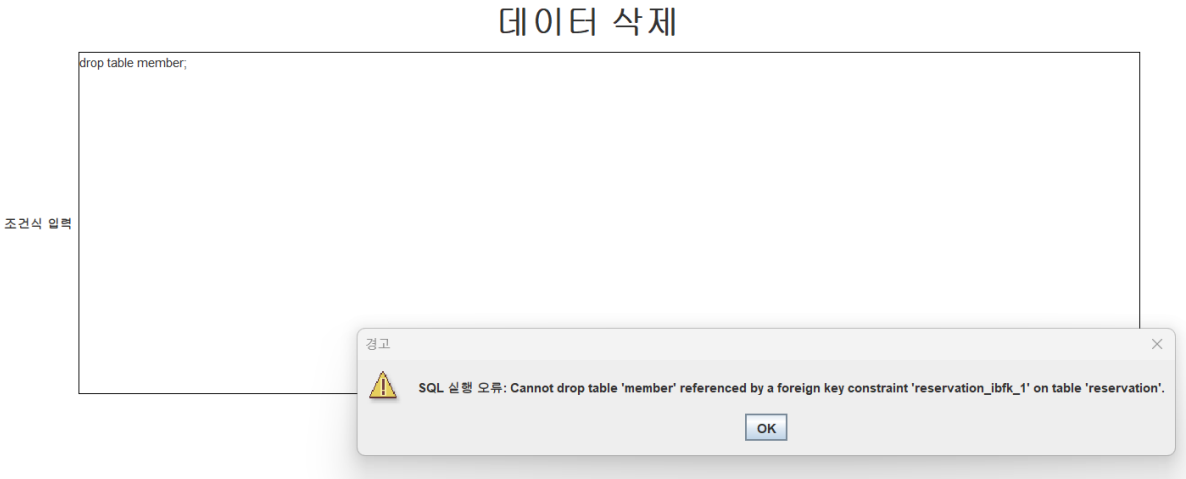


만약 삽입하려는 데이터가 SQL 실행 오류를 발생시키면 SQL 실행 오류의 원인과 함께 경고창을 띄운다.

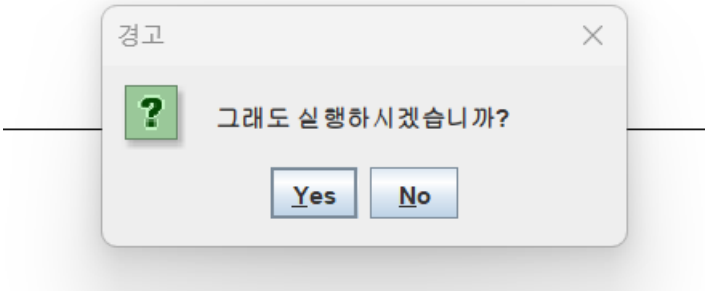
<데이터 삭제>



삭제 버튼을 클릭하였을 때의 화면이다.

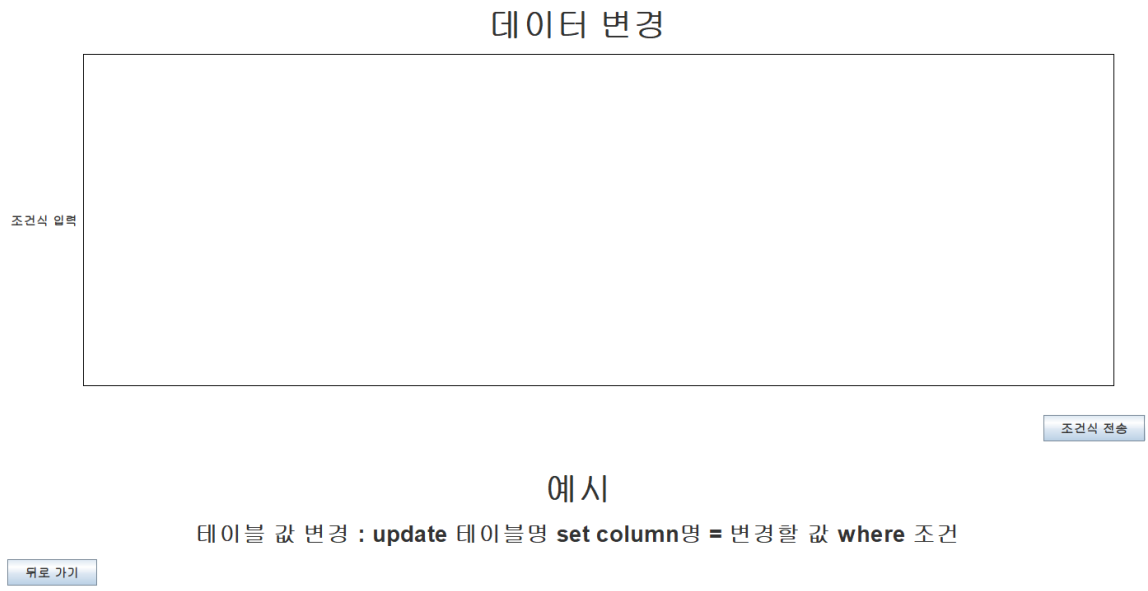


위와 같이 **drop table member;** 의 SQL 문을 실행할 경우 외래키 제약 조건으로 SQL 실행 오류 경고창을 띄운다.



하지만 관리자 판단 하에 테이블이나 데이터를 삭제 가능하도록 하기 위해 "그래도 실행하시겠습니까?" 라는 문구와 함께 질문창을 띄운다. Yes 버튼을 누를 경우에 코드 상에서 외래키 제약 조건을 검사를 하지 않도록 하고 삭제가 끝날 경우 외래키 제약 조건을 다시 검사하도록 변경한다.

<데이터 변경>



변경 버튼을 클릭하였을 때의 화면이다.

삭제에서의 기능과 똑같은 방식으로 구현되므로 설명은 생략한다.

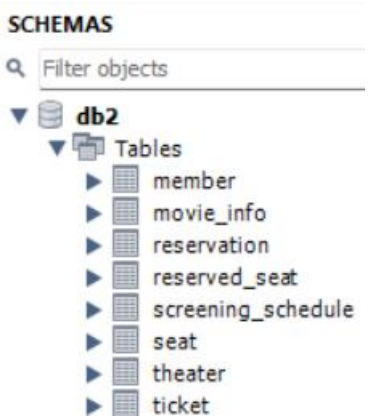
다음은 전체 테이블 보기 버튼을 클릭하였을 때의 화면이다.

데이터가 화면에 벗어날 경우를 대비하여 전부 scroll 이 가능하게 구현하였다.

왼쪽 하단을 보면 다음에 나타날 테이블의 이름이 버튼으로 되어 있다.

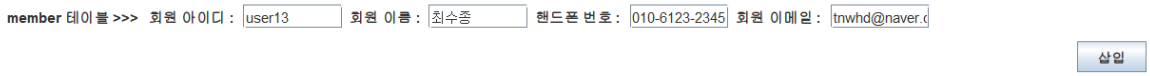
2 – 2. 관리자 모드 시 DB 구조

<초기화>



초기화의 경우 db2 데이터베이스에 모든 테이블이 잘 만들어진 것을 볼 수 있다.

<데이터 삽입>



('user13' , '최수종' , '010-6123-2345' , 'tnwhd@naver.com') 의 데이터를 member 테이블에 삽입하면,

user12	김지원	010-4356-8964	asd@gmail.com
user13	최수종	010-6123-2345	tnwhd@naver.com
user2	김지원	010-1234-5678	zywon@gmail.com

해당 데이터가 member 테이블에 잘 추가된 것을 볼 수 있다.

<데이터 삭제>

delete from member where member_id='user13'; 의 조건식을 입력 후에 조건식 전송 버튼을 누르면

user12	김지원	010-4356-8964	asd@gmail.com
user2	김지원	010-1234-5678	zywon@gmail.com

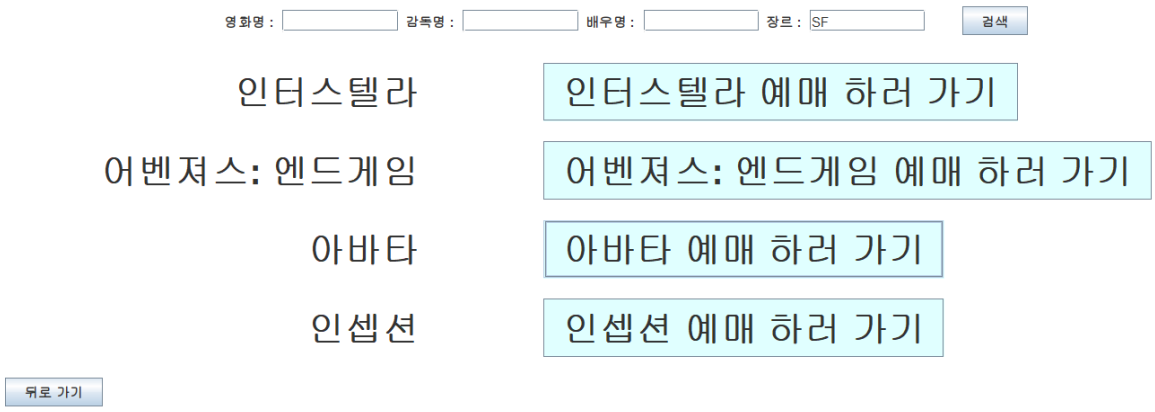
member 테이블에 해당 데이터가 잘 삭제 된 것을 볼 수 있다.

데이터 변경 시에는 이와 같은 방식으로 구현되므로 설명은 생략한다.

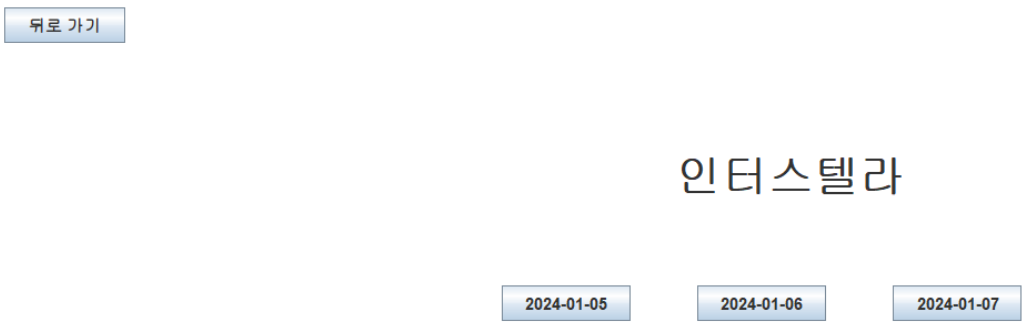
3 – 1. 영화 예매 시 테스팅 방법 및 GUI 구조

일반 회원으로 로그인 하고 영화 조회 버튼을 클릭하면 다음과 같은 화면이 표시된다.

만약 장르가 SF 인 영화들을 조회하고 싶다면, 장르의 검색 칸에 SF를 입력 후 검색 버튼을 클릭한다.

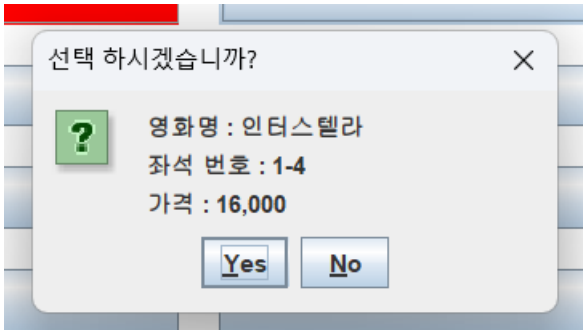


장르가 SF인 영화들이 나오고, 예매 가능한 버튼이 나온다. '인터스텔라'를 예매하러 간다고 가정한다.



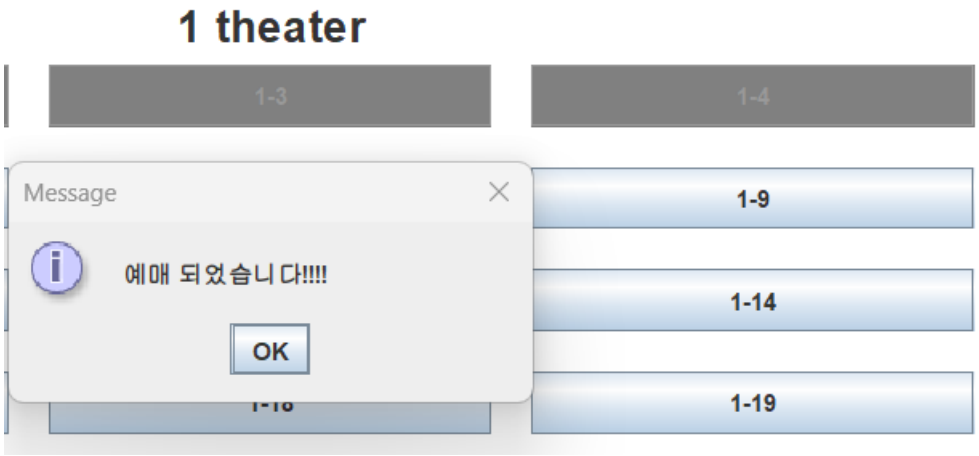
인터스텔라 예매 하러 가기 버튼을 클릭하면 해당 영화의 상영날짜들이 표시된다.

'2024-01-05' 날짜를 클릭했다고 가정한다. 그러면 해당 날짜에 상영될 상영관 번호와 함께 좌석들을 표시한다. 이미 예매된 좌석은 비활성화 시킴과 동시에 배경색을 회색으로 한다.



예매할 좌석을 클릭하면 확인 창과 함께 좌석 선택 확정 의사를 물어본다.

예매할 좌석들은 빨간색 배경으로 변경된다.



예매하기 버튼을 누르면 "예매 되었습니다!!!!" 라는 문구와 함께 메시지 창을 띄운다. 그리고 해당 좌석들은 비활성화 상태로 배경색은 회색으로 변경된다.

3 – 2. 영화 예매 시 DB 구조

reservation_id	payment_method	payment_status	payment_amount	member_id	payment_date
1	카드	1	32000	user1	2024-01-01 00:00:00
2	카드	1	16000	user1	2024-01-02 00:00:00
3	카드	1	32000	user1	2024-06-03 11:05:16
NULL	NULL	NULL	NULL	NULL	NULL

reservation 테이블에 좌석 2개를 예매했으므로 32000, 그리고 현재 날짜와 함께 잘 저장된 것을 볼 수 있다. 본 프로젝트에서는 카드로만 결제이 진행되는 것으로 가정했다.

ticket_id	screening_schedule_id	theater_id	seat_id	reservation_id	issuance_status	selling_price	standard_price
1	1	1	1-1	1	1	16000	16000
2	1	1	1-2	1	1	16000	16000
3	2	2	2-3	2	1	16000	16000
4	1	1	1-3	3	1	16000	16000
5	1	1	1-4	3	1	16000	16000
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

ticket 테이블에 초기화 데이터 밑에 4, 5번에 좌석 별로 티켓 데이터가 잘 추가된 것을 볼 수 있다.

reserved_id	seat_id	screening_schedule_id	theater_id
1	1-1	1	1
2	1-2	1	1
3	2-3	2	2
4	1-3	1	1
5	1-4	1	1
NULL	NULL	NULL	NULL

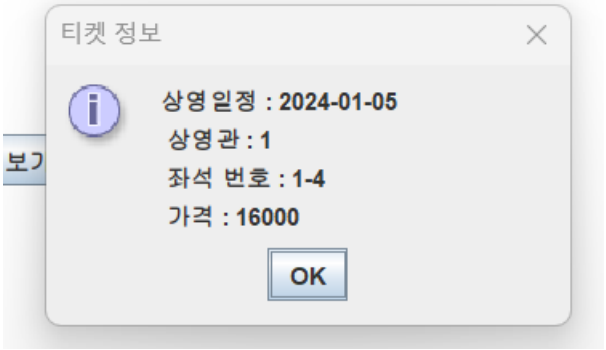
reserved_seat 테이블에 초기화 데이터 밑에 4, 5번에 데이터가 잘 삽입된 것을 볼 수 있다. reserved_seat 테이블에는 상영 일정 번호와 함께 저장되어 상영 일정 각각의 영화에 대한 상영관 번호와 예매된 좌석을 볼 수 있게 하였다.

4 – 1. 예매 내역 확인 시 테스팅 방법 및 GUI 구조

일반 회원으로 로그인 후 예매 내역 버튼을 클릭하면 위와 같은 화면이 나타난다.

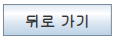
앞에서 추가했던 데이터들과 초기화 데이터들이 모두 화면에 표시된다. 데이터가 화면 밖을 벗어날 경우를 대비하여 scroll 이 가능하도록 하였다.

<상세보기>



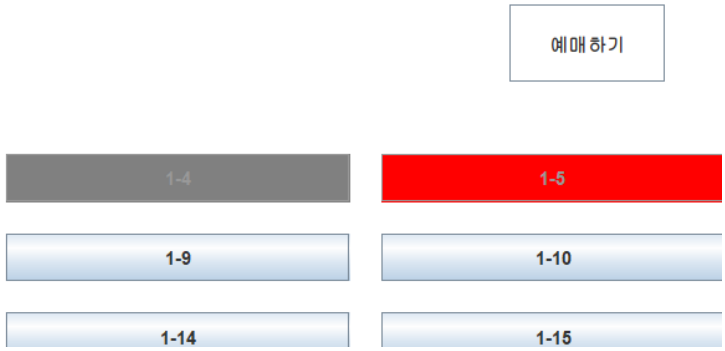
상세보기를 클릭하면 티켓 정보 창이 나타나도록 구현하였다.

<예매 변경>

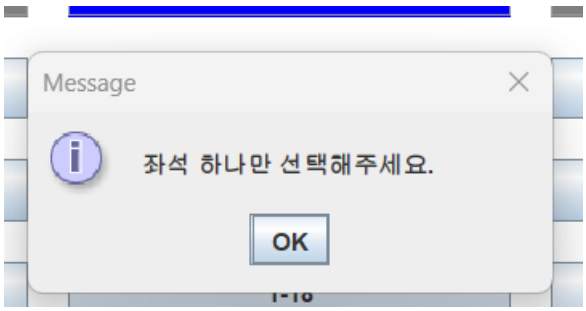


예매 변경 버튼을 클릭하면 다시 영화를 예매하는 화면이 나타나게 된다.

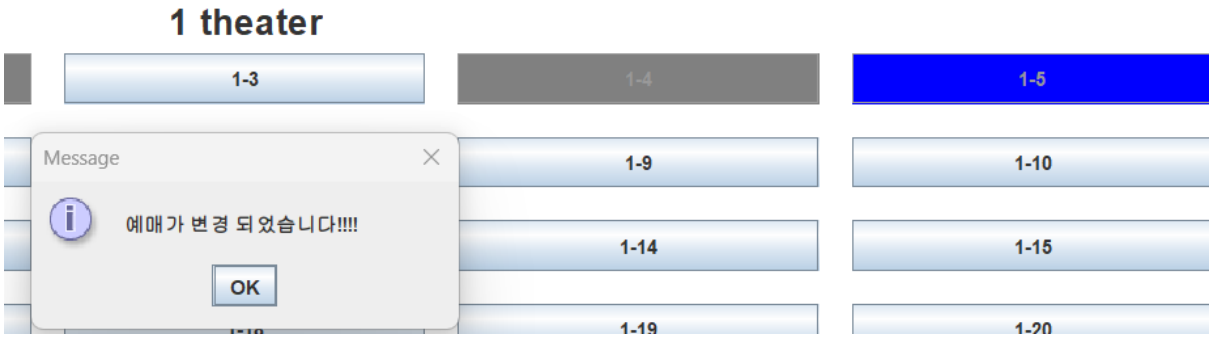
영화를 예매하는 경우와는 다른 점은 예매 변경 하려는 좌석이 파란색 배경으로 화면에 나타난다.



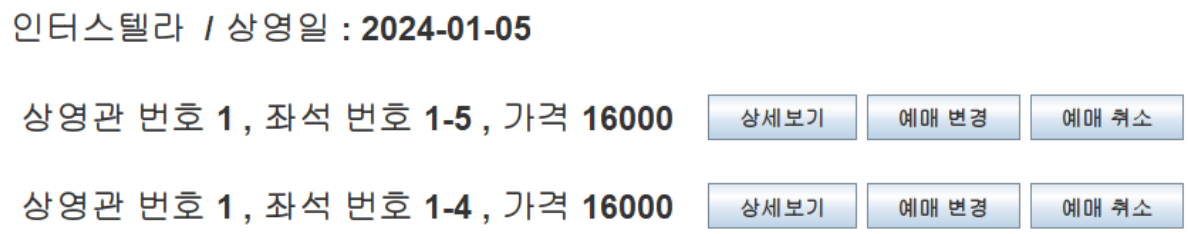
이전과 마찬가지로 예매하려는 좌석을 클릭하면 빨간색 배경으로 변경된다.



이전과 다른 점은 티켓 하나 당 예매를 변경하려는 작업이기 때문에 여러 개의 좌석을 클릭하면 "좌석 하나만 선택해주세요" 라고 하는 문구와 함께 메시지 창이 나타나게 된다.



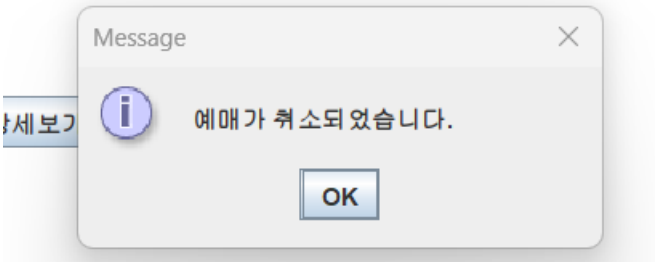
예매하기 버튼을 누르면 "예매가 변경 되었습니다!!!!" 라는 문구와 함께 메시지 창이 나타나고, 예매 변경한 좌석이 파란색 배경으로 나타나고 이전에 예매했던 좌석은 다시 선택 가능한 형태로 바뀌게 된다. 확인 버튼을 누르면 나의 예매 내역을 확인할 수 있는 화면으로 다시 변경된다.



예매 변경이 잘 된 것을 볼 수 있다.

<예매 취소>

위의 그림에서 좌석 번호가 1-5인 예매를 예매 취소 버튼을 눌러 취소하려고 한다.



"예매가 취소되었습니다." 라는 문구와 함께 메시지창이 뜨게 되고,

해당 좌석이 알맞게 취소되고 나의 예매 내역에서 삭제된 것을 볼 수 있다.

3 – 2. 예매 내역 확인 시 DB 구조

관리자 모드로 로그인 한 후 초기화 버튼을 클릭하여 데이터들을 초기화 상태로 바꾼다.

상영일이 2024-01-05 이고 좌석 번호가 1 – 1인 인터스텔라 영화의 예매를 상영일이 2024-12-21이고 좌석 번호가 2-39인 인셉션 영화로 변경하고 취소해본다고 가정한다.

해당 예매가 원하는 대로 잘 변경된 것을 볼 수 있다.

보고서 작성 도중에 같은 예매 내에서 변경된 것을 보여주기 위해 나의 예매 내역 화면에 예매 번호도 함께 보여주는 것으로 변경했습니다.

나의 예매 내역

인터스텔라 / 상영일 : **2024-01-05**

예매 번호 : **1** , 상영관 번호 **1** , 좌석 번호 **1-2** , 가격 **16000** 상세보기 예매 변경 예매 취소

인터스텔라 / 상영일 : **2024-01-06**

예매 번호 : **2** , 상영관 번호 **2** , 좌석 번호 **2-3** , 가격 **16000** 상세보기 예매 변경 예매 취소

변경했던 예매를 예매 취소 버튼을 누르면 예매 내역에서 사라진 것을 볼 수 있다.

<예매 변경 시>

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

ticket_id	screening_schedule_id	theater_id	seat_id	reservation_id	issuance_status	selling_price	standard_price
1	1	1	1-1	1	1	16000	16000
2	1	1	1-2	1	1	16000	16000
3	2	2	2-3	2	1	16000	16000
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

ticket_id	screening_schedule_id	theater_id	seat_id	reservation_id	issuance_status	selling_price	standard_price
1	34	2	2-39	1	1	16000	16000
2	1	1	1-2	1	1	16000	16000
3	2	2	2-3	2	1	16000	16000
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

위에는 초기화 데이터일 경우의 ticket 테이블의 데이터들이다. 영화 예매를 다른 영화에 다른 날짜로 변경했을 경우의 ticket 테이블의 변화가 밑의 그림이다.

reserved_id	seat_id	screening_schedule_id	theater_id
1	1-1	1	1
2	1-2	1	1
3	2-3	2	2
NULL	NULL	NULL	NULL

reserved_id	seat_id	screening_schedule_id	theater_id
1	2-39	34	2
2	1-2	1	1
3	2-3	2	2
NULL	NULL	NULL	NULL

위에는 초기화 데이터일 경우의 reserved_seat 테이블의 데이터들이다. 영화 예매를 다른 영화에 다른 날짜로 변경했을 경우의 reserved_seat 테이블의 변화가 밑에 그림이다.

ticket_id	screening_schedule_id	theater_id	seat_id	reservation_id	issuance_status	selling_price	standard_price
2	1	1	1-2	1	1	16000	16000
3	2	2	2-3	2	1	16000	16000
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

reserved_id	seat_id	screening_schedule_id	theater_id
2	1-2	1	1
3	2-3	2	2
NULL	NULL	NULL	NULL

예매를 취소하였을 경우에 ticket 테이블, reserved_seat 테이블 모두 데이터가 잘 삭제된 것을 볼 수 있다.

reservation_id	payment_method	payment_status	payment_amount	member_id	payment_date
2	카드	1	16000	user1	2024-01-02 00:00:00
NULL	NULL	NULL	NULL	NULL	NULL

같이 예매했던 예매들을 모두 삭제할 경우에는 reservation 테이블도 해당 예매 정보가 잘 삭제되는 것을 볼 수 있다.