Case Study for .Net Core

Match Details Management System

Create a C# console application with the below Mentioned class

I. MatchDetails Class

Create a MatchDetails class with the following properties:

- MatchId (int): A unique identifier for each match.
- Sport (string): The name of the sport for the match (e.g., "Soccer," "Basketball," "Cricket").
- MatchDateTime (DateTime): The date and time of the match.
- Location (string): The location or venue of the match.
- HomeTeam (string): The name of the home team.
- AwayTeam (string): The name of the away team.
- HomeTeamScore (int): The score of the home team.
- AwayTeamScore (int): The score of the away team.

II. MatchManagement class:

Create a class called MatchManagement

Create a program that performs the following operations using a List<MatchDetails>:

a. Add at least 6 MatchDetails objects to the list, representing different matches.

b. Display a list of all matches, including their details (e.g., match ID, sport, date, location, teams, scores).

c. Implement a method that takes a match ID as input and displays the details of the match with that ID.

d. Implement a method that allows users to update the scores of a match by providing the match ID.

e. Implement a method that allows users to remove a match from the list by providing the match ID.

III. Add the following features

a) Sorting Matches

Implement a feature that allows users to sort the list of matches based on criteria such as match date, sport, or location.

Allow users to choose the sorting order (ascending or descending).

b) Filtering Matches

Implement a filtering feature that allows users to filter matches based on specific criteria, such as sport, location, or date range.

Display the filtered matches in a separate list.

c) Statistics

Calculate and display statistics based on match scores, such as the average score, highest score, and lowest score for each team or sport.

Allow users to choose the specific statistics they want to view.

d) Search Matches

Implement a search feature that allows users to search for matches by entering keywords related to sports, team names, or locations.

Display the matches that match the search criteria.

IV. Validation Criteria

Check for the below validation criteria for the MatchDetails class properties:

i. MatchId:

Must be a positive integer.

Should be unique among all matches.

ii. Sport:

Should not be empty or null.

Should be a valid sports name (e.g., "Soccer," "Basketball," "Cricket").

Optional: You can also enforce a maximum length for the sport name.

iii. MatchDateTime:

Must be a future date and time.

Optional: You can add constraints like not allowing matches on weekends or after a certain date.

iv. Location:

Should not be empty or null.

Optional: You can enforce a maximum length for the location.

v. HomeTeam and AwayTeam:

Should not be empty or null.

Should be different teams (i.e., not the same name for both).

Optional: You can enforce a maximum length for team names.

vi. HomeTeamScore and AwayTeamScore:

Must be non-negative integers.

Implement these criteria as part of your validation logic when creating or updating a MatchDetails object in your C# program. Additionally, you can provide clear error messages to inform users about validation failures and guide them in correcting input data.

Program Class

From the Main method Provide a Menu to choose for the different functionalities.

Call the appropriate functionalities and display the result.