What are we calling our project?
Java Uno or just Uno

How will we communicate?
Numbers have been exchanged and a discord server has been set up. Google hangouts will also be used as screen share software if discord's software will not work.

Who is the lead (and if multiple, for which parts)?
Chris - Game Logic
John - Client
Brenden - Testing & Debugging.

How will we track tasks and bugs in the interface, if it is being coded by the group? Versioning?
Github

Who has the final say?
Equal say

If there are problems, how do we resolve them? Who is our outside (non-peer) arbitration?
We will resolve issues internally as a group effort. if needed the professor will be our outside arbitration

What are the code standards for the interface, how are they enforced?
We will follow best coding practice with relevant naming conventions, encapsulation, clean code, frequent commits, relevant commit comments.

From a user perspective:

- User will join a game and receive the starting amount of cards.
- User will wait for their turn. When It is their turn, they receive a message informing them to play.
- The user will have the option to play a card from their hand, if able, or draw a card from the deck.
- If the drawn card is able to be played, it will allow the player to do so.
- If the user gets down to one card left, a UNO button will be available to press.
- When a user plays their last card, a win message will be displayed to the winner, and a lose message to the losers (or do they continue playing?).
- After the game is over, the user can choose to play again, or exit.

Interface details:

- We will have a message handler class that will parse input messages from the server and output messages from the client.
- Sending messages can be on UI thread, receiving messages must be a seperate thread to not block UI thread.
- Client will handle all of the messages from server and then hand them off to the game OR client will handle then for the game.
- We would like to have the game instance on the server so there will be one deck instance on the server. Then we will pop off the card when it is drawn
- Server will pass out what cards each of the clients have in their deck.

*Note: Without seeing server code, it's hard to judge how comprehensive the interface is that interacts with the server.