| Functionality | Purpose | Status |
| --- | --- | --- |
| General | Grammar and Spelling are correct on the screen | Did not encounter any problems. Done |
|  | Rounds all floating point numbers to one decimal place | * Forced a case that caused 20/3 which resulted in 6.66…   + Implemented a round() method to fix this problem   Done |
| User Experience | Has a clean, easy-to-use user interface | * Slowly tuned the user-friendly of the UI over the course of the weeks, based of the feedback from my peers   Done |
|  | Handles improper user input with an appropriate error message | * A try-catch caught all the non-number characters when asked for a number   + Implemented the getInput() method that handles all invalid inputs.   Done |
|  | Prints an appropriate message when a user gives a valid input | * Organization of roster and matches wasn’t clear as to what was being organized   + Fixed confusion by specifying what it was sorting   Done |
|  | A valid option leads to the correct menu | Did not encounter any problems. Done |
|  | Insert appropriate pauses allowing the user to digest the information displayed on the screen | * The program would move to fast and didn’t give the user time to read the screen   + Imported java.util.concurrent.TimeUnit to add pauses allowing the user to digest in the information on the screen   Done |
| Storing Data | Encapsulates appropriate fields | * Some attributes did not need the accessor methods so I removed them   Done |
|  | Includes all necessary statistics for all classes (Player, Coach, Match, and Playoff Match) | Did not encounter any problems. Done |
| Add records | Reading match files successfully create a Match/PlayoffMatch in the ArrayList | * Fixed the problem of match.csv files being inconsistent due to testing. The files would contain information that was out of order   + Saved a backup with all the original files * Cleaned up the function because it was very messy * Figred out a way to move the match to the trash once it was loaded   Done |
|  | Reading the roster file successfully fills the Roster ArrayList | * I realized this function could be used to add extra players/coaches to the roster   + Changed it so it can also be used to load a single player/coach   Done |
|  | Trading players successfully adds a player | * Didn’t know where the extra player would come from   + Added a folder called extraPlayers in which the user can pick who to obtain * Had a lot of trouble with moving the player to the trash but it turns out you just needed to refresh the project.   Done |
|  | Hiring coach successfully adds a coach | * Didn’t know where the extra coach would come from   + Added a folder called extraCoaches in which the user can pick who to hire   Done |
| Modify Records | Reading a match file successfully updates Player stats | * Moved this operation from the match class to the team class, a more appropriate location * Noticed that modifyStatistic and saveStatistics were very similar   + Tried to combine modifyStatistic and saveStatistics but I couldn’t figure something just stuck with 2 methods   Done |
|  | Reading a match file successfully updates Coach stats | * Moved this operation from the match class to the team class * Noticed that modifyWinPercentage and saveStatistics were very similar   + Tried to combine modifyWinPercentage and saveStatistics but I couldn’t figure something just stuck with 2 methods * Simplified this section of code   Done |
|  | Resetting statistic works successfully with a warning | * Reseting statistics is a dangerous move   + Added a warning before preforming the operation asking the user if they are sure about resetting all statistics.   Done |
| Delete Records | Trading players successfully deletes a player | Did not encounter any problems. Done |
|  | Successfully fire a coach from the team | Did not encounter any problems. Done |
| Search Records | Search Players based on their name | * I initially chose to seach players by name OR jersey number but scrapped it to a more general purpose method that searches for either a player or a coach based on their name   Done |
|  | Search Coach based on their name | Did not encounter any problems. Done |
| Sort Records | Sort Players based on points per game | * Had to figure out how the Comparable interfaced worked * Chose the sort by points per game instead of the other stats because it was arguably the most impotant   Done |
|  | Sort Coaches based on win percentage | * Had to figure out how the Comparable interfaced worked   Done |
|  | Sort Matches/PlayoffMatches based on point differential | * Had to figure out how the Comparable interfaced worked * Decided not to sort by absolute win percentage because then it wouldnt put the won matches at the top   Done |
| Reading Files | Reading all necessary attributes from a match file to create a Match/PlayoffMatch | Did not encounter any problems. Done |
|  | Reading all necessary statistics for each player on a match file to update the player’s overall stats | Did not encounter any problems. Done |
|  | Reading all necessary statistics from a match file to update the coach’s win percentage | Did not encounter any problems. Done |
|  | Reading all necessary statistics on the roster file to load the entire roster | Did not encounter any problems. Done |
| Writing Files | Successfully saving all necessary attributes of a roster member to the roster file | * Implemented the idea that a method can return the information of roster member separated by comma which was used here (listOutStats()) * Distinguished between players and coaches the roster list   Done |
| \*\*Extra Moving Files | Successfully move match, extraPlayer, and extraCoach files to the trash once they are loaded by the program | * When you added an extra player/coach to your roster you shouldnt be able to add them again   + Added a trash bin so that when you load a file you can’t load it again * Initially thought it didnt work but I just had to refrect the project from the disk. The side bar in IntelliJ is slow   Done |

Project Review

* Did it work or did it address the goals and objectives?
  + Yes, it works in terms of the goals that I set for this project. My goal was to create a tracking system for a basketball team that would update roster information based on match results.
* Did it work for some data sets, but not others?
  + Initially yes. I remember that updating player statistics based on a match result worked, but updating coach statistics did not work. I overcame this by writing out the calculations that the program was performing on paper, which allowed me to get a better understanding of how the program works and to fix the problem of coach statistics not updating properly.
* Does the program in its current form have any limitations?
  + The number of statistics that it stores/reads for each player and more detailed match statistics. I also personally don't like the user interface being the IntelliJ console. A better interface would make it program more user-friendly and more aesthetically pleasing.
* What additional features could the program have?
  + I could have given my program the ability to read a .csv directly from the official NBA team tracking website. In order to achieve that, I would need to overcome the limitation of adding all the statistics for a player.
* Was the initial design appropriate?
  + The initial design of my program was appropriate because it was simple. Storing player and coach data and updating the data based on match results was simple but challenging. I also included basic features that modify, add, and delete data sets, as well as sorting the data sets.