Frontend Software Engineer - Code Challenge

Introduction & Purpose

Candidates for this role should be able to comprehend, analyze and deliver according to the requirements specified in the imaginary scenario described below. They should approach this challenge as if they were part of the company's engineering team.
The purpose of this test is to identify the candidate's methodology/train of thought and their technical skillset to design, architect, implement and deliver a solution, **even if it is not 100% complete**. While we do not expect all candidates to be able to finish the challenge within the time frame given below, **we would still want to go through their code.**

Scenario's Description

The company wants to release a new application for the gaming industry and the engineering team has been asked to design and create the game following modern standards. The application has to be maintainable and easily extendable, as the product will be subject to new features and requirements as the community grows.

Requirements

The product team has come up with a list of requirements and use cases that the minimum viable product should satisfy. The suggested product is a simple risk-taking browser game (web app).

The Minimum Viable Product (MVP) must include the following pages and functionality:

1. **Landing Page**

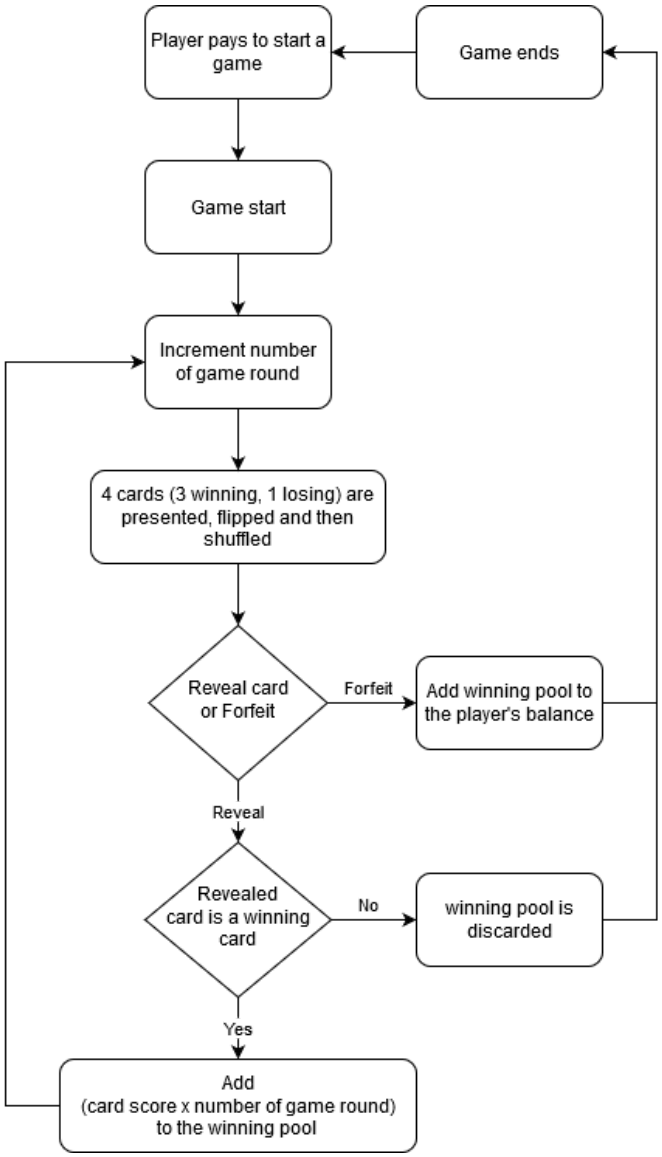
   The initial app page, where the player can choose a nickname and proceed to play the game. This page could also contain additional information, including the game's logo, a how-to guide or game tips. The selected nickname should be persisted across the player's session, along with the player's balance (this can simply default to a fixed value for the purpose of this challenge).

   Minimum required page functionality:

   - Type and submit a nickname
   - Create a session that includes the player's nickname and his balance
   - Navigate to the game page

2. Game Page

The product team has designed a simple turn-based decision-making game with the following description. For each game cycle, the player starts with an initial balance and has to spend a small amount from his balance to initiate a game. Then the player goes through consequent rounds where he is presented with four hidden cards. Only three are winning cards, so the player is given the option to either take the risk and reveal a card or forfeit. If the player chooses to reveal a card and wins, the card's score is multiplied by the round number and added to the winning pool for that game cycle. If the player reveals the losing card, the winning pool is discarded and the game cycle ends. On the other hand, if the player chooses to forfeit, the winning pool is added to his balance and the game cycle ends. Winning cards have a fixed score value and are stacked in a deck, from which 3 random cards are drawn each round and are shuffled back in after the round ends. The winning pool should be cleared at the start of a game cycle.



Minimum required page functionality:

- Have a place to view player's nickname and current balance
- Start a new game by subtracting a fixed amount from the player's balance
- Run mentioned game loop
- Logout and navigate back to the landing page, destroying the player's session

Nice-to-have features and functionality

The product team has shared some optional requirements that could further improve the player's experience with the game and would be nice to have.

- Have an event log showing player and game activity, as well as any player balance changes, added to the game page. This does not have to be persistent.

- Include transitions and animation sequences to the game flow.

- Automatically return to the landing page, if the player's balance drops below the minimum amount required to start a game. This would be nice to do after informing the player with some sort of an intuitive message.

- Propose technical simplifications and improvements in regards to the product design, as well as identify and correct any found pitfalls.

Delivery

Candidates should address all requirements mentioned in the *Requirements* section above, utilizing the **ReactJS** framework. Having code in **TypeScript** is highly preferred, but not mandatory. Candidates are free to use any development environment or tools they see fit, as long as instructions on how to run the app are provided. Additional features should be tackled in a way that helps to further demonstrate their skills and expertise.

Candidates are free to include any technical documentation and should include their code in a git repository. The deliverable should be the repository's URL.

Time Frame

Candidates should share their deliverables, via email, in 3 days (*before 12:00 pm of the 3rd day*), starting from the day they received the test/challenge.