



FULL-STACK DEVELOPER TEST

Question 1: Front-end React and CSS Component Design

Description:

You are required to develop a React component that represents a user profile card. This card should display the user's name, profile picture, and a short bio. The card should also have a button that, when clicked, toggles the display of the user's email.

Requirements:

- Create a React component called UserProfile.
- Use CSS to style the component. Ensure it's responsive to various device sizes.
- The component should accept the following props:
 - name: User's name
 - imageSrc: The URL of the user's profile image
 - bio: A short biography of the user
 - email: User's email address.

Sample Input:

```
<UserProfile  
  name="John Doe"  
  imageSrc="https://example.com/john.jpg"  
  bio="A passionate full-stack developer with over 3 years of experience."  
  email="john.doe@example.com"  
>
```

Expected Output:

A styled user profile card displaying the given data. When the button is clicked, the email toggles its display.

Ritik Bhardwaj
ribharscode@gmail.com



Question 2: Backend Endpoint with NodeJS, TypeScript, MongoDB, and Redis

Description:

Design a RESTful API endpoint to fetch user data from a MongoDB collection and cache the result in Redis for faster subsequent fetches.

Requirements:

- Setup a basic NodeJS application with TypeScript.
- Connect to a MongoDB database and create a model User with fields: name, email, bio, and profilePicture.
- Implement a GET endpoint `/api/users/:id` that:
 - Fetches the user data from MongoDB by the given id.
 - If the user data is already cached in Redis, fetch from Redis instead.
 - Caches the result in Redis if fetched from MongoDB.

Test Case:

- Add a sample user to MongoDB:
 - ```
{
 name: "Jane Doe",
 email: "jane.doe@example.com",
 bio: "Experienced backend developer.",
 profilePicture: "https://example.com/jane.jpg"
}
```
- On the first GET request to `/api/users/[Jane's ID]`, the server should fetch from MongoDB.
- On subsequent requests, the server should fetch from Redis until the cache expires.



## Question 3: Front-end React and CSS Component Design

### Description:

Write a detailed step-by-step guide on how you would deploy the NodeJS application developed in Question 2 to AWS.

### Requirements:

- Detail the AWS services you would choose and why.
- Provide steps on setting up the environment, including any required configuration.
- Describe how you would monitor the application's health and performance.

### Expected Output:

A written guide detailing the steps to deploy the NodeJS application to AWS, including justifications for chosen AWS services and configurations. The guide should also cover application monitoring.



## Submission Guidelines:

Submit your code as a ZIP file. Ensure the zip file includes a README with instructions on how to run your application, how you have used each library in the task, and their purpose.

## Evaluation Criteria:

- Adherence to coding standards.
- The efficiency of the solution.
- Correctness of the implementation based on the sample input and expected output.
- Proper handling of edge cases.
- Clarity of the AWS deployment guide.

Note: Don't worry about copyright. If you are not selected, this task is yours to showcase on your GitHub forever.