DOKUZ EYLÜL UNIVERSITY
ENGINEERING FACULTY
DEPT. OF COMPUTER ENGINEERING

<ONLINE CINEMA TICKET>

CME 3201 Database Management Systems
Term Project Report

Phase V

2019-2020 FALL

2015510122 FATMA KAHRAMAN
2015510066 OĞUZHAN YARDIMCI

1. Introduction

The Online Cinema Ticket Project, which will be realized according to the demands of the CEO, is software that members can buy online cinema tickets. The project is a database management system that enables the sale of cinema tickets over the Internet. This report includes the scenario of the project, which was created in accordance with the requests of the cinema CEO. Moreover, there are schemas that the developers will understand. There is an Entity Relation Diagram, Relation Diagram, Class Diagram, CRUD Matrix that developers understand.

It is aimed for the members to enter the cinema website and purchase online cinema tickets from the movies listed in the vision according to the city and region they want. For the first time, a person must be a member of the system to buy one or two tickets in the cinema. If she/he is a member of the cinema, she/he can enter the system directly with her identification numbers and names and buy movie tickets. They can also have pop-corn and cola on request.

Each cinema has a grocery. Seat prices vary by room type. There are 3 types of movie theaters: 3D, IMAX, 2D.

2. Overview

The member can only buy 2 tickets from their membership at one time. The member cannot buy tickets to another movie in the same session. There are 3 types of section; noon section, evening section and morning section. The audience can buy pop-corn and cola in addition to the ticket of the screen of the ticket purchase.

The CEO of Cinema can insert, update and delete movies to the vision. CEO can update productions prices in the grocery and their discount rates. The CEO can monitor the revenue of the cinema and filter this process according to various elements. She/he can get the statistics of cinema with the filters she/he wants.

3. Assumptions/Constraints/Risks

3.1 Assumptions

Database part of this project was developed using PostgreSQL.

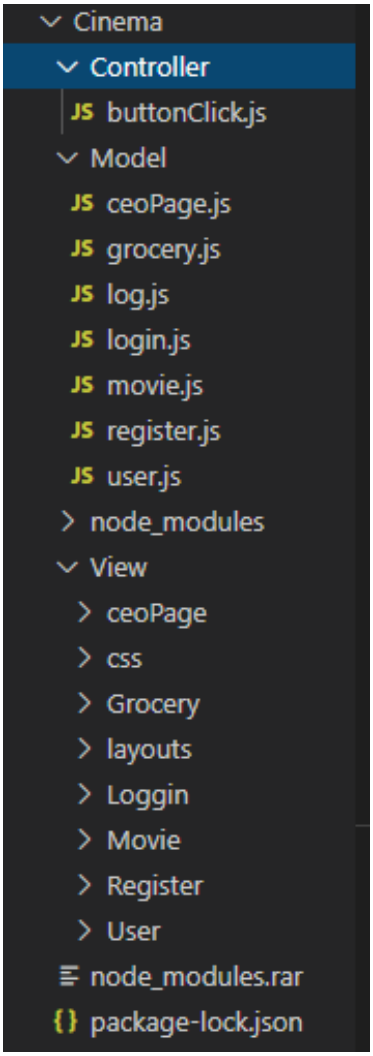
3.2 Constraints

A person must be a member of the system to buy tickets in the Online Cinema Ticket System. In order to buy pop-corn and cola, cinema tickets must be purchased first.

3.3 Risks

A person must be a member of the system to buy tickets in the Online Cinema Ticket System. Cinema tickets must be sold first. After that you can take a pop-corn and cola.

4. Software Architecture

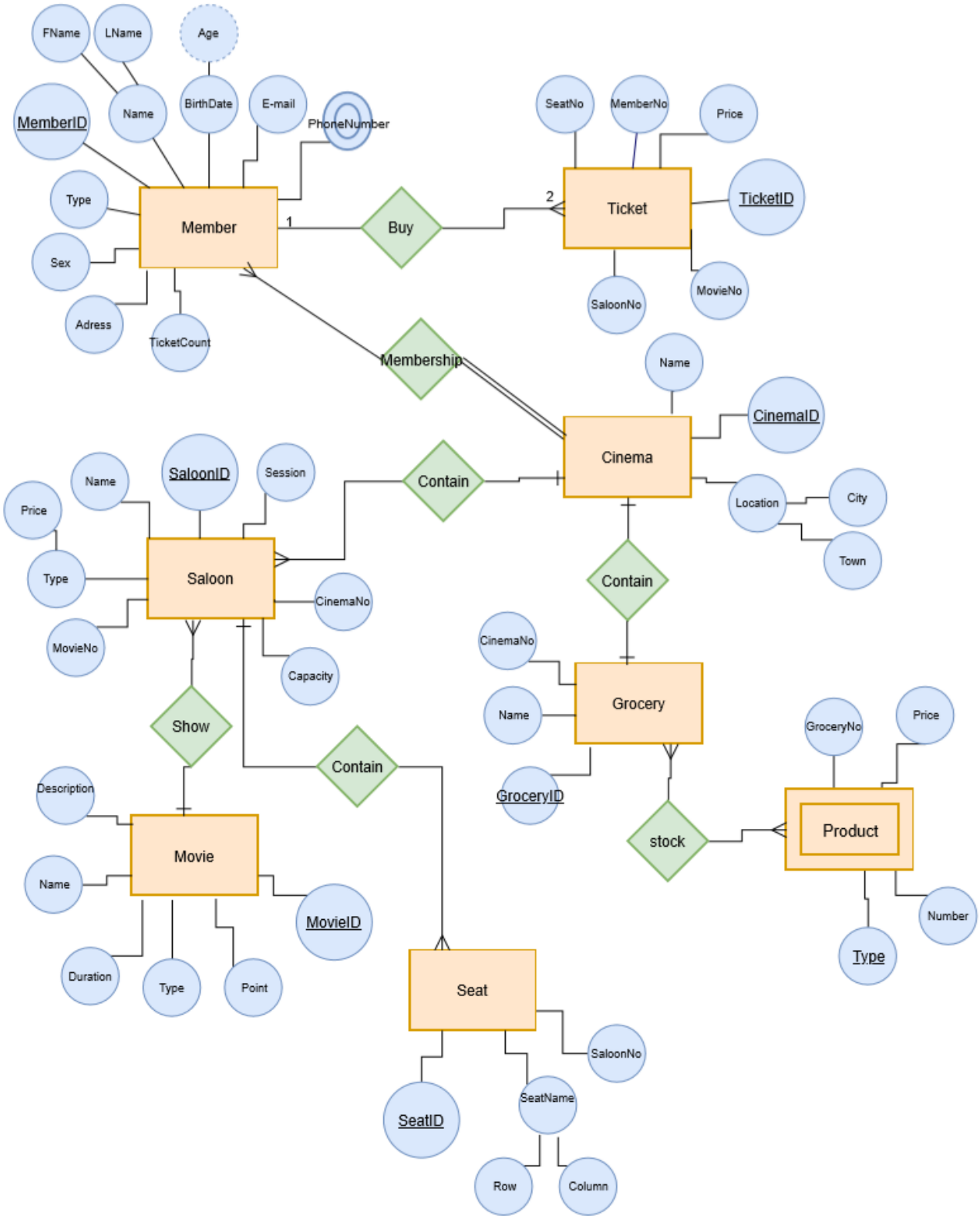


3-tier architecture was used to develop in the Online Cinema Ticket System. Data layer was created by PostgreSQL. Application layer was contained Node.js (Express Framework). Presentation layer was created using HTML5, JavaScript and CSS. The client was created with Node.js thanks to express framework. As a result, database and frontend connected successfully. In addition, Model View Controller was used it.

MVC was created in Node.js with file tree as in screenshot. Model included JavaScript folders mean that business job and back-end sections. View included front-end section. Task of Controller connected front-end and back-end each other. It includes button action.

5. Detailed System Design

5.1 Entity-Relationship Diagram



5.2 Relational Algebra Expressions

1. Names of films in the cinema hall

$$\pi_{cinemaName}(Saloon \bowtie_{movieNo=movieID} Movie)$$

2. Phone numbers and e-mails of members 18 years and older

$$\pi_{e_{mail},phoneNumber}\ (\sigma_{age \geq 18}(\text{Member}))$$

3. Films in Vision both in Izmir and Istanbul

$$\pi_{movieName}(\sigma_{location=\text{"İzmir"}}\ ((Cinema \bowtie_{movieNo=movieID} Saloon)$$
$$\bowtie_{movieNo=movieID} Movie\ \pi_{movieName}($$
$$\sigma_{location=\text{"İstanbul"}}\ ((Cinema \bowtie_{movieNo=movieID} Saloon) \bowtie_{movieNo=movieID} Movie)$$

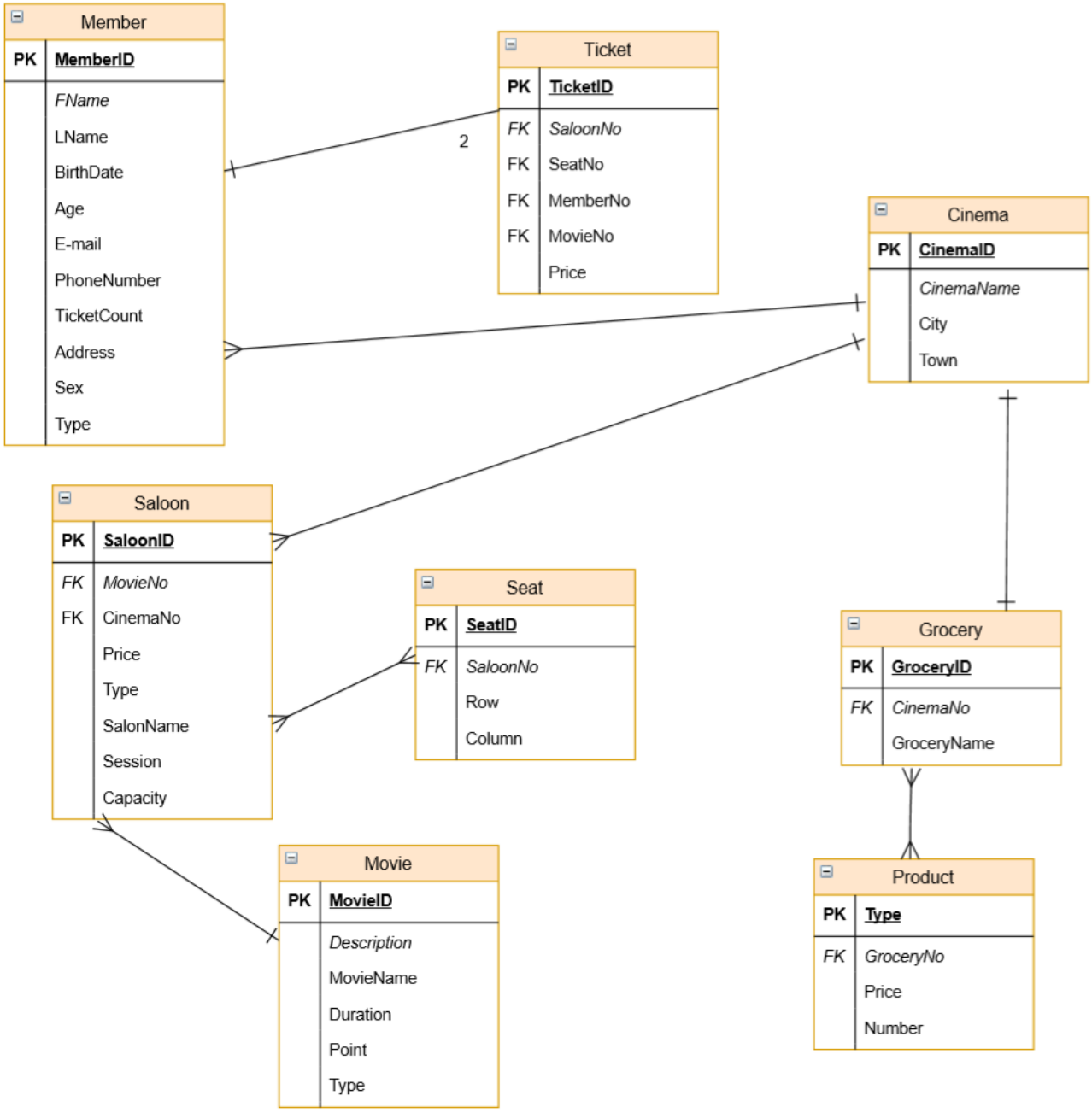
4. Names of members who watch horror movies

$$\pi_{movieFName,movieLName}((\sigma_{type=\text{"horror"}}((\sigma_{type=\text{"horror"}}((Movie \bowtie_{movieID=movieNo} Ticket))$$
$$\bowtie_{memberNo=memberID} Member)$$

5. Stores with more than 20 Coke and popcorn in stock

$$\pi_{groceryNo}(\sigma_{productNumber>20}(\sigma_{productName=\text{"cola"}}(Product)) \cup$$
$$(\sigma_{productNumber>20}(\sigma_{productName=\text{"popcorn"}}(Product)$$

5.3 Class Diagram

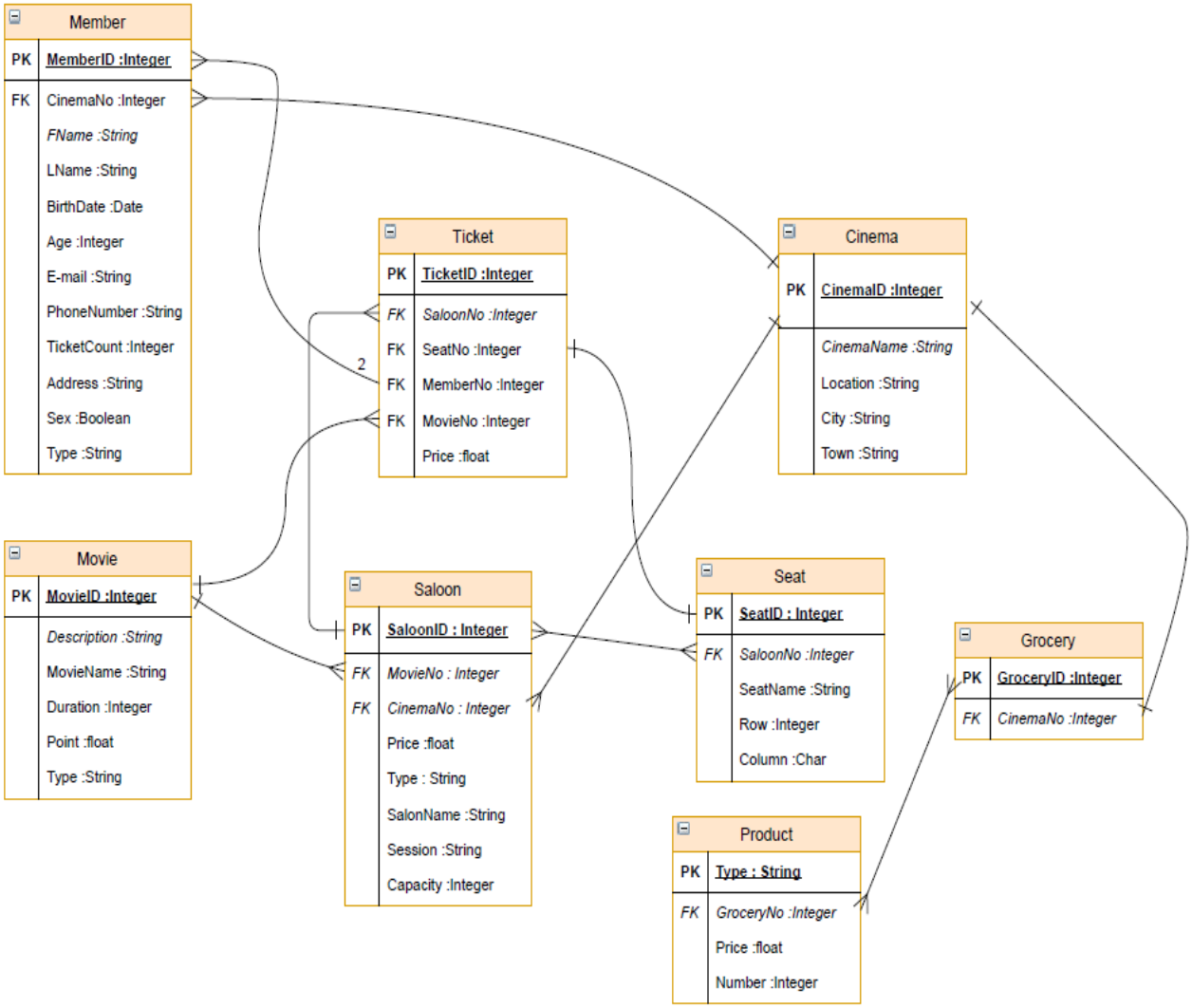


5.4 CRUD Matrix

Relations\Operations	Create	Read	Update	Delete
Cinemas	X	X	X	
Groceries	X	X	X	
Members	X	X		
Movies	X	X	X	X
Movies Log	X	X		
Products	X	X	X	
Saloons	X	X		
Seats	X	X	X	
Tickets	X	X		

All Crud Operations are not applied to all relationships. It was applied to the necessary relationships.

5.5 LOGICAL DATABASE SCHEMA



Appendix A: SQL Statements

CREATE OPERATION

```
CREATE TABLE Cinema (
        Cinema_ID SERIAL PRIMARY KEY,
        Cinema_Name TEXT not NULL UNIQUE,
        Location_City Text not NULL,
        Location_Town Text not NULL
);

CREATE TABLE Members (
        Member_ID SERIAL PRIMARY KEY,
        Cinema_No int NOT NULL Default 1 references Cinema(Cinema_ID),
        First_Name TEXT NOT NULL,
        Last_Name TEXT NOT NULL,
        Age int NOT NULL,
        Email TEXT NOT NULL UNIQUE,
        PhoneNumber TEXT NOT NULL UNIQUE,
        MemPassword TEXT NOT NULL,
        TicketCount int NOT NULL DEFAULT 0,
        Address TEXT NOT NULL,
        Sex TEXT NOT NULL,
        MemberType TEXT NOT NULL Default 'Normal',
        CHECK (Sex IN ('Female', 'Male')),
        CHECK (MemberType IN ('Ceo','Student', 'Normal')),
        CHECK (Email ~* '^\w+@\w+[.]\w+$'),
        CHECK (char_length(MemPassword)>=8)
);

CREATE TABLE Movie (
        Movie_ID SERIAL PRIMARY KEY,
        MovieName Text not NULL UNIQUE,
        Duration int not NULL ,
        Points float not NULL,
        MovieType Text not NULL,
        Description TEXT
);

CREATE TABLE Saloon(
        Saloon_ID SERIAL PRIMARY KEY,
        SaloonName TEXT NOT NULL,
        Movie_No int NOT NULL References Movie(Movie_ID),
```

```
        Cinema_No int NOT NULL DEFAULT 1 References Cinema(Cinema_ID),
        Price float NOT NULL,
        SaloonType TEXT NOT NULL,
        SaloonSession TEXT NOT NULL,
        Capacity int NOT NULL Default 30,
        check (SaloonType IN ('IMAX', '3D','2D')),
        check (SaloonSession IN ('Morning','Noon','Evening')),
        CHECk (Capacity <= 30)
);

CREATE TABLE Seat(
        Seat_ID SERIAL PRIMARY KEY,
        SeatName TEXT NOT NULL,
        Saloon_No int NOT NULL References Saloon(Saloon_ID),
        SRow int NOT NULL,
        SColumn Char NOT NULL,
        check (SRow IN (1,2,3,4,5,6)),
        check (SColumn IN ('A', 'B','C','D','E'))
);

CREATE TABLE Ticket(
        Ticket_ID SERIAL PRIMARY KEY,
        Saloon_No int not null References Saloon(Saloon_ID),
        Seat_No int not null References Seat(Seat_ID),
        Member_No int not null References Members(Member_ID),
        Movie_No int not null References Movie(Movie_ID),
        TicketPrice float not null
);

CREATE TABLE Grocery(
        Grocery_ID SERIAL Primary key,
        Cinema_No int NOT NULL References Cinema(Cinema_ID)
);
CREATE TABLE Product(
        ptype TEXT NOT NULL Primary Key ,
        groceryno int NOT NULL References Grocery(Grocery_ID),
        price float NOT NULL,
        stock int NOT NULL
);
```

BASIC OPERATION ON DATABASE

SELECT SUM(ticketprice) FROM cinema_app.tickets

SELECT ptype,price,stock FROM cinema_app.products

INSERT INTO cinema_app.products(ptype, price, stock) VALUES ($1, $2, $3)

SELECT * FROM cinema_app.products WHERE ptype=$1

UPDATE cinema_app.products SET ptype=$1, price=$2, stock=$3 WHERE ptype=$4

SELECT * FROM cinema_app.movies_log

SELECT * FROM cinema_app.members WHERE email=$1 AND mempassword=$2

SELECT * FROM cinema_app.movies ORDER BY movie_id ASC

CALL cinema_app.insert_movie($1,$2,$3,$4,$5)

SELECT * FROM cinema_app.movies WHERE movie_id=$1

UPDATE cinema_app.movies SET moviename=$1, duration=$2, points=$3, movietype=$4, description=$5 WHERE movie_id=$6

DELETE FROM cinema_app.movies WHERE movie_id=$1

INSERT INTO cinema_app.members(first_name, last_name, email, phonenumber, mempassword,address,sex,age) VALUES($1,$2,$3,$4,$5,$6,$7,$8)

SELECT * FROM cinema_app.movies ORDER BY movie_id ASC

SELECT productType , ProductPrice FROM cinema_app.product

SELECT * FROM cinema_app.seats seat INNER JOIN cinema_app.saloons saloon ON seat.saloon_no=saloon.saloon_id WHERE movie_no=$1 AND saloonsession=$2

SELECT * from cinema_app.showticket WHERE member_id=$1 AND movie_id=$2 AND saloonsession=$3 AND seatname IN ($4,$5)

SELECT * FROM cinema_app.showgrocery

COMPLEX QUERY

```
INSERT INTO cinema_app.tickets (saloon_no, seat_no, member_no, movie_no, ticketprice)
    SELECT saloon_no, seat_id, $3, $4, $5
    FROM cinema_app.seats
    WHERE seatname=$2
      AND
        saloon_no = (SELECT saloon_id
          FROM cinema_app.saloons
          WHERE movie_no=1 AND saloonsession=$1)
```

TRIGGER 1:

```
SET search_path to cinema_app;
SET ROLE cinema_app;

CREATE OR REPLACE FUNCTION cinema_app.seat_trg () RETURNS
TRIGGER AS
$$
BEGIN
    UPDATE cinema_app.seats
    SET isfull=true
    WHERE seat_id = (SELECT seat_no
                     FROM cinema_app.tickets
                     ORDER BY ticket_id DESC
                     LIMIT 1);
    RETURN NULL;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER seatfull AFTER
    INSERT ON cinema_app.tickets
    FOR EACH ROW
        EXECUTE PROCEDURE cinema_app.seat_trg ();
```

TRIGGER 2:

```
CREATE OR REPLACE FUNCTION cinema_app.movies_log_trg () RETURNS
TRIGGER AS
$$
BEGIN
    IF TG_OP='INSERT' THEN
        INSERT INTO cinema_app.movies_log SELECT NEW.*, 'INSERT', NOW();
    ELSEIF TG_OP='UPDATE' THEN
        INSERT INTO cinema_app.movies_log SELECT NEW.*, 'UPDATE', NOW();
    ELSEIF TG_OP='DELETE' THEN
        INSERT INTO cinema_app.movies_log SELECT OLD.*, 'DELETE', NOW();
    END IF;
    RETURN NULL;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER moviestrigger AFTER
    INSERT OR UPDATE OR DELETE ON cinema_app.movies
    FOR EACH ROW
        EXECUTE PROCEDURE cinema_app.movies_log_trg ();
```

STORED PROCEDURE

```
CREATE OR REPLACE PROCEDURE cinema_app.insert_movie(moviename_text text,
                    duration_int int, points_int int, movietype_text text, description_text text)
LANGUAGE plpgsql
AS
$$
BEGIN
    INSERT INTO cinema_app.movies(moviename, duration, points, movietype, description)
    VALUES(moviename_text, duration_int, points_int, movietype_text, description_text);
END;
$$;

CALL cinema_app.insert_movie('filmadı',1,1,'filmtipi','açıklama');
```

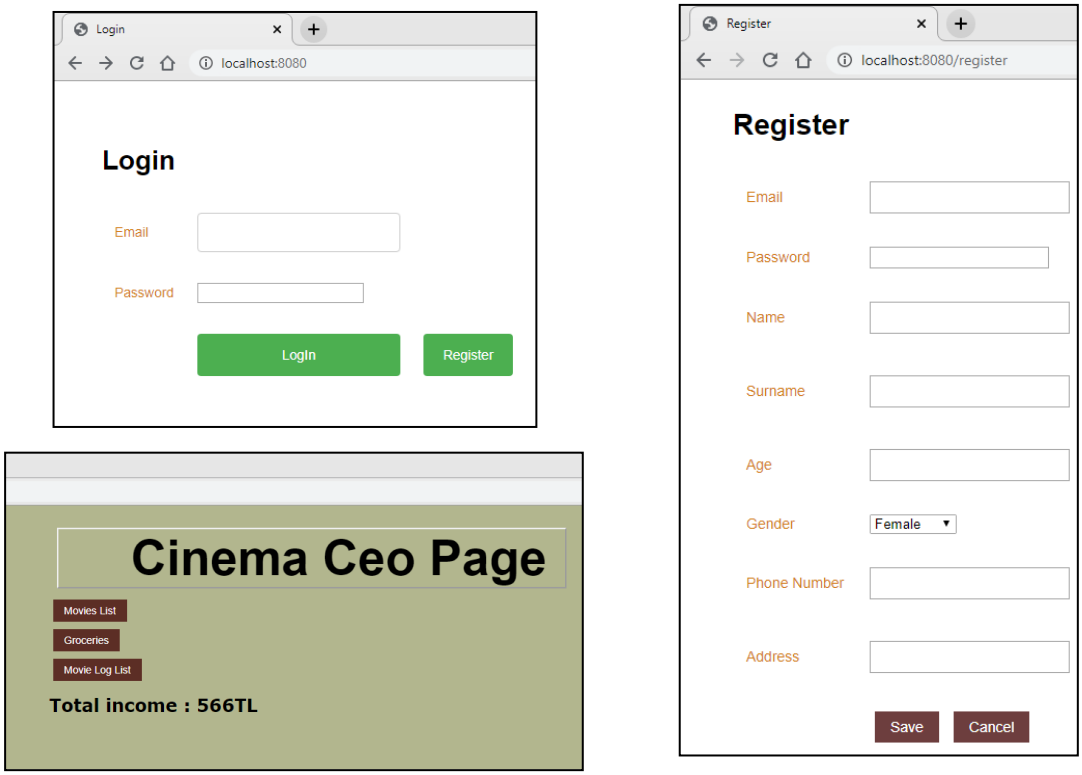
VIEW: 1

```
CREATE VIEW cinema_app.showticket AS
 SELECT members.member_id,
    movies.movie_id,
    cinemas.cinema_id,
    members.first_name,
    members.last_name,
    members.membertype,
    members.ticketcount,
    movies.moviename,
    saloons.saloonname,
    saloons.saloontype,
    saloons.price,
    saloons.saloonsession,
    cinemas.cinema_name,
    cinemas.location_city,
    cinemas.location_town,
    seats.seatname
   FROM cinema_app.members
     JOIN cinema_app.cinemas ON members.cinema_no = cinemas.cinema_id
     JOIN cinema_app.saloons ON saloons.cinema_no = cinemas.cinema_id
     JOIN cinema_app.movies ON movies.movie_id = saloons.movie_no
     JOIN cinema_app.seats ON seats.saloon_no = saloons.saloon_id;
```

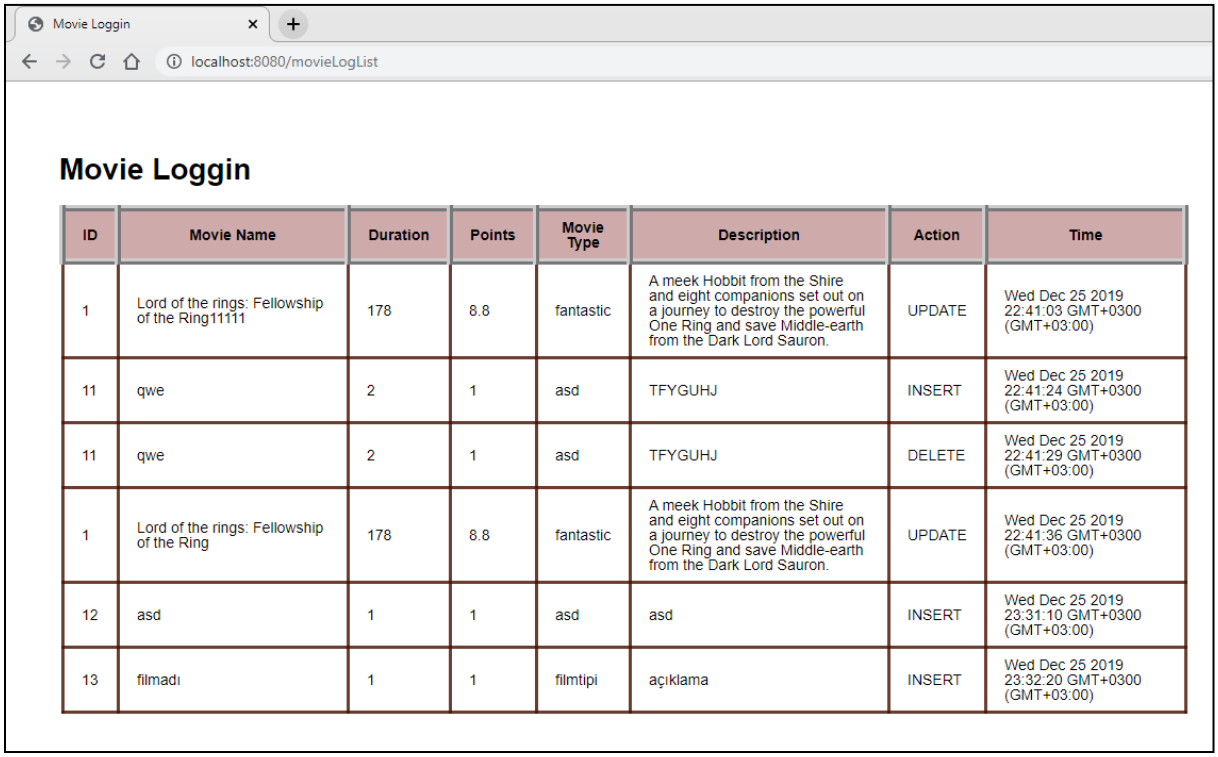
VIEW: 2

```
CREATE VIEW cinema_app.showgrocery AS
      SELECT groceries.cinema_no,
    products.ptype,
    products.price AS pprice,
    products.stock
   FROM cinema_app.products
     JOIN cinema_app.groceries ON products.groceryno = groceries.grocery_id;
```

Appendix B: Screenshots

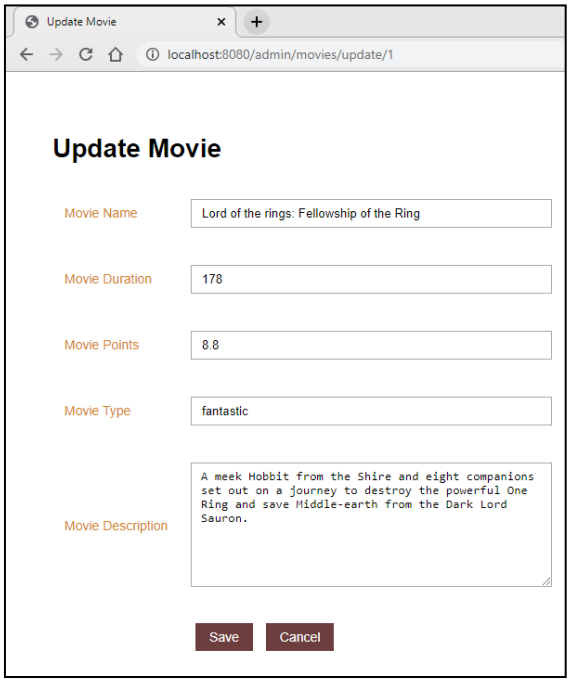


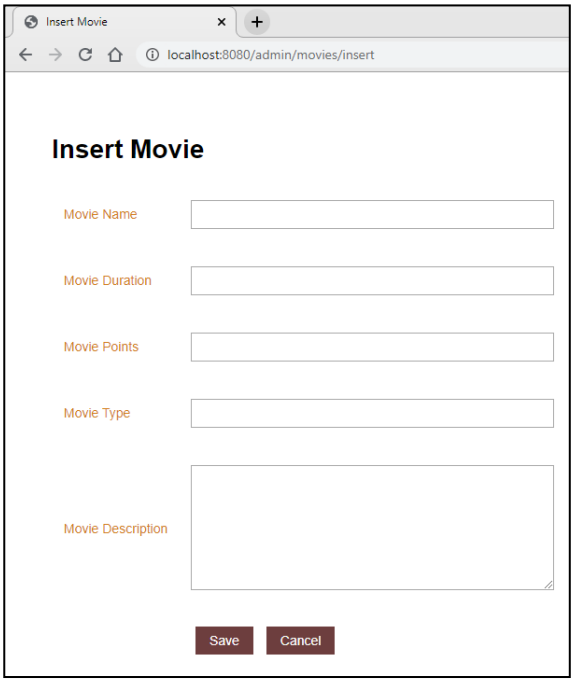
Login Screen and Register Screen interfaces in addition Admin Page

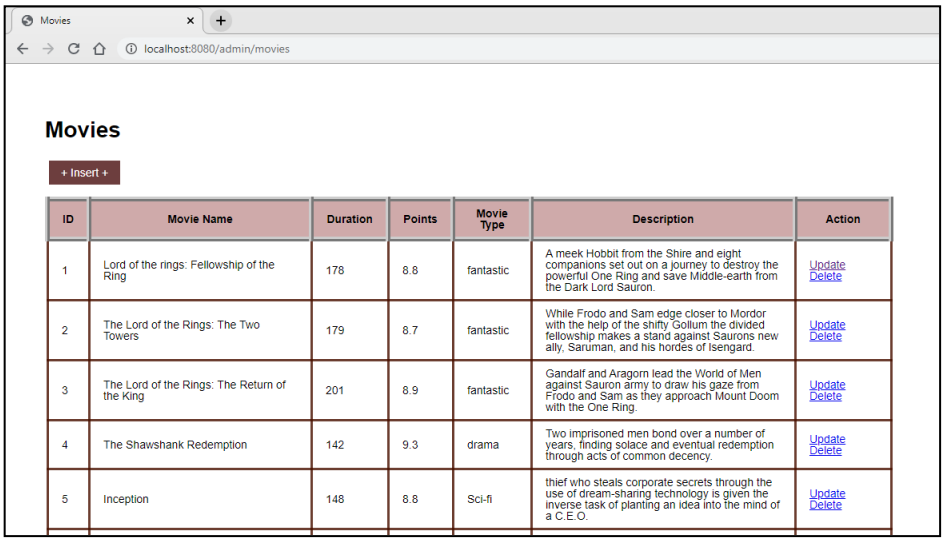


ID	Movie Name	Duration	Points	Movie Type	Description	Action	Time
1	Lord of the rings: Fellowship of the Ring11111	178	8.8	fantastic	A meek Hobbit from the Shire and eight companions set out on a journey to destroy the powerful One Ring and save Middle-earth from the Dark Lord Sauron.	UPDATE	Wed Dec 25 2019 22:41:03 GMT+0300 (GMT+03:00)
11	qwe	2	1	asd	TFYGUHJ	INSERT	Wed Dec 25 2019 22:41:24 GMT+0300 (GMT+03:00)
11	qwe	2	1	asd	TFYGUHJ	DELETE	Wed Dec 25 2019 22:41:29 GMT+0300 (GMT+03:00)
1	Lord of the rings: Fellowship of the Ring	178	8.8	fantastic	A meek Hobbit from the Shire and eight companions set out on a journey to destroy the powerful One Ring and save Middle-earth from the Dark Lord Sauron.	UPDATE	Wed Dec 25 2019 22:41:36 GMT+0300 (GMT+03:00)
12	asd	1	1	asd	asd	INSERT	Wed Dec 25 2019 23:31:10 GMT+0300 (GMT+03:00)
13	filmadı	1	1	filmtipi	açıklama	INSERT	Wed Dec 25 2019 23:32:20 GMT+0300 (GMT+03:00)

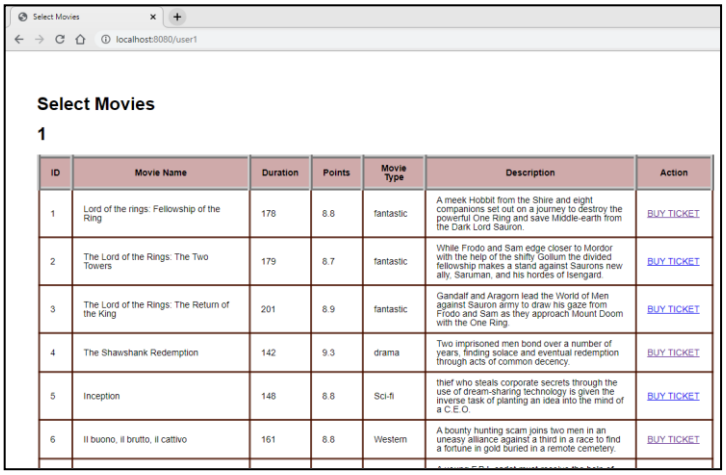
The log information is displayed by admin when changes are made to the movie table.

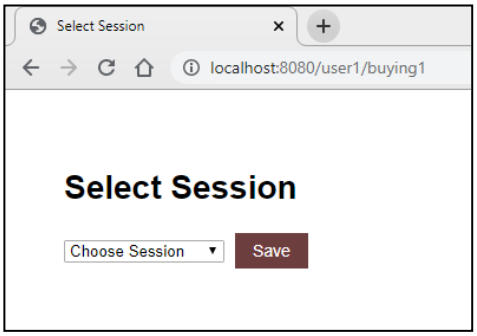






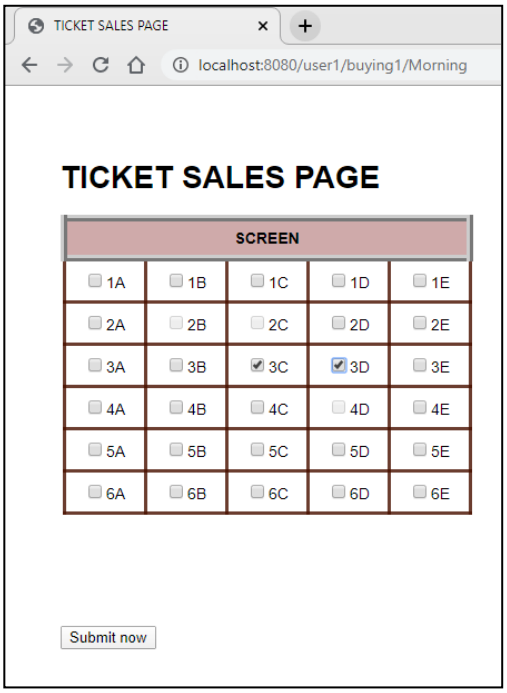
Movie table crud operation can be with this screens.



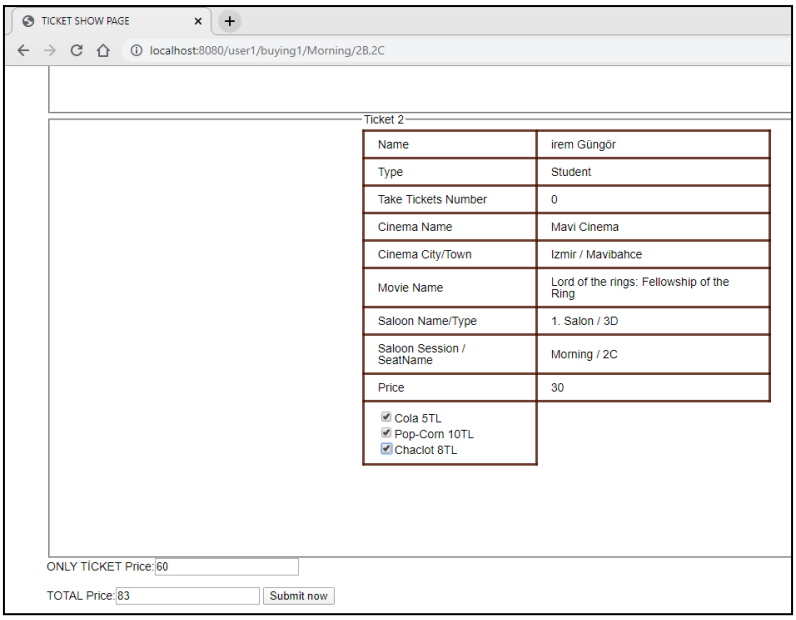


Take a ticket process 1: Select Movie

2: Select Session



3: Select seat



4: Select Product

5: Finish the taking a ticket process