Take Home Project: Blackjack

Hey there,

The team is excited to move onto the next portion of the interview with you! This part of the interview will be a take-home assignment, where you'll showcase your process and approach when it comes to working on a realish-world feature - **a small React project implementing a simple version of Blackjack**. This project **should take about 2-4 hours** – we know your time is valuable, so if takes any longer than that, then cut it short and just show us what you've got.

This project will show us **how you work with APIs and how you structure your React code**. When you come on-site, we'll spend one of the interview sessions collaboratively expanding upon your work, so write it with some extensibility in mind.

Now, onto Blackjack. If you're not familiar with the game, here are the **simplified rules we will be going by for this project:**

1. The game consists of two players: You vs The House (the computer), where the goal is to beat the The House's hand, without going over 21
2. A card contains a "point" value equivalent to it's number (the 3 of club is worth 3 points…the 9 of spades is worth 9 points…etc etc). Face cards (Jack, Queen, King) are worth TEN points, and the Ace card is either worth 1 or 11, whichever is most helpful for the player's hand. For example:
    a. If the player has a Jack and a Queen, and then draws an Ace, the Ace will be worth 1 point to add up to 21
    b. If the player has a Queen and an Ace, the Ace will be worth 11 points to add up to 21
    c. If the player has a 2 and an Ace, the Ace will be worth 11 points to get closer to 21
    d. If the player has a 2 and a 5, and then draws an Ace, the Ace will be worth 11 points to get closer to 21. If the player then draws a 10, the Ace will now be worth 1 point
3. The House is initially dealt TWO face up cards and no more! **This isn't part of the regular rules for Blackjack, but it is for us. In other words, the House will always only have 2 cards.**
4. You are also initially dealt two face up cards, but you have one of the following options:
    a. Hit: You are dealt one more card to add to your point value. For this project, the player may hit as many times as they like, until their card value exceeds 21, at which point the game ends in **an automatic loss**
    b. Stand: Ends the round (for the purposes of this project, this will end the game)
5. Once you end the round, the game is over, and there should be a display of whether you won or lost
    a. You win if:
        i. The House's total is > 21 and your total is < 21 (for the purposes of this project, you can ignore this condition, since the House will only have two cards and cannot get a total > 21)
        ii. Your current total is < 21 but higher than the House's total
        iii. Your current total is 21 and the House's total is not 21
    b. You lose if:
        i. Your current total totals over 21 (don't forget to factor in the different edge cases of the Ace card!)
        ii. You current total is < 21 but lower than the House's total
        iii. You tie with the House

To implement this, you'll use this API for card management: http://deckofcardsapi.com/. You should be able to initialize one deck and deal out cards from the deck using this API.

What we are looking for:

1. Setting up a React project. You are welcome to use create-react-app.
    a. Show us judgment in how you structure and decompose your code. Remember to optimize as much for readability as well as performance. Good abstractions and indirections = extensibility without over-engineering.
    b. Show us good understanding of React fundamentals and best practices
2. Interacting with the API at http://deckofcardsapi.com/
3. A reasonable implementation of a basic Blackjack game according to the above rules
4. Cards played should be displayed with their respective images (the API provides image urls)
5. There should be a simple display of "You Win" or "You Lose" when the round has ended

Bonus points:

- Tests! We've provided plenty of rules to follow, which may be a good fit for your test cases 🙂
- Styling, but only if you have time and you would like to!
- Speed - we don't want you to rush your implementation, and we don't expect you to get around to every single little detail. But the time constraint will show us what you choose to focus on when you need to get scrappy

Although this may take between 2-4 hours, this trivial project really shows us a good reflection of how you write code. Once you come on-site we'll definitely expand upon it!

Another note, there are obviously various online projects of Blackjack and again, this isn't really an assessment of giving the right or wrong answer, but an opportunity to showcase your thought process and highlight your strengths. We hope you have fun with the assignment, and please let us know if you have any questions. Please send back a GitHub repo with your solution - we look forward to seeing what you put together!