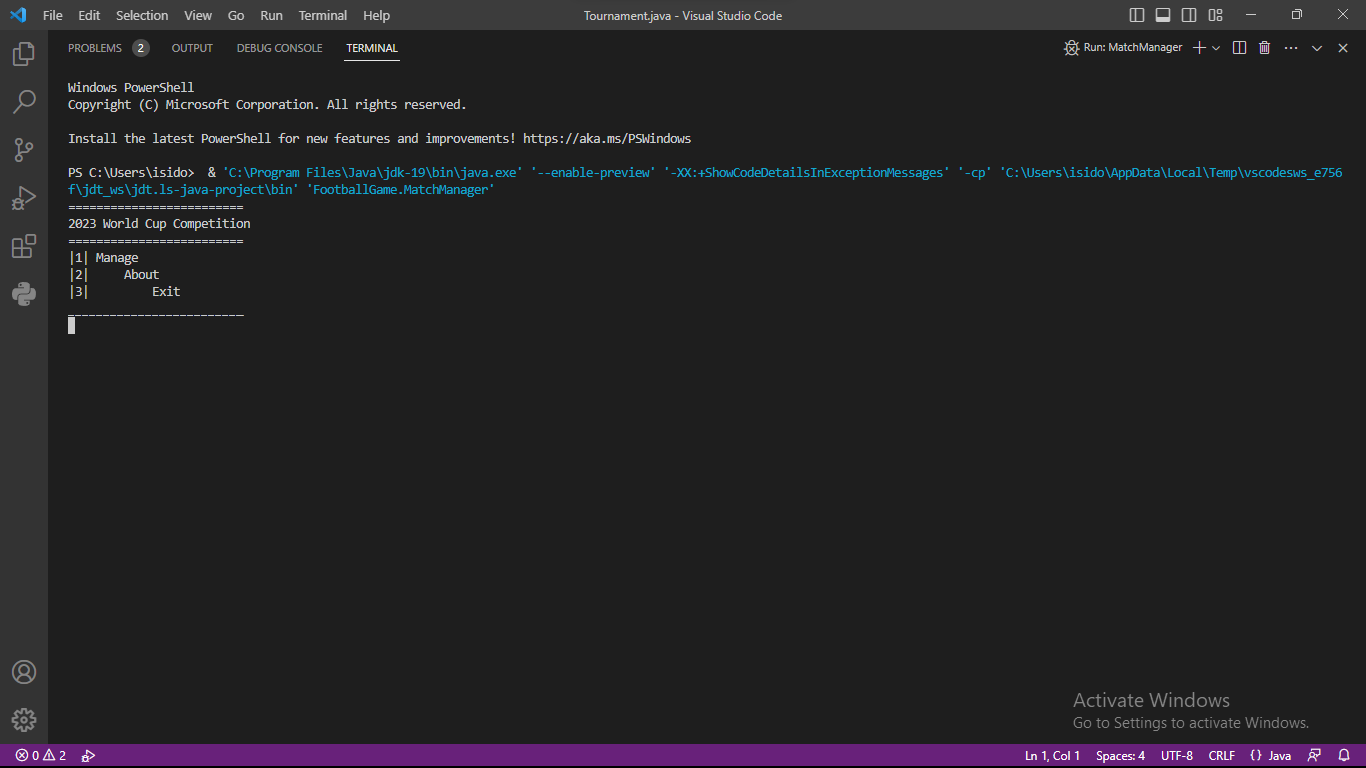
**Project Documentation**

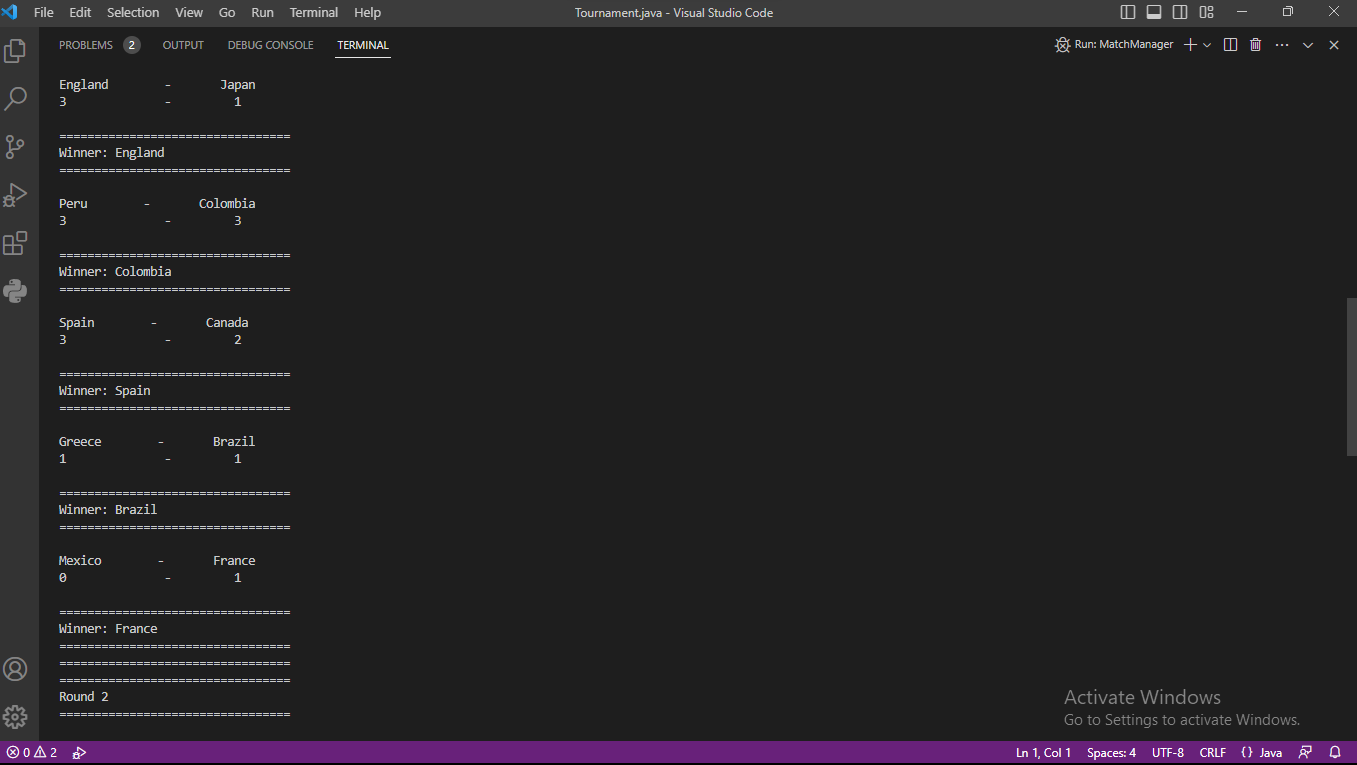
**Project Title: Football Tournament Simulation**

**1.1 Introduction:** A simple Java software application that simulates a football tournament for sixteen (16) teams. Similar to that of a World Cup competition, it has Round of 16 up to the finals. Sixteen countries around the world have a team represented at the tournament. Only the second round to finals were considered. Each game was set up to have a winner, each team can either win or lose, via direct win or loss. All scores and points are randomly generated. There are also penalty rounds for when the same numbers are generated for both teams. The 8 teams with the highest goals, and penalty kicks if done, goes on to the 2nd round, and the best 4 to the semi-finals and the best two to the finals. The team that attains the highest score in the final is the winner.

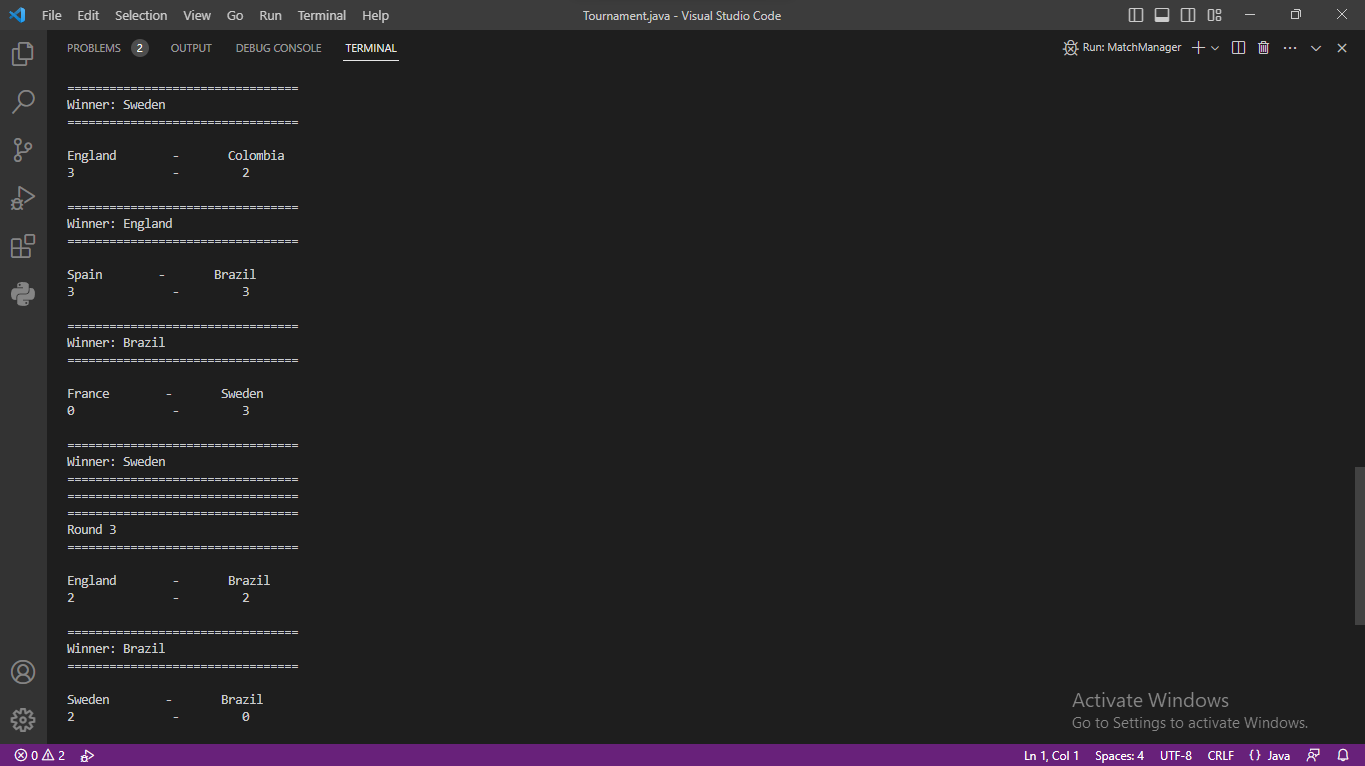
**1.2 Output:**

**Fig 1.1**

**A screenshot of a computer

Description automatically generated**

**Fig. 1.2**



**Fig. 1.3**

Graphical user interface, text

Description automatically generated

**Fig. 1.4**

**2. Development:** My code's simulation of the tournament starts with the round of sixteen and goes all the way to round two. My text file, which identified all the teams competing in the tournament, the rounds that would be played to decide the victor, and the statistics that would show the list of teams and their goals at various stages in the competition, was displayed after I first displayed my introduction page.

**3. Coding Concepts:** Throughout the entire process, I extensively relied on classes, arrays, queues, while loops, variables, and plenty access modifiers. Classes were used to manage the various aspects of the tournament, such as teams, games, and rounds. Each class had the appropriate methods and attributes to manage its specific responsibilities. Arrays were used to store the data for each team and keep track of their scores throughout the tournament. A sorting algorithm of choice was used to output the final list of participants sorted in order of tournament placing. The choice of the sorting algorithm depended on the size of the input data and the desired output. A queue structure was used to manage the tournament. This allowed for the easy addition and removal of teams as they advanced or got eliminated. Linked lists were used to manage the data for each team, such as their name, points, penalty wins, and goals scored. Linked lists were also used to manage the tournament bracket and ensure that each team was in the correct position.

**4. Challenges:** The tournament followed a specific structure, and the program needed to keep track of which teams had advanced to which round. This required careful management of data structures to ensure that each team was in the correct position in the tournament bracket. Implementing the scoring system was also a challenge. The program needed to track the goals scored by each team in each game and update the scores accordingly. Each game had a winner, and therefore, it was necessary to implement a solution to handle tiebreakers if necessary. Implementing the sorting algorithm was another challenge. The final list of participants needed to be sorted in order of tournament placing. Choosing an appropriate sorting algorithm and implementing it correctly was essential for the program to produce the correct output. Handling user input was also a challenge. The program needed to prompt the user when to start a round and wait for the appropriate input. The program also needed to handle any errors or invalid input from the user. Finally, handling random number generation was a challenge. The program needed to use the Math object to generate a random number to simulate the games. The random number generation needed to be implemented correctly to ensure that the games were fair and unbiased.