# Team Project

**Problem Description**

As you saw in the project teaser, your task will be to make a word-playing game.

You will develop a new word game inspired by games like: Scrabble, Wordle, Boggle, and Banana Grams. The basic rules of the game are as follows:

All players are provided the same set of "N" letters, and each player must generate as many words from those letters as possible in "S" seconds. At the end of "S" seconds, each player's answers are compared against a list of known words, and a summary score is created for each player. The player with the highest score will be ranked at the top of the leaderboard, 2nd highest score is 2nd, etc…

*Letter Sequence provided to all players*

| | | | | |
|---|---|---|---|---|
| M | N | A | D | D |

*Player 1 answers:*
MAD
MAN
DAD

*Player 2 answers:*
DAD
MAD

As a team, your job is to create a GUI/networking-based platform suitable for hosting tournament play. Tournament play allows a set of players (say 16 players) to participate in multiple rounds of play, and have their performance aggregated until a single winner is identified. The client applications should not store any data. All the words, state, scores, and users are stored in the server. The server will start a new tournament each day. Each tournament would have at least 5 rounds.

You must decide on the scoring method and tell the user how to play the game. User scores should be non-trivial. More points should be given to player who generates more complex word lists (see below for some hints about how to compute word list complexity).

It is up to your team whether you will allow users to keep playing in the tournament after missing a round or multiple rounds. The game needs to support multiple users playing in the tournament, and keep a sorted list of the players scores and the rounds completed. The current leaders of the tournament need to be displayed.

You must create a **FUN** game to play. You must add features to make the game more interesting.

Your project must minimally include:

1. **<u>Persistent Log</u>**: the backbone of your program will be persistent storage of players and games. The persistent log must allow for multi-day tournaments where the server can be shut down and restarted while preserving the current status of each player. The persistence log could be a csv/txt file, but we recommend implementing a SQL database.
2. **<u>Scoring</u>**: no competitive game would be complete without a scoring system to update player/character stats and dole out points to the players in real-time as the database is updated
3. **<u>Client/Server:</u>** A single server, and at least 3 clients (full credit requires at least 16 supported clients). The clients and servers will be played and tested on the Linux computer environments in the engineering building.
4. **<u>User Interface</u>:** At least 1 client interface. Full credit requires both a graphical user interface and a minimal command line client user interface. The minimal command line interface may only implement the most basic gameplay operations, where the graphical user interface should implement more advanced features.
5. **<u>Documentation:</u>** Your team project must contain at least 3 wiki pages: 1) A home page with a 1 paragraph summary of your project, at most 1 pictures, and links to the user and developer documentation; 2) User documentation that describes how to play your game (the judges will read the user documentation to figure out how to play the game). 3) The developer documentation must enumerate the SWD technologies used to develop the game (i.e. Junit testing, Inheritance, Polymorphism, JavaFX, Multithreading, etc) *AND* how those technologies were used in your solution.

Word complexity ideas:

- Correct words
- words using unusual letters or letter combinations
- words starting with unusual letters
- using all the letters in at least one word.
- Negative scores for short words
- Negative scores for words not in the wordlist
- No score for short words (i.e. less than 3 characters)

Possible Fun *Features*

- Add sound to play
- Add interesting graphical effects to display words
- Add wildcard characters in the provided letter sets where the players can use the wildcard as any letter they want. (either for the entire round or per word choice).
- Provide real-time feedback on per-word scoring
- Allow the number of characters provided to change per round of the tournament
- Allow players to submit a wordlist before the time ends for bonus scoring. (i.e., if a player submits the word list in 15 seconds for a 30-second round, perhaps that player gets to double their points).
- The provided characters may have interesting attributes. For example, some letters may be designated as “reusable” so that the letter may be used multiple times in the word.

Keep in mind, these are ***MINIMUM*** requirements. Correctly implementing your version of these features will only earn at most an ***85%*** grade. To earn more points, your team must add additional features to make your design more unique, functional, and interactive. At least three more ***fully functioning*** features will be needed to earn a 100% grade.

You will still complete Java Docs and Wiki for this project. In the developer documentation your team must explain which features you've added to get credit for their completion.

You have until the early morning of Monday December 5, 2022 at 5:00 am to complete this project. Best of luck!

**Schedule**

- This document released at 9:30am December 1, 2022
- Contest ends **Monday December 5, 2022 at 5:00am;** All code is committed on GitLab team project
- All high-level documentation due **Monday December 5, 2022 at 5:00am**
- Team contribution quiz Wednesday December 7th **11:59pm**
- During class – finalists announced Tuesday, December 6th **In class**
- During class – finalists' presentations Thursday, December 8th **In class**
- Final rankings quiz due Thursday, December 8th **11:59 pm**