Assignment_ID: assignment_category_0003

Introduction

Attention web developers! We're on the lookout for a talented individual to build a user-friendly job-seeking website. As the developer, you'll be responsible for creating a platform where users can post job listings, search for positions, and apply seamlessly. If you're passionate about web development and ready to take on this exciting project, we want to hear from you!

-On-Site Job
-Remote Job
-Hybrid
-Part-Time

The website will show the above job types. For example, If a user clicks on the Remote Jobs Tab, he will see only remote jobs. [Explained in Task No. 5]

Make sure your website design is unique. Visit ThemeForest, Dribble, google, etc., to get some ideas.

You can explore component libraries other than DaisyUI. Remember, a unique project will add more value to your portfolio.

Main Requirements

Key Rules:

- Include a minimum of 18 notable GitHub commits on the client side
- Include a minimum of 8 notable GitHub commits on the server side
- Add a meaningful readme.md file with the name of your website and a live site URL. Include a minimum of five bullet points to feature your website. Write your selected category's name here.
- Make it responsive for all devices. You need to make it responsive for mobile, tablet, and desktop views.
- After reloading the page of a private route, the user should not be redirected to the login page.
- Use the Environment variable to hide the Firebase config keys and Mongodb credentials.
- Don’t use any Lorem ipsum text; you can not use the default alert to show any error or success message.

1. **Focus on making the website visually appealing. Ensure that**

- Color contrast is pleasing to the eye
- The website has proper alignment and space
- If needed, customise the design of any component you are taking from any component library. (For example, if you are using Daisy UI & have taken a card component from Daisy if needed, customise the styling of the card to make it reasonable rather than just copy & paste it.)

Note: Your website can not be related to your previous assignments' layout/ design or any practice project shown in the course modules or our conceptual sessions. Ex: You can't copy any design or similar functionality/ layout of

- Car Doctor
- Any of your previous assignments or conceptual session projects.

****If any similarities are found, you will get zero(0) as a penalty.****

2. Make sure to keep the navbar and footer on all the pages **except on the 404 page**. Create a reasonable and meaningful footer. (including website logo, name, copyright, some contact information, social media links, address, etc.)

3. Your website should have a navbar with the **Website logo, Website name, Home, All Jobs, Applied Jobs, Add A Job, My Jobs(Jobs that a user has created through Add A Job page), Blogs, and User Profile.**

Note: The **User profile picture, Add A Job, My Jobs, and Applied Jobs** on the navbar are conditional based on login. If the user is logged in, the navbar will show the profile picture; otherwise, it will show the **Login button**. If the username is available, the user's name will be visible when the mouse hovers over the profile picture.

4. **Login & Registration systems:** On the Registration and Login pages, display relevant error messages when necessary.

Login Page: When a user clicks on the login button, they will be redirected to the login page having the following:

- Email/Password

- Google Sign-in
- A link that will redirect to the registration page

Registration Page: The Registration page will have the Email/Password form having the following fields:

- Name
- Email
- Password
- Photo URL

Note: Do not enforce the email verification method, as it will inconvenience the examiner. If you want, you can add email verification after receiving the assignment result.

5. **Home page:**

- **Banner section** - A slider/banner/ a meaningful section. Inside the banner, there will be a Heading Title, a Short Description

Job by category - Implement a tab system for the **Job by Category** section. Explore [React-tabs](https://www.npmjs.com/package/react-tabs), or you can implement your tab system. There will be a minimum of 4 tabs. Each tab will show the job type mentioned above. There will be a tab named All Jobs, if the user clicks here, he will see all category jobs. Implement the Active Tab feature.

Each tab will have a minimum 2 jobs of in that category, and each job card will have the following information:

- Name who posted the job,
- Job Title
- Job Posting Date
- Application Deadline
- Salary range
- Job Applicants Number
- View Details Button

Note: When a user is not logged in and if he/ she clicks on the View Details button, notify the user with a message **You have to log in first to view details** by using a toast/ notification/ anything. Also, redirect him/ her to the login page. With a login, you can visit the single job details page.

Extra Section: Add two relevant sections. Try to make them attractive.

6. **Blogs page**: Create a Blog where you will have to answer the following questions:

- What is an access token and refresh token? How do they work and where should we store them on the client side?
- What is express js? What is Nest JS (google it)?
- Explain your code.

7. **All Jobs page:** Create an All Jobs page where you will see the jobs all the users have added in the tabular form. Each row of the All Jobs table/list will have the following information:

- Job Title
- Job Posting Date
- Application Deadline
- Salary range
- Details Button

Implement a search system on this page, based on the Job Title

Without logging in, if a user clicks on the **View Details button**, they will be redirected to the Login Page. Make sure to take the user to the Details Page after the user successfully logs in.

8. Single Job details route will be a private route:
After clicking on the **View Details** button, he/ she will be redirected to the **Job Details** route **(/job/:id)** containing the information **(job Banner, job title, description, salary range, Number of Applicants, Apply Button)**

Apply Button: Clicking on the apply button, a modal will open. Where logged in user name and email will be auto-filled, and an input field for the submit resume link

will be provided. There will be a submit application button, on clicking this button the application will be submitted and saved in a MongoDB collection & Job Applicants Number will be increased in MongoDB for that job.
Note: Don't let the applicant apply for a job if the deadline is over. Also, don't let the employer apply for his job.
Clue: Use Date. now() to get the current date and then compare it with the deadline. You can implement it on the front end.

9. **Add A Job page will be a private route(only logged in users can add jobs):**

Create an **Add A Job** page where there will be a form having the following fields:

- Picture URL of the Job Banner,
- Job Title,
- Logged In User Name and email (read-only)
- Job Category (For example: On Site, Remote, Part-Time, Hybrid)
- Salary range,
- Job Description,
- Job Posting Date
- Application Deadline Use the [**Date Picker Package**]
- Job Applicants Number(by default it will be 0)

NOTE: Make sure that if a user applies for a job the applicant number gets updated. Explore about MongoDB $inc operator.

10. **My Jobs page will be private routes:** If a user logs in, he/she will see the My Jobs page, which will show all the job information they have added from the Add A Job page in a tabular form. Each row will have an update and delete button so that users can update and delete jobs they posted.

- **Update Action** - If they click the `update` button, they can update the Job information.

Note: you can show the update form in a modal or another route.

- **Delete Action** - If they click the delete button, the Job will be removed from the list. Before the delete, ask for a delete confirmation.

Note: If users log in, they will only see the jobs they have added. The user cannot see the jobs other users added.

11. The **Applied Jobs page** will be a private route: If a user logs in, they will see the Applied Jobs page, which will show all the job information they have applied from the Apply Modal in the Job Details page.

Implement a filter system on this page, based on the **Job Category**. You can add a select field, on change of the field value, data in the table will change according to the selected value.

Note: If users log in, they will only see the jobs they have applied for. The user cannot see the jobs other users applied for.

12. For all the CRUD operations, show relevant toast/ notification/ anything with a meaningful message

13. 404 page: Create a 404 page. Add any interesting jpg/ gif on the 404 page. **Do not add header & footer on this page.** Just add a jpg/ gif & a **Back to Home** button. **The Back to Home** button will redirect the user to the home page.

Challenging Task -

1. **Implement Tanstack query:** Instead of the loader and use effect use Tenstack query for data fetching in all pages on your projects

2. **Implement a Theme Toggling** Button for changing themes from light to dark & dark to light. Changing the theme it will make the full system dark / light based on user interaction.

3. **Implement Framer Motion Package,** Use the Framer Motion on the home page (at least in one section), Using Framer Motion on other pages is optional.

4. **Implement JWT for any 2 routes.**

5. Implement **React-to-Pdf** (In the applied job page, keep a download summary button. On clicking this button a pdf will be downloaded from that page.) Explore it by following this link

<u>Optional (But Highly Recommended)</u>
<u>Implement any two tasks from the following optional list:</u>

1. Add a spinner when the data is in a loading state. You can add a gif/jpg, use any package, or customize it using CSS.

2. Use Swiper JS for the banner and slider.

3. Use the React Hook Form for handling any form.

4. Add one extra feature of your own. This will help you in the future to differentiate your project from others.

6. Implement a Dynamic Website Title. The website title will be changed according to the route you are clicking. Suppose your website name is PHero. Then, on the 'about' route, your website title will be 'PHero | About us'.

7. Implement Pagination on the "All Jobs page"

8. Make the component name, folder structure, and route name meaningful. If needed, add comments.

9. Implement a Search Input Field with a button in the banner on the job title, this search result will be shown below the slider/banner

Additional information:
1. You can host images anywhere.
2. You can use vanilla CSS or any library.
3. Try to host your site on Firebase (Netlify hosting will need some extra configurations)
    - Firebase Hosting Setup Complete Issue

4. Host your server-side application on Vercel. If needed, you can host somewhere else as well.
    - How to deploy a Node/Express server using Vercel CLI
    - Some Common Vercel Errors
5. Make Sure you deploy server-side and client-side on the first day. If you have any issues with hosting or GitHub push, please join the "Github and deploy" related support session.

What to submit :

- **Your assignment ID/Variant**
- **Your client-side code GitHub repository**
- **Your server-side code GitHub repository**
- **Your live website link**

<u>Some Guidelines:</u>

1. Do not waste much time on the website idea. Just spend 15-20 minutes deciding, find a sample website, and start working on it.
2. Do not waste much time finding the right image. You can always start with a simple idea. Make the website and then add different images.

3. Don't look at the overall task list. Just take one task at a time and do it. Once it's done, pick the next task. If you get stuck on a particular task, move on to the next task.
4. Stay calm, think before coding, and work sequentially. You will make it.
5. Be strategic about the electricity issue.
6. use `chatGPT` to generate JSON data. You can use chatGPT for Other purposes as well.

No Pain, No Gain
The most beautiful moments in life come after going through hardships and challenges.