Name:

OVERVIEW

1.
Project Background and Description

To create a fully functioning blackjack game played against the dealer. With the basic game of blackjack, the scope would be for the dealer to hit with anything under 17, while the player can hit or stand whenever wanted. Whoever would be closest to 21 without going over would win. With the same score, the code would present a “push” restarting the round.

2.
Project Scope

- Game Logic and Rules
  - Implement basic rules of blackjack
  - Implement scoring system and determine winner
- User Interface
  - Develop and interface which shows or explains the players hand and dealers
  - Provide buttons or commands for player actions
  - Display messages to communicate the events
- Player Interactions
  - Allows players to place their bets
  - Handle players inputs and their actions
- Dealer interactions
  - Reveal the dealer's card when appropriate
  - To ensure dealer follows the logic of the game
- Testing and debugging
  - Test to see if the game is running as expected
  - Debug any issues that arise

3.
High-Level Requirements

- The ability for the player to register
- track wins and losses and display to the user
- The ability to track the dealers and or other players hands

4.
Implementation Plan

Github URL:
https://github.com/HuntSean17/SYST17796-Group-7-Final-Project/upload/main
This will be our team's remote repository called "Group 7 repository, it will include a single file that contains our Java codes and everyone will be capable of accessing and modifying that specific file. There will also be a UML diagram directory or file that will contain the code's UML diagram created using the Visual paradigm.

5.
Design Considerations

- Encapsulation
  - The first example of encapsulation is the player class and all it's variables primarily the name variable
  - Encapsulation will be used again when it is time to implement the player score tracking as well as the dealers and players hands during gameplay
- Delegation
  - The best example of delegation is the method that will be used to generate the deck at the start of playing
  - One example that already exists in the code is the shuffle method it will be used in the playing of the game nearly every time but is stored as its own method
- flexibility/maintainability
  - A good example of this would be what will become the play method which would allow you to change aspects of the game such as number of players without changing any of the other classes
  - The whole of Game.java is also very useful for flexibility for the same reason the play method will be as you have access to many key features but not to their composition so you can change things without causing issues