**MOVIE DATABASE PROJECT**

**MOVIE DATABASE REQUIREMENTS**

1. **CONCEPT NOTE**

* Movie(MovieID (Primary Key), Title, ReleaseDate, Runtime, Language, Budget, BoxOfficeRevenue) to Genre((GenreID (Primary Key) , Name , Description)): A movie belongs to one genre, and a genre can include many movies. This is a one-to-many relationship.
* Movie to Director(DirectorID (Primary Key), FirstName, LastName, BirthDate, Nationality): A movie has one director, but a director can direct many movies. This is also a one-to-many relationship.
* Movie to Actor(ActorID (Primary Key) , FirstName , LastName , BirthDate , Nationality) through Character(CharacterID (Primary Key), Name, Description): A movie has many characters, and each character is played by one actor. An actor can play many characters across different movies. This is a many-to-many relationship, with Character serving as the associative entity.
* Movie to ProductionCompany(CompanyID (Primary Key) , Name , Address , Email, Phone): A movie can be produced by one or many production companies, and a production company can produce many movies. This could be a many-to-many relationship.
* Movie to Rating (RatingID (Primary Key), MovieID (Foreign Key), Source, Score,

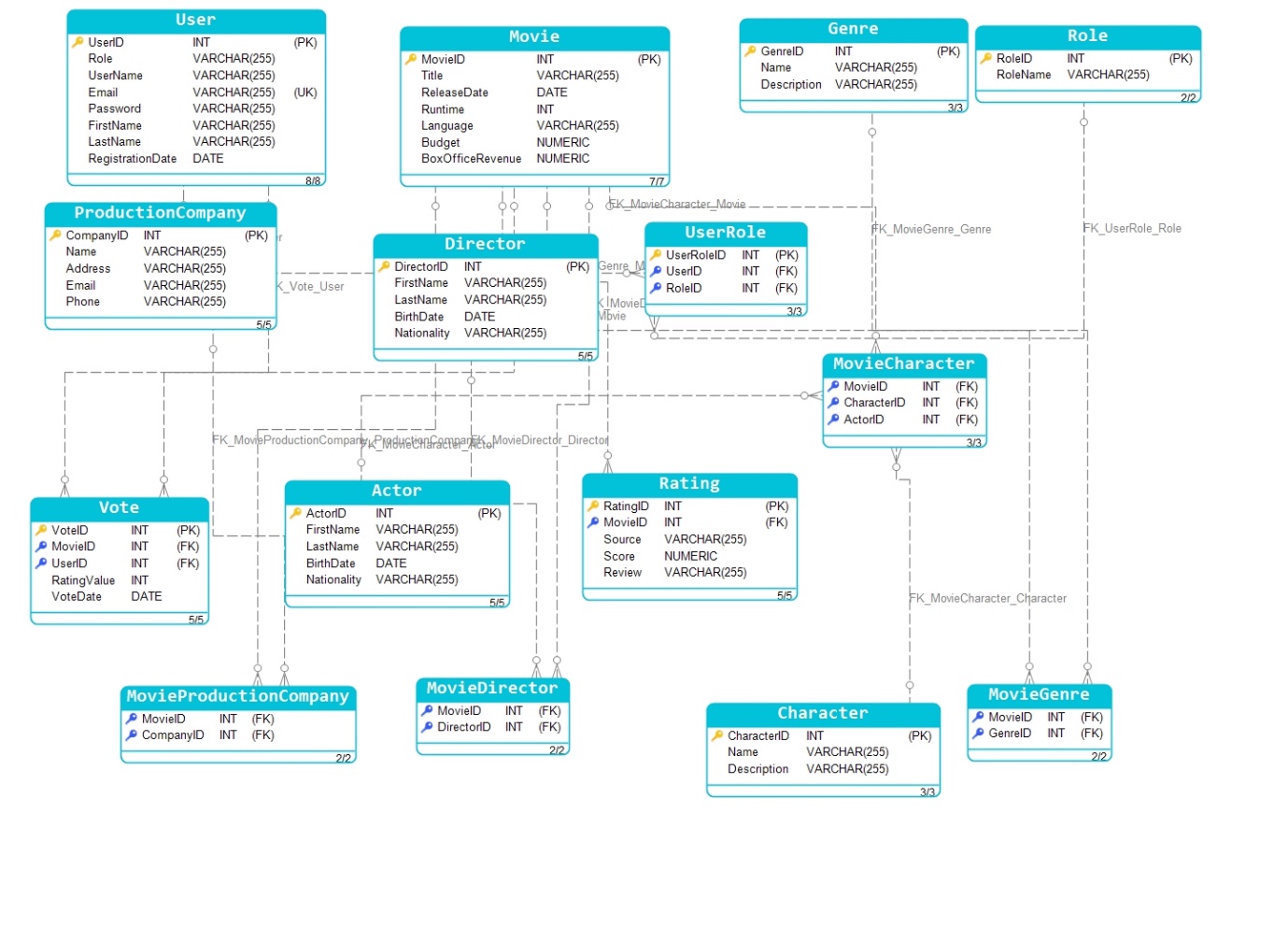
Review): A movie can have many ratings from different sources. This is a one-to-many relationship.

* User((UserID (Primary Key) , role, UserName, Email (Unique) Password , FirstName , LastName , RegistrationDate)) to Movie: A user can add, edit, update, and view movies. This implies that there should be permissions set up in the system to allow these actions. This is not typically represented as a relationship in an ERD but rather handled by the application logic and database permissions.
* User to Vote(VoteID (Primary Key), MovieID (Foreign Key), UserID (Foreign Key), RatingValue (Typically a number, e.g., on a scale of 1-5), VoteDate): A user can cast a vote on a movie. This is a one-to-many relationship (one user can vote on many movies, but only once per movie).
* Movie to Vote: A movie can have many votes from different users. This is a one-to-many relationship (one movie can have many votes).
* The Admin will be allowed to upload the movie so that Users can be able to view the movies on the other end
* To accommodate the role of an Admin who can upload movies, you would need to introduce a role-based access control system into your database schema. This typically involves creating additional tables to manage user roles and permissions. Additional Tables for Role-Based Access Control: Role (RoleID (Primary Key) , RoleName (e.g., 'Admin', 'User')) to UserRole (UserRoleID (Primary Key) , UserID (Foreign Key) , RoleID (Foreign Key))

Considerations:

* User can only vote once per movie
* User Can Create, Update View and Delete Movie
* Authentication and Authorization

1. **ENTITY RELATIONSHIP DIAGRAM**



**3. DATA DEFINITION LANAGUAGE**

-- Create Genre Table

CREATE TABLE Genre (

GenreID INT IDENTITY(1, 1) PRIMARY KEY,

Name VARCHAR(100) NOT NULL,

Description TEXT

);

-- Create Director Table

CREATE TABLE Director (

DirectorID INT IDENTITY(1, 1) PRIMARY KEY,

FirstName VARCHAR(100) NOT NULL,

LastName VARCHAR(100) NOT NULL,

BirthDate DATE,

Nationality VARCHAR(100)

);

-- Create Movie Table

CREATE TABLE Movie (

MovieID INT IDENTITY(1, 1) PRIMARY KEY,

Title VARCHAR(255) NOT NULL,

ReleaseDate DATE,

GenreID INT,

DirectorID INT,

Runtime INT,

Language VARCHAR(50),

Budget DECIMAL(18, 2),

BoxOfficeRevenue DECIMAL(18, 2),

FOREIGN KEY (GenreID) REFERENCES Genre(GenreID),

FOREIGN KEY (DirectorID) REFERENCES Director(DirectorID)

);

-- Create Actor Table

CREATE TABLE Actor (

ActorID INT IDENTITY(1, 1) PRIMARY KEY,

FirstName VARCHAR(100) NOT NULL,

LastName VARCHAR(100) NOT NULL,

BirthDate DATE,

Nationality VARCHAR(100)

);

-- Create Character Table

CREATE TABLE Character (

CharacterID INT IDENTITY(1, 1) PRIMARY KEY,

Name VARCHAR(100) NOT NULL,

ActorID INT,

MovieID INT,

Description TEXT,

FOREIGN KEY (ActorID) REFERENCES Actor(ActorID),

FOREIGN KEY (MovieID) REFERENCES Movie(MovieID)

);

-- Create ProductionCompany Table

CREATE TABLE ProductionCompany (

CompanyID INT IDENTITY(1, 1) PRIMARY KEY,

Name VARCHAR(255) NOT NULL,

Address TEXT,

ContactInfo VARCHAR(100)

);

-- Create User Table

CREATE TABLE Users (

UserID INT IDENTITY(1, 1) PRIMARY KEY,

Email VARCHAR(255) UNIQUE NOT NULL,

Password VARCHAR(255) NOT NULL, -- Should be hashed and salted in a real-world application

FirstName VARCHAR(100),

LastName VARCHAR(100),

RegistrationDate TIMESTAMP

);

-- Create Vote Table

CREATE TABLE Vote (

VoteID INT IDENTITY(1, 1) PRIMARY KEY,

MovieID INT,

UserID INT,

RatingValue TINYINT CHECK (RatingValue BETWEEN 1 AND 5),

VoteDate TIMESTAMP,

FOREIGN KEY (MovieID) REFERENCES Movie(MovieID),

FOREIGN KEY (UserID) REFERENCES Users(UserID),

UNIQUE (MovieID, UserID) -- Ensures one vote per user per movie

);

-- Create MovieProductionCompany Relationship Table (for many-to-many relationship)

CREATE TABLE MovieProduction (

MovieID INT,

CompanyID INT,

PRIMARY KEY (MovieID, CompanyID),

FOREIGN KEY (MovieID) REFERENCES Movie(MovieID),

FOREIGN KEY (CompanyID) REFERENCES ProductionCompany(CompanyID)

);

-- Create Role Table

CREATE TABLE Role (

RoleID INT IDENTITY(1, 1) PRIMARY KEY,

RoleName VARCHAR(50) UNIQUE NOT NULL

);

-- Create UserRole Table

CREATE TABLE UserRole (

UserRoleID INT IDENTITY(1, 1) PRIMARY KEY,

UserID INT,

RoleID INT,

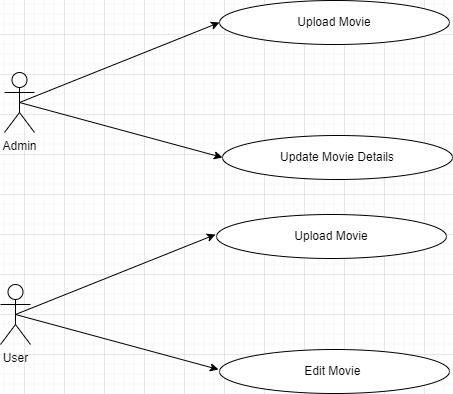
FOREIGN KEY (UserID) REFERENCES Users(UserID),

FOREIGN KEY (RoleID) REFERENCES Role(RoleID),

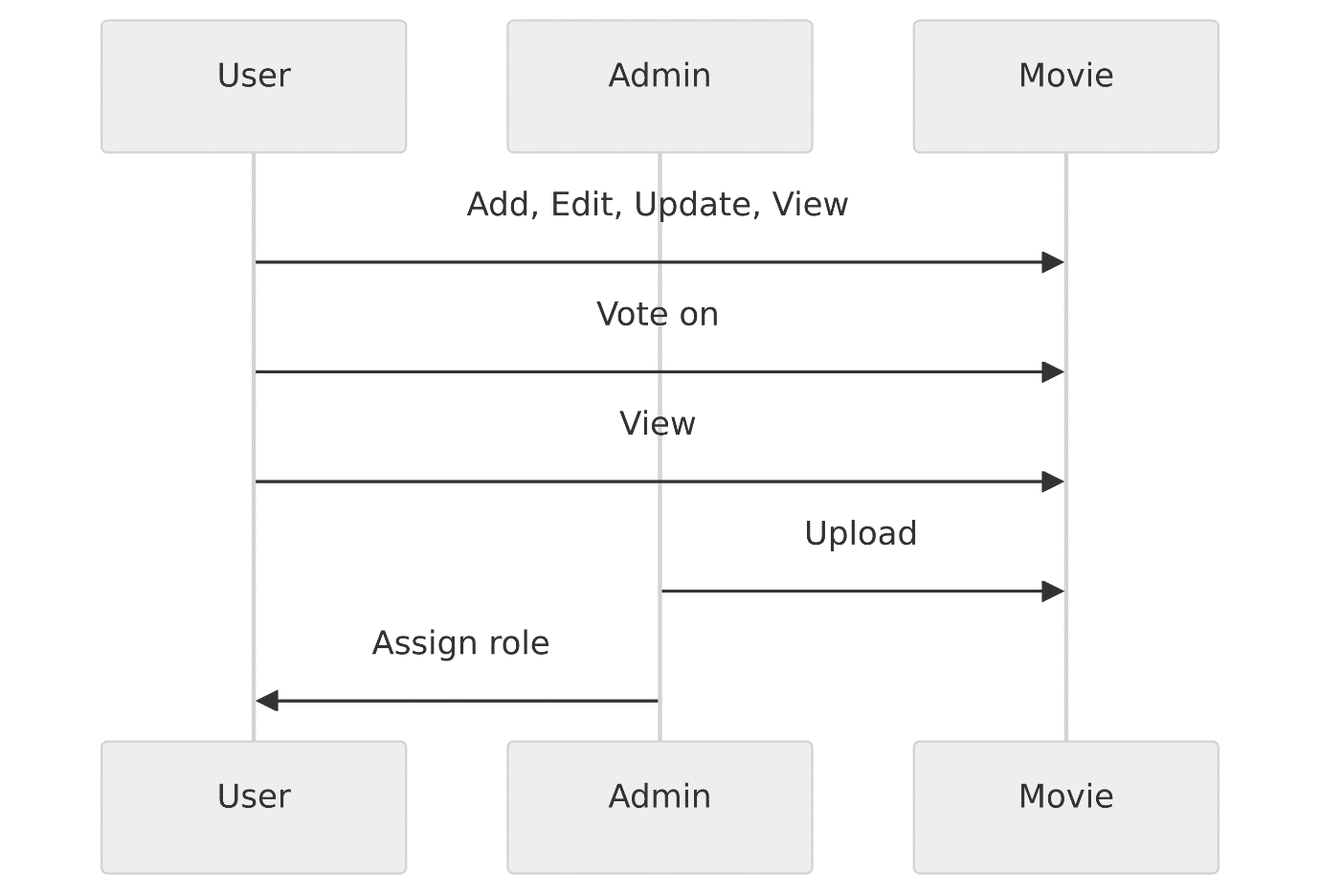
UNIQUE (UserID, RoleID) -- Ensures a user doesn't have the same role multiple times

);

**USE CASE**



**SEQUENCE DIAGRAM**

****