Fullstack Dev - Test/Task Document

Following up on our interview process, we would like to take the next step and test your development skills!

Requirements

Please carefully read the requirements.

- Create the UI using your framework of choice. (**Vue, ReactJs**)
- Create a backend using **NodeJS** and **Express**.
- You could optionally implement a **Rest API** or **GraphQL** API to store and retrieve data in the database that you're most comfortable with.
- Integrate frontend and backend.
- Add Unit and Functional tests in the code backend and frontend apps.
- Dockerize both frontend and backend apps.
- We suggest you keep your code clean and up to mark with the best practices.
- Resources: IMDB, TMDB (you can use these for design inspirations, you can try to make it look better and unique)

Frontend:

- Routing
- Test Cases
- Authentication
- Screens
    - Authentication

    - Registration
    - Login
  - Home (Movie List)
  - Movie Details

Backend:

- DB Integration
- Routes
- Controllers
- Validations
- Services
- Models
- Test Cases

Assignment:

Develop a web application that includes a homepage showcasing a curated collection of movies. On the homepage, users will find an option to add new movies, but this functionality is exclusively available to authenticated users. When a user clicks on a movie title, they will be seamlessly directed to a dedicated movie detail page. On this page, they can both watch the movie's trailer and access a concise summary of the film, in addition to a list of user-generated movie reviews.

Unauthenticated users have the privilege to view the entire movie collection. However, the ability to contribute reviews for a movie is reserved for authenticated users only.

Movie Web App Flow:

- A user can sign up and Login to the app.
- A user will view all movies on the homepage in the form of cards with their posters.
- A user can view movies without login but to give reviews or to add movies he/she has to sign up and login.
- A user should be able to search movies by title
- Movies can be created and deleted by the respective user.
- There should be an edit movie feature that opens a modal. In that modal you can edit details such as: Movie name, release date etc.
- Reviews can be created, updated, and deleted by the respective user.
- No two movies with the same name can exist
- Create separate database tables for movie, user, and review and join them. (each entity must have at least three properties)
- Use aggregation to get movies with the most reviews and rank them according to that and display rank on the movie page. (suppose the movie with the most reviews has the highest rank)
- can add image urls for image fields (from sites like unsplash) to display movie posters and covers to make it look better.
- can use video links and play videos with iframe or video elements, uploading images and videos not necessary when creating movie posts.
- Write unit and functional test cases for both frontend and backend.
- Create a docker compose file to run the app.

Evaluation Criteria:

1. Usage of Custom CSS
2. Appropriate API Management
3. Clean UI
4. Intuitive UX
5. Use of Best Code Practices
6. Usage of Latest Frameworks.
7. Unit Testing

Submission Note:

1. Please take care of the presentation. Simple un-styled code with plain HTML buttons is undesirable.
2. Your application should be compressed in one self-contained Github repository with clear instructions on how to run the application in an accompanying readme.md file. Please provide **the GitHub link for the task**
    a. **Keep Repo Private**
    b. **Give access to**
        i. affan.rehman.cowlar@gmail.com
        ii. bilal.haider.cowlar@gmail.com
        iii. abdullah.yasir.cowlar@gmail.com
        iv. S-Ashjaa
        v. Sami-ullah-cowlar
        vi. KausarNaqvi
        vii. maryamnoor123
        viii. Umer-ilyas
        ix. Hafiz-usman-cowlar
        x. Usman-najmi-cowlar
        xi. daniyal-zafar-cowlar
3. Put sample.env in the repository.
4. Your submission must run on the Google Chrome browser.

5. Apart from your problem understanding, implementation, solution approach, syntax, and semantics, your overall project structure (components and layers) and tool use will also be considered during the evaluation.
6. Use the latest versions of the framework whatever you can use.
7. After completion of task, **Record a video** of the code where you explain:
    a. Your Understanding of the task
    b. How did you approach the task?
    c. Method & Considerations while writing the code
    d. End-to-end working of the Final Web App.
8. Upload the video on Google Drive (Make sure it is accessible by everyone.)
9. Add **Link of the video** to your final submission
10. Submissions can be made via email, “careers@cowlar.com” or Linkedin of the Talent Acquisition team member.