Requirements

Project Name:
Just Another Movie

Team Members:
Hayden Richards, Kaitlyn Hardin, Emmett Storey, Robert Jones

Abstract:
Just Another Movie is an application in which the user will be prompted to enter information about what type of movies they like. Users will be able to search by different genres or different actors or actresses that they like. The application will take that information and using different classification and recommendation techniques the application will search through our database and will return a list of movies that the user should enjoy. Users can also remove movies from the list if they have either seen it before or do not like the movie

Tools and Technologies:

- Windows OS
- Visual Studio 2019
- IMDB database
- Discord communication
- Desktop application
- Data: Kaggle dataset
- Github
- C#, SQL

Requirements:

Frontend

1. User can load the startup Window
    1.1. Design of the application will be seen immediately
        1.1.1. Includes app logo
        1.1.2. Includes the app's color scheme
2. New window in application loads for searching
            2.1.1.1. User can search by inputting information about movie attributes
            2.1.1.2. User will put movie attribute information in input boxes for each attribute
                2.1.1.2.1. Input box for actors

2.1.1.2.2. Drop-down menu for year
2.1.1.2.3. Drop-down menu for genre
2.1.1.2.4. Input box for director
2.1.1.2.5. Input box for title

2.2. Error messages will be displayed to user for any invalid searches
- 2.2.1. Invalid searches for actors, director, and title input boxes
  - 2.2.1.1. System will return message, “no results found”, if the system cannot find the information in the database.

2.3. Results window

2.4. System outputs the recommended movie list on one page for the user.

2.5. User searching by the movie attributes
- 2.5.1. The system will output a movie list based on movie attributes inputted by the user.
  - 2.5.1.1. The user does not have to include information for all attributes before the system generates a movie list.
    - 2.5.1.1.1. All the fields for the movie attributes can be left empty.
      - 2.5.1.1.1.1. The system will output a generic/standard movie list based on user ratings.
- 2.5.2. The recommended movies are selected based on our selecting Model
  - 2.5.2.1. Model: ML.NET
    - 2.5.2.1.1. provided by our IDE: Visual Studio

3. Window for Movie List
   - 3.1. This window shows the list of movies
     - 3.1.1. The user can peruse through the list of movies using left and right arrows
   - 3.2. The window includes a back button
     - 3.2.1. This returns the user the previous window
   - 3.3. The user can delete a movie from the list
     - 3.3.1. Each movie will have an “X” button that means delete.
     - 3.3.2. The user will have a message window pop up to undo any recent deletions.
4. Expanded Movie Window
   - 4.1.1. The user can click on the movie to get more information about that movie.
   - 4.1.2. New window will pop once the movie is clicked
     - 4.1.2.1. Movie window will include more information about the movie
       - 4.1.2.1.1. Year
       - 4.1.2.1.2. Title
       - 4.1.2.1.3. Director
       - 4.1.2.1.4. Date
       - 4.1.2.1.5. Genre
       - 4.1.2.1.6. Cast and Crew
       - 4.1.2.1.7. Average ratings
     - 4.1.2.2. The user will be able to create a new movie list based on the movie.
       - 4.1.2.2.1. There will be a “create list” button for the user
       - 4.1.2.2.2. Movie list format will be the same
       - 4.1.2.2.3.

Backend

1. Make a connection to MS SQL Server
    1.1. Get connection string
        1.1.1. Create function to get the connection string
2. Get movie information from the user
    2.1. Get input based by actor.
    2.2. Get input based on year.
    2.3. Get input based on Genre.
    2.4. Get input based on director
    2.5. Get input based on Title.
        2.5.1. If input consists solely of common words such as: the, and, or it will be considered invalid.
    2.6. Pass the criteria to the ML algorithm
3. Return list of movies
    3.1. Returns list of movies that match search criteria involving multiple inputs.
    3.2. Returns an error message if invalid input is given in search criteria.
    3.3. Returns an empty list if no movie matches search criteria.
    3.4. Receive list from ML algorithm.

4. Output list of movies to frontend component
    4.1. User can remove movies from list
        4.1.1. User can click a button to remove a movie from a list
            4.1.1.1. The movie the user selected will be removed from the list
    4.2. User can traverse the movie list
        4.2.1. Users can click on a right arrow to scroll right on the list.
        4.2.2. Users can click on a left arrow to scroll left on the list.
    4.3. User can click on movie
        4.3.1. User can look at movie details
            4.3.1.1. User can see the year it was made
            4.3.1.2. User can see director
            4.3.1.3. User can see title
            4.3.1.4. User can see genre
            4.3.1.5. User can see Movie poster if available
        4.3.2. User can create a new movie list from movie
            4.3.2.1. Create a list of movies they want to watch

Database

1. Import .csv files from our movie dataset
    1.1. Online data source: Kaggle dataset
    1.2. Movie data files include:
        1.2.1. Credits

        - 1.2.2. Keywords
        - 1.2.3. Metadata
        - 1.2.4. User Rating
        - 1.2.5. User demographics

2. Store datasets in our chosen database server
    - 2.1. Database server: MS SQL Server
    - 2.2. Group data based on original files from the movie dataset
        - 2.2.1. Example: Movies.csv -> movies table
3. Prepare the data
    - 3.1. Sanitize data that will not be used for our project
        - 3.1.1. Create standard naming conventions
        - 3.1.2. Remove unnecessary data
            - 3.1.2.1. Timestamps
            - 3.1.2.2. Gender of the cast and crew
            - 3.1.2.3. Detailed information of the cast and crew of each movie
                - 3.1.2.3.1. Gender
                - 3.1.2.3.2. Job description
                - 3.1.2.3.3. Department
    - 3.2. Create relationships between tables
        - 3.2.1. For example: User ratings is related to movie titles.
            - 3.2.1.1. A connection will be made between all related data used in our application.

Machine Learning (ML.NET)

- 1.1. Load data into model
    - 1.1.1. Create the connection string to the SQL server
    - 1.1.2. Load data into the model from the SQL server
- 1.2. Build and train model
    - 1.2.1. Add the appropriate Nuget packages
        - 1.2.1.1. Microsoft.ML
        - 1.2.1.2. System.Data.SqlClient
        - 1.2.1.3. Microsoft.Extensions.Configuration
        - 1.2.1.4. Microsoft.Extensions.Configuration.Json
        - 1.2.1.5. Microsoft.Extensions.Configuration.FileExtensions
    - 1.2.2. Add a movie class to load the appropriate columns to train the algorithm
    - 1.2.3. Use matrix factorization to train the model
    - 1.2.4. Use collaborative filtering to find recommendations
- 1.3. Evaluate and test model
    - 1.3.1. Train model using data from SQL server
        - 1.3.1.1. Input test data into model
        - 1.3.1.2. Run the model
        - 1.3.1.3. Make adjustments if necessary
    - 1.3.2. Evaluate model

1.3.2.1. Check the output to see if the model works
1.3.2.2. Make adjustments if necessary
1.3.2.2.1. The ML.NET model builder provides an assessment to determine the algorithm that best fits our application.
1.4. Return list of recommendations to backend.
1.4.1. Pass user information into model
1.4.2. Generate recommendations
1.4.3. Send recommendations to frontend

Training the ML algorithm

2. Create the model for which the ML will be trained
2.1. The model will be trained from different user movie reviews
2.1.1. Load the test data from the SQL server
2.1.2. Use matrix factorization to create recommendations
2.2. Run the model
2.2.1. Run the model with the test data
2.2.2. Check output to see if is giving good recommendations
2.2.3. Make improvements
2.3. Evaluate model
2.3.1. Possible create multiple different generations if necessary
2.4. Improve the model
2.4.1. Make improvements so that the model can be more accurate with recommendations
2.5. Save the final model
2.5.1. Save the model when no more improvements can be made to it

Timeline

Week Of	Hayden	Emmett	Kaitlyn	Robert
9/5	System design and SQL Server			
9/12	Database			
9/19	Training AI Backend	Backend Design	GUI Design	
9/26		Backend Design		

10/3		Backend Class implementation		
10/10		Backend Class implementation		
10/17		Backend Class implementation	Start Window	
10/24			Search Window	
10/31			List Window	
11/7			Movie Window	
11/14				
11/21	Test			
11/28				
12/5	Prepare for Demo and Presentation			