

Assessment Criteria

What we would like to see:

- Ability to code based on given requirements
- Code Functionality and Readability
- Best practices in terms of Folder structure and reusability of components
- [Front-End] Reduce and use your own custom style even when using the CSS framework
- [Front-End] Ability to use different React libraries such as Redux, Router, Redux Form
- [Front-End] Ability to use Redux State management

Submission

You may upload the technical test to Github and provide us the link, or you may zip it up and send it over via email. Please ensure that your code can compile and is able to run. You should also provide a short Readme on how to compile and run the code.

[Optional]

Dockerize everything above and provide instructions on operating it

All the best!

Backend Technical Assessment

Develop a program to complete the following tasks using either

1. Nodejs on node 18.x or above
2. Python 3.X
3. .NET 6.x and above
4. MySQL, SQLite or MongoDB

Task

Instructions

Create a database of your own choice (either MySQL or MongoDB) where the program can connect to. You can design the database structure as you deem fit, but it needs to be able to handle the following.

1. Employee data

Key	Type	Description
id	Required	Unique employee identifier in the format 'UIXXXXXXX' where the X is replaced with alphanumeric
name	Required	Name of the employee
email_address	Required	Email address of the employee. Follows the typical email address format.
phone_number	Required	Phone number of the employee. Starts with either 9 or 8 and have 8 digits.
gender	Required	Gender of the employee (Male/Female)

2. Café data

Key	Type	Description
name	Required	Name of the cafe
description	Required	A short description of the cafe
logo	Optional to implement	Logo of the café. This will be used to display a logo image on the front-end.
location	Required	Location of the cafe
id	Required	UUID

3. Which employee work for which café, and the employee start date
4. No same employee can work in 2 cafes (this constraint can be handled either within the code or database)

- Create a GET endpoint `/cafes?location=<location>`
The response of this endpoint should be the below and sorted by the highest number of employees first

If a valid location is provided, it will filter the list to return only cafes that is within the area

If an invalid location is provided, it should return an empty list

If no location is provided, it should list down all cafes

Key	Description
name	Name of the cafe
description	A short description of the cafe
employees	Number of the employees. It must be an integer
logo <i>(optional)</i>	Logo of the café. This will be used to display a logo image on the front-end.
location	Location of the cafe
id	UUID

- Create a GET endpoint `/employees?cafe=<café>`
The response of this endpoint should be the below and sorted by the highest number of days worked. It should list all the employees.

If a café is provided, it should list down only employees that belong to that café.

Key	Description
id	Unique employee identifier in the format 'UIXXXXXXX' where the X is replaced with alpha numeric
name	Name of the employee
email_address	Email address of the employee.
phone_number	Phone number of the employee.
email_address	Email address of the employee.
days_worked	Number of days the employee worked It must be an integer and is derived from the current date minus the start date of the employee in the cafe
cafe	Café's name that the employee is under [leave blank if not assigned yet]

- Create a POST endpoint `/cafe`
This should create a new café in the database.
- Create a POST endpoint `/employee`
This should create a new employee in the database.
This should also create the relationship between an employee and a café.
- Create a PUT endpoint `/cafe`
This should update the details of an existing café in the database.
- Create a PUT endpoint `/employee`
This should update the details of an existing employee in the database.
This should also update the relationship between an existing employee and a café.

- Create a DELETE endpoint `/cafe`
This should delete an existing café in the database. It should also delete all employees under the deleted cafe
- Create a DELETE endpoint `/employee`
This should delete an existing employee in the database.

You should also provide the seed data for the database design you have designed.

Frontend Technical Assessment

Use **React JS** as the JavaScript framework.

Use **React-router** for views management

Use **Redux** for state-management

[Bonus] **Redux-saga** for side effect management

Use **Aggrid** for table

Use [Material-UI](#) or [Antd](#) as the CSS framework

Make use of **Create React App** cli to create a simple “Café Employee” manager that reads from the **Backend Program** created earlier.

Task

Instructions

Create a Web application where it has 2 pages. Cafes and Employees.

Café Page:

- This should call the `GET /cafes` endpoint.
- Display a list of cafes in a table:
 - Logo
 - Name
 - Description
 - Employees
 - Location
 - **Edit/Delete** button on each row at the end of the row.
- There should be a way to filter the list via the location.
- Upon clicking on the employees, it should go to the employee page and show a list of the employees under the café.
- There should be an “Add New Café” button on the page

Employee Page:

- This should call the `GET /employees` endpoint.
- Display a list of employees in a table:
 - Employee id
 - Name
 - Email address
 - Phone number
 - Days worked in the café
 - Café name
 - **Edit/Delete** button on each row at the end of the row.
- There should be an “Add New Employee” button on the page

Add/Edit Café Page:

- A form with the following:
 - Name [Reusable Textbox] *Minimum 6 character and max 10 characters validation.*
 - Description [Reusable Textbox] *Max 256 chars validation*
 - Logo [File] *Max 2mb validation*
 - Location [Reusable Textbox]
 - Submit [Button] which will call `POST/café` or `PUT/café`
 - Cancel [Button] which will bring you back to café page.
- *[Optional] Warn user if there are unsaved changes on the page before navigating away.*
- If user enters this page via **Edit**, the form should be prefilled with the café's information

Add/Edit Employee Page:

- A form with the following:
 - Name [Reusable Textbox] *Minimum 6 character and max 10 characters validation.*
 - Email address [Reusable Textbox] *email validation.*
 - Phone number: *SG phone number validation (starts with 8 or 9, and have 8 digits).*
 - Gender [Radio Button Group]
 - Assigned Café (optional) (Dropdown box)
 - Submit [Button] which will call `POST/employee` or `PUT/employee`, and assign the employee to a café (if selected).
 - Cancel [Button] which will bring you back to employee page.
- *[Optional] Warn user if there are unsaved changes on the page before navigating away.*
- If user enters this page via **Edit**, the form should be prefilled with the employee's information

The **Delete** button on all pages will have a confirmation pop-up message to proceed with the deletion. After the deletion, the page should refresh to reflect the deletion.