# NODE JS – ASSIGNMENT

## Question 1

Create a portal where the employees can post their resumes and apply for the available opening positions in the company. Also the project managers can add an opening for the project.

A position needs to have the following values:

* Project Name – String - required
* Client Name – String - required
* Technologies – Array of string - required
* Role – Enum – required
* Job Description – String - required
* Status (open/closed) – Bit – required

Status will determine if the position is visible to the employees of the company or not. A project manager will update the position state when the position is no longer available.

The portal will support 2 kinds of users:

1. Project Managers
2. Employees

If an employee is interested in a position, he/she can apply for the same.

For simplicity, the following needs to be implemented:

1. Create a web server using express framework.

2. Add the following users in the database:

1. Project Managers - Add 2 users in the database (Id, Name,UserName,Password, Role)
2. Employees – Add 5 users in the database (Id, Name,UserName,Password, Role)
3. Expose the following REST APIs. Make sure you use the correct HTTP verbs to implement the below APIs

* **See a list of available openings in the company**

This API fetches the high level details of each opening (Project Name, Role, Technology) and returns them as a result.

* **Check the details of any opening (using a valid identifier)**

This API fetches all the details relating to a specific opening. Return relevant status code for success and errors.

* **Apply for the opening**

This API allows the employees to show interest in a specific opening. Apply with the help of a userId of an employee already present in the database. Save the entry in the database. Once applied, same user cannot apply again for the opening and relevant message should be displayed in the response.

When an employee applies for a position, a notification should be sent to the recruiter. To imitate an opening, you can imitate it by logging onto the console. Explore custom events to print this notification.

* **Add an opening**

This API allows the recruiter to add a job opening. This should add an entry in database. Make sure to implement relevant validations to facilitate this change. In case of an invalid ID return relevant error code with description

* **Delete an opening**

This API allows the recruiter to delete his/her added opening. This should delete an entry in the database.

* **Update an opening**

This API allows the recruiter to update the opening, if the opening position is closed, a notification should be sent to the employee who has shown an interest. To imitate the update, you can imitate it by logging onto the console. Explore custom events to print this notification.

* **Check Applications**

This API will return the applicant information, return a list of userIds for the userIds which have applied.

* **Login**

The user needs to login using a password and userId present in the database. The API should return a valid token on successful authentication which can be used for further requests.

* **Upload Resume**

Create an API which allows a user to upload a resume (file) to the system.

There can only 1 resume per employee. Any subsequent uploads should result in overwriting the previous resume files.

Any uploaded files shall go in a specified directory on the server and the entry for the corresponding user should go in the database.

* **Download Resume**

Create an API which allows the logged in user to download his resume using his userId.

1. Make use of middlewares for the below items
   1. Validations
   2. Errors
   3. Authorization
2. Create an authentication mechanism using PassportJS and JWT strategy.
3. Authorize the following APIs
   1. Create an Opening – Manager of the opening
   2. Delete an Opening – Manager of the opening
   3. Update an Opening – Manager of the opening
   4. Check Applications – Manager of the opening
   5. Apply for Opening – Any Employee
   6. Available Openings – Employee + Manager (not open for all)
   7. Details of an opening -Employee + Manager (not open for all)
   8. Upload resume – Employee with current session
4. Every API should return relevant status code and messages in both success and errors.
5. Create a Unit Testing Framework using JEST (or any other framework of your choice) and testing the above mentioned APIs Create 2 tests per API (1 Positive + 1 Negative).
6. Use best practices for code structuring.
7. Use relevant schema in Mongo DB and host them over any free cloud service (e.g : <https://mlab.com/>) and specify the connection string in the solution.

## Question 2

Create a sample webpage which changes colors and shows the timer on screen.

1. The color should be changed every 5 seconds.
2. The counter needs to be implemented at the backend and the information needs to be sent using Socket.IO.
3. If multiple pages are open using the same address (http://localhost :3000/sample). Each page should change colors simultaneously i.e use the same counter implemented at the backend.
4. Use best practices for code structuring

**Upload the source code in a GitHub or Bitbucket repository and share the link.**

**Note that commit message names, code cleanliness in GitHub or Bitbucket repo factors into marking scheme. Also make sure you won’t copy the code from internet. Any sort of plagiarism found will lead to zero marks in assignment**