PSEUDOCODE

ADMIN

1. **Login as admin(29/12/2022)**

- Fetch input data from request.
- Validate email and password

  email - required, must be email
  password – required
- if validation failed ->return error message
- checking record existing in database if not return error message
- if exists check the email and password
- if email and password matches direct to admin dashboard that is consists of list of jobs , adding jobs, list of applicants
- if email and password doesn’t match then return invalid creditials

2. **Make Addjob api(for editing and adding the jobs)(29/12/2022)**

3. **check the id) if the id is null :**

(a) form according to job table

validate each fields

- job_title: required ,must be string
- job_description:required , must be string
- skills: required, must be string
- job_location: required, must be a string
- experiance: required, integer
- expiry: required, date

(b) if the validation failed return error message

(c) if the validation success, form submission, insert the data into table

4. **check the id) not null:**

(a) display form filled with data entries according to id then edit the fields:

(b) if the validation failed return error message

(c) if the validation success, form submission, update the data into table in accordance of the id

5. **ListJob** api(pagination)**(30/12/2022)**

- Fetch the data from the table and store it in a variable
- declare limit, offset , count : find the total number of records in the job table and validate eah of the value using symfony validator
- Pass the records as an array to front end according to limit and store it in a variable list
- Return the list , count to ajax call response

FrontEnd

- ajax call to the **ListJob** api to get the contents to display
- Calulate the number of item from the response and numberofpages using ceil function
- Function: to add buttons according to the number of items.
- Function : to load the contents according to each buttons.

6. Edit : making ajax call and pass id and pass it to the api **Addjob** if the id is given set the record details.

7. delete: ajax call to **Deletejob** api by passing the idand remove data from the table(**02/01/2023)**

- Ajax call when button is clicked
- Pass id and calling to a Deletejob api
- Deletejob : deleting the entries from the table

8. List the Applicants:**ListApplication Api(02/01/2023)**

- Fetch the data from the applicant table table and store it in a variable
- pass the variable as an array to frontend
- declare limit, offset , count : find the total number of records in the job table and validate eah of the value using symfony validator

- Pass the records as an array to front end according to limit and store it in a variable list
- Return the list , count to ajax call response

9. View api: to view the resume also(pass id with a ajax call to Viewapi)(**03/01/2023)**

- find the resume by using the id from public folder
- get the contents and display it.

10. Unit testing (04/01/2023)