

Instructions

This is a competency test for the Full Stack Software Developer position at Bizcuit Solution. The goal for the test is to determine the way you solve problems and competency with relevant tech.

There are 3 questions in this test, the first is to check you basic logic and reasoning. The second is a simple backend api using NodeJS. The third is a simple react app that uses your api.

You can submit your solutions by emailing the link to your public Github (or any vendor) repo. You get bonus points for having a clean and organized repo.

There are some bonus segment marked with a 🌟. These are not required but are good to have. Only implement these when you are done with main requirements.

You are free to use any resource you want. Good luck 🍀


Question 1: Basic coding competency check

Create a function that takes 2 alphanumeric strings and returns a sum of the two string as a string. Example

```
function sum(a, b) {  
    // your logic goes here  
}
```

```
const result = sum("12", "2");  
console.log(result, typeof result); // Should print 14 string
```

Throw error whenever the arguments to your function doesn't match the requirements. Example

```
console.log(sum("12", "abc")); //  Error
```

Your code should be written in **Nodejs**. Do not use any external packages for this solution unless you are using it to work on the bonus segment of the solution.

🌟 Bonus Segment

You can get bonus points by implementing the following extra requirements but these are not requirements.

- Unit Tests
- CLI to accept the arguments for the sum function and display error
- Written in Typescript

Remember, you can use any npm packages to assist you with these requirements.

Question 2. NodeJS Competency Test

Build a beer api using NodeJS with 2 endpoints. You will need to have a database, *ideally MongoDB but MySQL or PostgreSQL is also ok*.

You are free to use any npm package that you would want. You can use ExpressJS or HapiJS or FeatherJS or Fastify or whatever.

Endpoint 1: Random Beer

This is a GET endpoint that returns a random beer as a JSON response. No arguments accepted. The path for this endpoint should be **/beer/random**.

You'll be replicating this endpoint, https://random-data-api.com/api/beer/random_beer. The result should look like this,

```
{
  _id: "abcd1234",
  uid: "b72fcca0-0e3c-4af8-957d-140c884a68b7",
  brand: "Dos Equis",
  name: "Schneider Aventinus",
  style: "Light Lager",
  hop: "Warrior",
  yeast: "1007 - German Ale",
  malts: "Caramel",
  ibu: "57 IBU",
  alcohol: "9.5%",
  blg: "8.2°Blg",
  randomCount: 0
}
```

with status code = 200.

The api should also persist the count for the number of times a beer was returned by this endpoint. Eg: If beer *abcde123* was returned by this random api increment the field *randomCount* by 1, so value for *randomCount* becomes 1 (0 + 1).

Endpoint 1: Create Beer

This is a POST endpoint that returns a success or fail payload based on the success of the insert status. The path for this endpoint should be **/beer**.

The argument is a JSON object that looks like this,

```
{
  uid: "b72fcca0-0e3c-4af8-957d-140c884a68b7",
  brand: "Dos Equis",
  name: "Schneider Aventinus",
  style: "Light Lager",
  hop: "Warrior",
  yeast: "1007 - German Ale",
  malts: "Caramel",
  ibu: "57 IBU",
  alcohol: "9.5%",
}
```

```
    blog: "8.2°Blog"  
}
```

All fields are required so api will reject if any field is missing. The api should return a success or failed string.

🌟 **Bonus Segment**

1. Use docker-compose to start up your app in localhost,
2. Use the micro-service methodology to design your api,
3. Duplicate record validation on *uid* field in create endpoint,
4. Initialize the database with some mock data using docker-compose
5. Written in Typescript

Question 3. React Competency Test

Build a react app that calls an api and renders a random beer. You can use create-react-app or gatsby or any other starter template.

The api to use is the one you built in Question 2.

The requirements for this app are,

1. Display all fields from the api in your UI,
2. Have a "Next" button to fetch a new random beer and display data on the UI,
3. Initial render when there is no beer should show a loading icon,

🌟 Bonus Segment

You can get bonus points by implementing the following extra requirements but these are not requirements.

1. Back button to show previous random beer. Disable this button if current beer is the first beer,
2. Pretty UI,
3. Mobile Friendly,
4. Dark Mode / Light mode toggle,
5. Comments for your code,
6. Doc for your component Library
7. Written in Typescript

Try to design your components in a way where they are easily reusable. You are free to use bootstrap or burma. Please do not use any existing component libraries, a major part of this test is to see how you design your components.