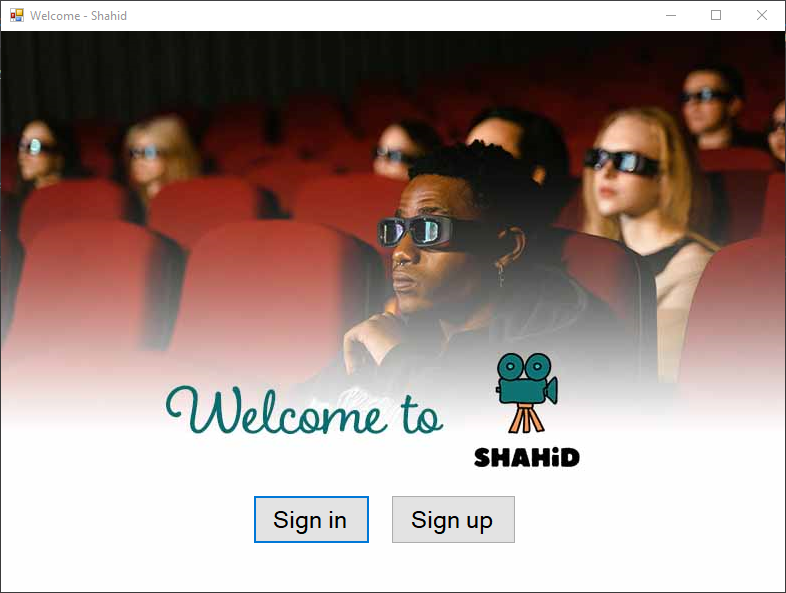
# **Shahid**

Movies and TV shows library and recommendation system

## Screenshots



## Guide

1. Run tables.sql to create necessary tables (SQL code listed below as well)
2. Run procedures.sql to create necessary procedures (SQL code listed below as well)
3. Run init.sql to populate the database with few initial data (SQL code listed below as well)
4. Use id:0 password: root to login as admin

## Create tables

CREATE TABLE User\_Info (

    UserID INT PRIMARY KEY,

    Username VARCHAR(255) NOT NULL,

    Email VARCHAR(255) NOT NULL,

    Password VARCHAR(255) NOT NULL,

    RegistrationDate DATE NOT NULL

);

CREATE TABLE Content (

    ContentID INT PRIMARY KEY,

    Title VARCHAR2(255) NOT NULL,

    con\_Description CLOB, -- Use CLOB for large text data

    Type\_content VARCHAR2(50),

    Tags VARCHAR2(255),

    ReleaseYear INT,

    Genre VARCHAR2(50),

    MeanRating DECIMAL(3,2)

);

CREATE TABLE Watchlist\_Info (

    WatchlistID INT NOT NULL,

    UserID INT,

    ContentID INT,

    FOREIGN KEY (UserID) REFERENCES User\_Info(UserID),

    FOREIGN KEY (ContentID) REFERENCES Content(ContentID)

);

CREATE TABLE UserInterests (

    UserID INT PRIMARY KEY,

    Genre VARCHAR2(255) NOT NULL,

    Weight INT

);

CREATE TABLE Rating (

    RatingID INT PRIMARY KEY,

    UserID INT,

    ContentID INT,

    Rating DECIMAL(3, 2),

    Review CLOB, -- Use CLOB for large text data

    Timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

    FOREIGN KEY (UserID) REFERENCES User\_Info(UserID),

    FOREIGN KEY (ContentID) REFERENCES Content(ContentID)

);

CREATE TABLE GenreTracking\_Info (

    TrackingID INT PRIMARY KEY,

    UserID INT,

    Genre VARCHAR2(50),

    VisitCount INT,

    FOREIGN KEY (UserID) REFERENCES User\_Info(UserID)

);

CREATE TABLE Recommendation (

    RecommendationID INT PRIMARY KEY,

    UserID INT,

    ContentID INT,

    RecommendationScore DECIMAL(5, 2),

    FOREIGN KEY (UserID) REFERENCES User\_Info(UserID),

    FOREIGN KEY (ContentID) REFERENCES Content(ContentID)

);

## Create procedures

CREATE OR REPLACE PROCEDURE Check\_User\_Info (

    p\_userid IN VARCHAR2,

    p\_password IN VARCHAR2,

    p\_message OUT VARCHAR2

) AS

    v\_count NUMBER;

    v\_username VARCHAR2(255);

BEGIN

    SELECT COUNT(\*) INTO v\_count

    FROM User\_Info

    WHERE UserID = p\_userid

    AND Password = p\_password;

    IF v\_count = 1 THEN

        SELECT Username INTO v\_username

        FROM User\_Info

        WHERE UserID = p\_userid;

        p\_message := 'Hello, ' || v\_username;

    ELSE

        p\_message := 'Wrong information please try again';

    END IF;

EXCEPTION

    WHEN OTHERS THEN

        p\_message := 'Error: ' || SQLERRM;

END;

CREATE OR REPLACE PROCEDURE RegisterUser(

    p\_Username IN VARCHAR2,

    p\_Email IN VARCHAR2,

    p\_Password IN VARCHAR2

)

IS

BEGIN

    -- Insert new user into User\_Info table with auto-generated UserID and current date for RegistrationDate

    INSERT INTO User\_Info (UserID, Username, Email, Password, RegistrationDate)

    VALUES (user\_id\_seq.NEXTVAL, p\_Username, p\_Email, p\_Password, SYSDATE);

    -- Display confirmation message

    DBMS\_OUTPUT.PUT\_LINE('Registration successful');

EXCEPTION

    WHEN DUP\_VAL\_ON\_INDEX THEN

        DBMS\_OUTPUT.PUT\_LINE('Error: Email address already registered');

    WHEN OTHERS THEN

     DBMS\_OUTPUT.PUT\_LINE('Error registering user: ' || SQLERRM); END;

CREATE OR REPLACE PROCEDURE AddContentListing(

    p\_Title IN VARCHAR2,

    p\_Description IN VARCHAR2,

    p\_Type IN VARCHAR2,

    p\_Tags IN VARCHAR2

)

IS

BEGIN

    INSERT INTO Content(Title, con\_Description, Type\_content , Tags)

    VALUES (p\_Title, p\_Description, p\_Type, p\_Tags);

    DBMS\_OUTPUT.PUT\_LINE('Content listing added successfully');

EXCEPTION

    WHEN OTHERS THEN

        DBMS\_OUTPUT.PUT\_LINE('Error adding content listing: ' || SQLERRM);

END;

CREATE OR REPLACE PROCEDURE SearchContent(

    p\_Title IN VARCHAR2,

    p\_Type IN VARCHAR2

)

IS

    -- Define variables to store search results

    v\_ContentID Content.ContentID%TYPE;

    v\_Title Content.Title%TYPE;

    v\_Description Content.con\_Description%TYPE;

    v\_Type Content.Type\_content%TYPE;

    v\_Tags Content.Tags%TYPE;

BEGIN

    -- Perform search based on user input and store results in variables

    SELECT ContentID, Title, con\_Description, Type\_content, Tags

    INTO v\_ContentID, v\_Title, v\_Description, v\_Type, v\_Tags

    FROM Content

    WHERE Title LIKE '%' || p\_Title || '%'

    AND Type\_content = p\_Type;

    -- Display search results

    IF v\_ContentID IS NOT NULL THEN

        DBMS\_OUTPUT.PUT\_LINE('Search results displayed:');

        DBMS\_OUTPUT.PUT\_LINE('Content ID: ' || v\_ContentID);

        DBMS\_OUTPUT.PUT\_LINE('Title: ' || v\_Title);

        DBMS\_OUTPUT.PUT\_LINE('Description: ' || v\_Description);

        DBMS\_OUTPUT.PUT\_LINE('Type: ' || v\_Type);

        DBMS\_OUTPUT.PUT\_LINE('Tags: ' || v\_Tags);

    ELSE

        DBMS\_OUTPUT.PUT\_LINE('No results found');

    END IF;

EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

        DBMS\_OUTPUT.PUT\_LINE('No results found');

    WHEN OTHERS THEN

        DBMS\_OUTPUT.PUT\_LINE('Error searching content: ' || SQLERRM);

END;

CREATE OR REPLACE PROCEDURE AddToWatchlist(

    p\_ContentID IN INT,

    p\_UserID IN INT

)

IS

    v\_UserCount INT;

    v\_ContentCount INT;

BEGIN

    -- Check if the user is logged in

    SELECT COUNT(\*)

    INTO v\_UserCount

    FROM User\_Info

    WHERE UserID = p\_UserID;

    IF v\_UserCount = 0 THEN

        RAISE\_APPLICATION\_ERROR(-20001, 'User not logged in.');

    END IF;

    -- Check if the content exists

    SELECT COUNT(\*)

    INTO v\_ContentCount

    FROM Content

    WHERE ContentID = p\_ContentID;

    IF v\_ContentCount = 0 THEN

        RAISE\_APPLICATION\_ERROR(-20002, 'Content not found.');

    END IF;

    -- Add content to user's watchlist

    INSERT INTO Watchlist\_Info (UserID, ContentID)

    VALUES (p\_UserID, p\_ContentID);

    COMMIT;

    DBMS\_OUTPUT.PUT\_LINE('Content added to watchlist successfully.');

EXCEPTION

    WHEN OTHERS THEN

        ROLLBACK;

        DBMS\_OUTPUT.PUT\_LINE('Error adding content to watchlist: ' || SQLERRM);

END;

CREATE OR REPLACE PROCEDURE TrackUserInterests(

    p\_UserID IN INT

)

IS

    v\_UserCount INT;

BEGIN

    -- Check if the user is logged in

    SELECT COUNT(\*)

    INTO v\_UserCount

    FROM User\_Info

    WHERE UserID = p\_UserID;

    IF v\_UserCount = 0 THEN

        RAISE\_APPLICATION\_ERROR(-20001, 'User not logged in.');

    END IF;

    -- Iterate through the user's watchlist to increment genre weights

    FOR v\_Genre IN (SELECT c.Genre

                    FROM Watchlist\_Info w

                    INNER JOIN Content c ON w.ContentID = c.ContentID

                    WHERE w.UserID = p\_UserID)

    LOOP

        -- Increment weight for the visited genre

        UPDATE UserInterests

        SET Weight = Weight + 1

        WHERE UserID = p\_UserID AND Genre = v\_Genre.Genre;

    END LOOP;

    COMMIT;

    DBMS\_OUTPUT.PUT\_LINE('User interests tracked successfully.');

EXCEPTION

    WHEN OTHERS THEN

        ROLLBACK;

        DBMS\_OUTPUT.PUT\_LINE('Error tracking user interests: ' || SQLERRM);

END;

CREATE OR REPLACE PROCEDURE SubmitReviewRating(

    p\_ContentID IN INT,

    p\_UserID IN INT,

    p\_Rating IN INT,

    p\_Review IN VARCHAR2

)

IS

    v\_UserExists INT;

BEGIN

    -- Check if the user is logged in

    SELECT COUNT(\*) INTO v\_UserExists

    FROM User\_Info

    WHERE UserID = p\_UserID;

    IF v\_UserExists = 0 THEN

        RAISE\_APPLICATION\_ERROR(-20001, 'User not logged in.');

    END IF;

    -- Insert new review and rating

    INSERT INTO Rating (ContentID, UserID, Rating, Review)

    VALUES (p\_ContentID, p\_UserID, p\_Rating, p\_Review);

    -- Recalculate mean rating for the content

    UPDATE Content

    SET MeanRating = (

        SELECT AVG(Rating) FROM Rating WHERE ContentID = p\_ContentID

    )

    WHERE ContentID = p\_ContentID;

    COMMIT;

    DBMS\_OUTPUT.PUT\_LINE('Rating and review submitted successfully.');

EXCEPTION

    WHEN OTHERS THEN

        ROLLBACK;

        DBMS\_OUTPUT.PUT\_LINE('Error submitting rating and review: ' || SQLERRM);

END;

CREATE OR REPLACE PROCEDURE GenerateRecommendations(

    p\_UserID IN INT

)

IS

    v\_UserGenre VARCHAR2(50);

    v\_Title Content.Title%TYPE;

    v\_Description Content.con\_Description%TYPE;

    v\_ReleaseYear Content.ReleaseYear%TYPE;

BEGIN

    -- Get the most visited genre of the user

    SELECT Genre INTO v\_UserGenre

    FROM UserInterests

    WHERE UserID = p\_UserID

    ORDER BY Weight DESC

    FETCH FIRST 1 ROW ONLY;

    -- Get recommended movies based on the user's most visited genre

    FOR rec IN (SELECT Title, con\_Description, ReleaseYear

                FROM Content

                WHERE Genre = v\_UserGenre

                ORDER BY ReleaseYear DESC)

    LOOP

        v\_Title := rec.Title;

        v\_Description := rec.con\_Description;

        v\_ReleaseYear := rec.ReleaseYear;

        -- Output the recommendations to the user

        DBMS\_OUTPUT.PUT\_LINE('Title: ' || v\_Title);

        DBMS\_OUTPUT.PUT\_LINE('Description: ' || v\_Description);

        DBMS\_OUTPUT.PUT\_LINE('Release Year: ' || v\_ReleaseYear);

    END LOOP;

    IF SQL%NOTFOUND THEN

        DBMS\_OUTPUT.PUT\_LINE('No recommendations found for User interests.');

    END IF;

EXCEPTION

    WHEN OTHERS THEN

        DBMS\_OUTPUT.PUT\_LINE('Error generating recommendations: ' || SQLERRM);

END;

## Init data

-- Create sequence and trigger to auto increment ContentID

CREATE SEQUENCE CONTENT\_ID;

create or replace TRIGGER new\_content

BEFORE INSERT ON Content

FOR EACH ROW

BEGIN

  SELECT CONTENT\_ID.NEXTVAL

  INTO   :new.CONTENTID

  FROM   dual;

END;

-- Insert initial users

INSERT INTO User\_Info (UserID, Username, Email, Password, RegistrationDate)

VALUES (0, 'admin', 'admin@shahid.com', 'root', SYSDATE);

INSERT INTO User\_Info (UserID, Username, Email, Password, RegistrationDate)

VALUES (1, 'test', 'test@shahid.com', 'test', SYSDATE);

-- Insert data into Content table

INSERT INTO Content (title, type\_content, con\_description, genre, tags, releaseyear)

VALUES ('Central Intelligence', 'movie', 'After he reconnects with an awkward pal from high school through Facebook, a mild-mannered accountant is lured into the world of international espionage.', 'Crime', 'Crime, Action, Comedy', 2016);

-- Insert data into Content table

INSERT INTO Content (title, type\_content, con\_description, genre, tags, releaseyear)

VALUES ('The Shawshank Redemption', 'movie', 'Over the course of several years, two convicts form a friendship, seeking consolation and, eventually, redemption through basic compassion.', 'Drama', 'Drama, Crime', 1994);

-- Insert data into Content table

INSERT INTO Content (title, type\_content, con\_description, genre, tags, releaseyear)

VALUES ('The office', 'tv show', 'A mockumentary on a group of typical office workers, where the workday consists of ego clashes, inappropriate behavior, and tedium.', 'Comedy', 'Comedy, Romance', 2005);

-- Insert data into Content table

INSERT INTO Content (title, type\_content, con\_description, genre, tags, releaseyear)

VALUES ('The Godfather', 'movie', 'The aging patriarch of an organized crime dynasty transfers control of his clandestine empire to his reluctant son.', 'Crime', 'Crime, Drama', 1972);

-- Insert data into Watchlist\_Info table

INSERT INTO Watchlist\_Info (WatchlistID, UserID, ContentID)

VALUES (1, 1, 1);

-- Insert data into Watchlist\_Info table

INSERT INTO Watchlist\_Info (WatchlistID, UserID, ContentID)

VALUES (1, 1, 2);