****

**React & Express JS exercise**

In the following exercise you will be required to create a job candidates app using [React JS](https://reactjs.org/docs/getting-started.html) (creating single page application) & [Express JS](https://expressjs.com/) (creating the API server).

The app will have 4 screens –

Signin - containing a login form or an option to go the signup page.

Signup - containing a signup form.

Candidates (home page) - Display in low level all the candidates from the system DB.

Candidate profile – Display the full details of a chosen candidate.

* **Please pay attention** to the structure, organization, validations, error handling, code reusability & scalability.

Your state of mind should be as if these apps are going to expand massively and include a lot more features in the future.

**Server (Express JS)**

* You are provided with the express\_server folder that contains a very initial express server app, and an [SQLite](https://www.sqlite.org/index.html) DB (the file is located inside ./src/db) that contains two tables – user & candidate.

You will use these tables to perform 3 API calls (use SQLite Data Browser to see the DB structure & data).

To run the app:

Open the folder in your terminal

Run the *npm install* command.

Run the *npm run start* command.

* You need to add 3 API call handlers
  + **Sign up**

Route: /api/auth/signup (POST)

Input: {

username

email

password

}

Output: {

success: true

token

} (in case of success)

{

success: false

msg

} (in case of failure)

Logic:

* You need to add the user to the DB user table.
* Hash the password (using the installed [bcryptjs](https://www.npmjs.com/package/bcryptjs) library) before saving it in the DB.
* Generate a [JWT](https://jwt.io/) token for the user (using the installed [jsonwebtoken](https://www.npmjs.com/package/jsonwebtoken) library) and send that token in the success response.
  + **Sign in**

Route: /api/auth/signin (POST)

Input: {

username

password

}

Output: {

success: true

token

} (in case of success)

{

success: false

msg

} (in case of failure)

Logic:

* You need to find the user in the DB user table.
* Validate the password (using the installed [bcryptjs](https://www.npmjs.com/package/bcryptjs) library).
* Generate a [JWT](https://jwt.io/) token for the user (using the installed [jsonwebtoken](https://www.npmjs.com/package/jsonwebtoken) library) and send that token in the success response.

o **Get Candidates**

Route: /api/candidates (GET)

Input: none (only need to send the token in the header)

Output: {

success: true

candidates:[

{

id

first\_name

last\_name

email

gender

job\_title

job\_description

avatar

},

]

} (in case of success)

{

success: false

msg

} (in case of failure)

Logic:

* You need to return an array of all the candidates in the candidate DB table (the file already contains 100 candidates).
* Before returning the candidates, you need to validate the provided JWT token from the client.

**Client (React JS)**

* You are provided with the react\_app folder that contains a very initial react app (created using create-react-app).

To run the app:

Open the folder in your terminal

Run the *npm install* command.

Run the *npm start* command.

* You need to create a single page application that have 4 screens:
  + **Sign up**

Contain a sign up form for registering users to the system.

The form will include: username, email & password.

Use the signup API call you created in the server to send the data and get the JWT token.

* + **Sign in**

Contain a sign in form for login users to the system.

The form will include: username & password.

Use the signin API call you created in the server to send the data and get the JWT token.

* + **Candidates**

Once you got a JWT token from the server you will redirect the user to the home page that will display the list of available candidates.

The candidates data will be fetched from the server using the candidates API call you created (make sure to send the JWT token in the request header).

In the list you will display the candidate's avatar, full name & job title.

You will also have a 'full details' button that will redirect you to the following screen:

* + **Candidate profile**

Here you will display the full details we got from the candidates API call of the requested candidate.

* You can add features, add validations and design the app as you see wish.

You are welcome to add anything that we didn't mention, that will improve the user experience.

Feel free to contact us if you have any questions.

GOOD LUCK!