Work Division:

UML Diagram (Raphael):
For the UML Diagrams, I just used my IntelliJ Ultimate to generate the UML Diagrams for Client and Server.

Activity Diagram (Raphael):
I went through how the game was going to be played and put in which system would handle it, whether it was client or server. I checked what I had drawn out with the rest of the team to make sure everything looked correct and it was how the game was played.

WireFraming (Raphael):
I created what I had in my mind for this project and how it would look. I drew it out and then checked with the whole team of how the GUI would look and see if it was simple enough to follow/make.

Server Logic (Raphael, Wayne):
Wayne: The files that were used were called: WordGuessServer.java, GameCommunicator.java and GameTracker.java were the inner workings of the server. These dealt with setting up the categories and checking the player's guess against the word that was chosen randomly. The way the server and client communicate is through GameCommunicator. This class has important variables that hold the state of the game and information about a particular client. The way the server is able to identify a particular client is by using GameTracker. This class has variables the server needs to do to determine if the player's guess is valid or not, having enough guesses left, whether they have enough attempts left for that particular category, and if they were able to guess the word or not.
Raphael: Provided test cases

Client Logic (Raphael, Wayne):
Raphael: The files that were used were called: WordGuessClient.java and GameCommunicator.java inner workings for the client. These dealt with taking in the player's input and adding it onto the GameCommunicator so that the Server could handle the actual gameplay.
Wayne: Provided test cases

Server UI (Kai):
Kai:The file was called GUIServer.java. It created some server basic GUI elements with Port and IP address on the first start scene. Function isNumber used regular expression to let port's number correctly. On the main server scene, a listview was created to show GameTrack and communicator information.

Client UI (Jeremiah):
Jeremiah: The client was created using the traditional JavaFX nodes and a HashMap of Scenes. There are 4 scenes and each represents a different moment in the game, such as the login scene, gameplay scene, etc. Each scene is constructed with a method. Inside each method, the nodes are instantiated and given event listeners. These were connected to the server in order to actually play the game.

Collaboration:
Our group worked together on a Discord server. We exchanged messages and gave each other updates whenever we added files to the Github repository. We also made sure we all agreed on a decision being made.

What worked, what did not:

- **Worked:**
    - Teamwork and Communication
    - Program runs smoothly and pretty cleanly
- **Didn't work:**
    - Couldn't figure a way to clear clients from the client ArrayList