

**Node js experience task**



Prepared by: Sumeet / 29-11-2023

| **Use :** Node js & express js  **Database :** mongo/ sql/ postgres (Any one of this)  **What do we expect ? :**   1. As an experienced candidate, we anticipate a well-organized folder structure, meticulously defined routes, an API structure, and api validation using tools like Joi. 2. We expect to follow the specified schema for tables and accurate request responses as mentioned in the document. Expecting to do much with queries that use manual work to create output. 3. Handle every possible error and return error response with suitable status code. 4. If you are working with mongo then use of populate prohibited. 5. We expect join(in sql) & lookup(mongodb) |
| --- |

**All the best !! feel free to ask if you have any doubt**

Missing any point and use of AI like chat gpt may diminish your likelihood of being selected.

**Task 1: project & employee management system**

**Tables**

1. **Department**

| id | departmentId | name | createdOn |
| --- | --- | --- | --- |
| 1 | DEPT001 | john | dd/mm/yyyy |

1. **Employees**

| id | employeeId | fNAme | lName | departmentID | onBoardDate | age |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | EMP001 | john | robert | 2 | dd/mm/yyyy | 45 |

1. **Projects**

| id | projectId | name | startedOn |
| --- | --- | --- | --- |
| 1 | PROJ001 | john | dd/mm/yyyy |

1. **Employee\_project\_track**

| id | projectId | employeeId | joined | exit |
| --- | --- | --- | --- | --- |
| 1 | PROJ001 | john | dd/mm/yyyy | dd/mm/yyyy |

Join and exit indicate date employee joined that project and exit from that project

* Develop an Employee Management System that stores information about **employees**, **departments**, and **projects** and **projects\_employees**.
* Each project is associated with a department as well. Employees can work on multiple projects, and each project may have multiple employees.
* Employee, department and projects schema can have basic details as per above.

Develop below api operation on above data using join/lookup. We are expecting results in minimum queries.

**Basic important api**

* Create department static db (No api needed)
* Create project static db (No api needed)
* Create api to add employees with department **: -** While creating employees generate unique if for every employee as per onboarding. e.g for 1st employee it's EMP001 then for second EMP002 and in incremental order (no random number allowed should exactly look like EMP + 3 digit (001) ).
* Create api to add project wise employee tracking.

**Other data retrieval api**

1. Retrieve a list of all employees with their details (including the department and projects they are associated with).

In same api we can pass below filter

**employeeId : -** pass employeeId e.g. EMP001

**searchquery : -** this value can be anything and need to search in employeeName, employeeId, departmentName, project Name

**Response : -** [{

employeeId : EMP001

employeeName: john robert

departmentId : DPT001

departmentName : IT department

currentlyWorkingProject : {

projectId : PROJ001

projectName : WAYA project

}

}]

1. Get a list of employees in a specific department.

**Request : -** pass departmentId

**Response : -** [{

departmentId : DPT001

departmentName : IT department

employeeId : EMP001

employeeName: john robert

}]

1. Get a list of employees currently working on a specific project

**Request : -** pass projectId

**Response : -** [{

employeeId : EMP001

employeeName: john robert

projectId : PROJ001

projectName : “WAYA”,

WorkingFrom : “The day he started working on this project”

}]

1. Get a overall list of employees working and worked on a specific project

**Request : -** pass projectId

**Response : -** [{

employeeId : EMP001

employeeName: john robert

projectId : PROJ001

projectName : “WAYA”,

WorkingFrom : “The day he started working on this project”

}]

1. Get a list of employees working in a specific period of project.

**Request : -** Start date & end Date & projectId

**Response : -** Same as above (Point 4)

1. Get average age of all departments

**Response : -** [{

departmentId : DPT001

departmentName : IT department

avg\_age : 55

}]

1. Delete an employee from the system. delete will be soft delete and that entry we should not able to see in any kind of listing in above api’s

**Task 2: File-upload**

Accept file from postman and store in folder of repo without using any external file upload library.