Full-Stack Javascript Developer Test

1. Information

This test is for Skivori to assess your approach to problem-solving and development skills. The Back-End needs to be developed using *Node.js* or *NestJS* while the Front-End should be developed in any modern Front-End Framework/Library (e.g. *ReactJs*, *Angular*, *Vue*, *etc.*).

The project needs to contain a *README* file with instructions.

2. General Requirements

- Use *Node.js/NestJS*.
- Validate the request body.
- Organise folder structure.
- Comment your code.
- Deploy both the Front-end and Back-end using *Azure, AWS, GCP*, or any cloud service.
- Apply a mobile-first approach.
- Use a common UI library for the Front-End (e.g. *Tailwind, Bootstrap, Ant Design, etc.*).
- Ensure that at least two different HTTP methods are used (e.g. GET, POST).
- Use of an AI code editor/assistant is recommended.

2.1. Questions

2.1.1. Question 1

Create a page that displays a list of all the games provided in the *game-data.json* file. Use the *thumb.url* property to display the game thumbnails. The games must be loaded from the Back-End using a REST API endpoint.

2.1.2. Question 2

Create a search functionality.

- When the user types into the search bar, the game list should update accordingly.
- The games must be filtered on the back-end using a REST API endpoint.

2.1.3. Question 3

Consider a Slot machine defined like this:

- *Reel 1: [“cherry”, "lemon”, “apple”, "lemon”, “banana”, “banana”, "lemon”, "lemon”]*
- *Reel 2: ["lemon”, “apple”, "lemon”, “lemon”, “cherry”, “apple”, "banana”, "lemon"]*
- *Reel 3: ["lemon”, “apple”, "lemon”, “apple”, “cherry”, “lemon”, "banana”, "lemon"]*

The user starts with 20 coins. Each spin will cost the user 1 coin.

Rewards

- 3 cherries in a row: 50 coins

- 2 cherries in a row: 40 coins
- 3 apples in a row: 20 coins
- 2 apples in a row: 10 coins
- 3 bananas in a row: 15 coins
- 2 bananas in a row: 5 coins
- 3 lemons in a row: 3 coins

Note: A match is only valid if it is in order from left to right.
For example:

- Apple, Cherry, Apple – **no win**
- Apple, Apple, Cherry – **win**

Create an endpoint that returns the result of the spin and the coins the player won/lost. Additionally, the user's balance should be updated after every spin.

Note: The balance and spins should not be stored in a database. Return the updated balance in the response from the used endpoint.

2.1.4. Question 4

In the REST API endpoints, use any middleware you deem necessary to make the endpoints more robust. Clearly mark the code that addresses this.

2.1.5. Question 5

Given that the search functionality filters on every keystroke, the back-end performance is starting to degrade. Find ways to reduce the number of hits

and/or optimise the search endpoint. Mark clearly the code that solves this issue.

2.1.6. Question 6

Add a button that converts the current balance to another currency for display purposes only. The button should simply convert the balance using an external API, such as https://www.exchangerate-api.com/docs/standard-requests.

2.1.7. Question 7

Use the following description to design a schema for a database that stores this information:

You are working on an online casino platform. A casino has games. Each game has a unique type. Each game is available in one or more countries where players are allowed to play. A player may or may not have a favourite game. Every spin on any game should be recorded, including the amount of money won or lost.

Create the schema and write the SQL statements to create the database and tables.

2.1.8. Question 8

If an AI Code Editor/Assistant has been used, give a detailed description of how and where it was used in the README file.