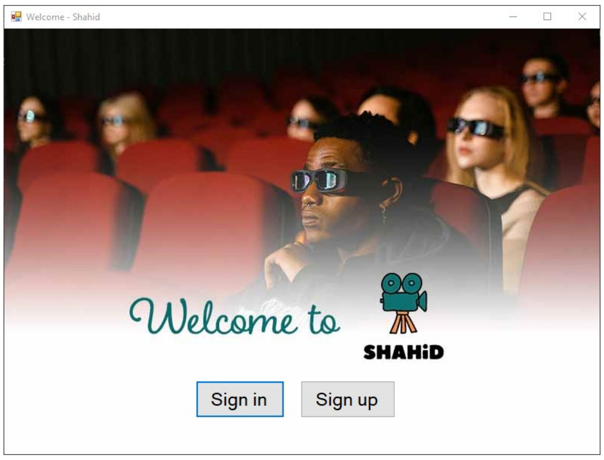
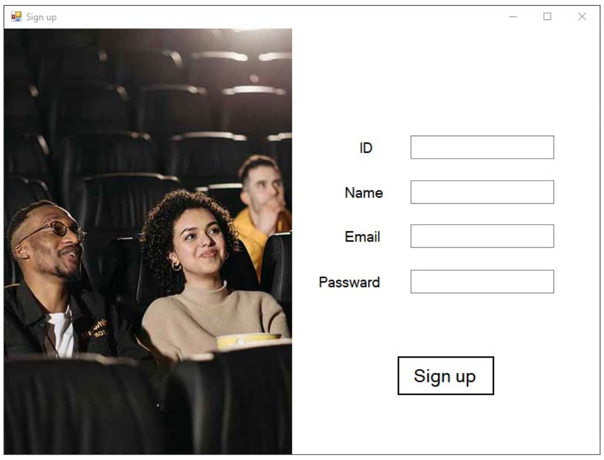
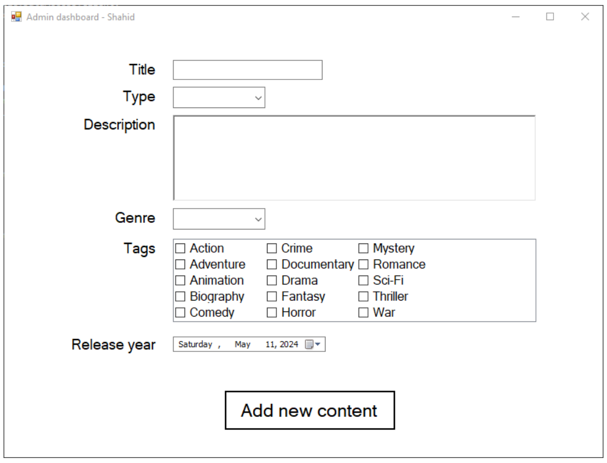
Screenshots







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

1. Create tables

```sql
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)
);
```

2. 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(
```

```sql
    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;
```

3. Init data

```sql
-- 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)
```

```sql
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);
```